SINGAPORE MANAGEMENT UNIVERSITY

QF635: Market Microstructure and Algorithmic Trading

Project Topic: Trading System for Funding Rate Arbitrage

Project Report

Table of Contents

Executive Summary

This project aims to build an automated trading system for funding rate arbitrage. It aims to connect to Binance Spot and Futures Testnet, enabling real-time trading and efficient capital deployment while managing risk. This system has multiple modules for data retrieval and management, signal generation, trade execution, account and PnL monitoring, and risk management. The system is able to search for viable trades and execute them real time. It also has stop loss and manual kill switch in place for error handling. The primary challenges have been in terms of handling two different testnets with different margin system and the scarcity of signals. We also observed difference between testnet and mainnet, which can pose difficulties if system is deployed live. For future extensions we have considered implementing more dynamic systems for risk management and performance analysis. We have also taken into account various considerations if the system is further scaled up. Overall, this system establishes a foundation for an automated trading platform designed to capitalize on funding rate arbitrage opportunities. Its modular and flexible architecture allows for future scalability and enhancement as trading volume, complexity, and capital requirements grow.

1.Introduction

1.1. Background

The advancement of computing power and AI technologies has enabled new paradigms in for high frequency trading. However, trading at these speeds is beyond the capabilities of manual human traders. This brought on the innovation of the algorithmic trading systems.

Algorithmic trading - also called automated trading, black-box trading, or algo-trading - uses an algorithm (computer program) to place a trade. The defined sets of instructions are based on timing, price, quantity, or any mathematical model.

At its core an algorithmic trading system is based on 3 key principles: (1) buy the asset when there is a buy signal (2) sell the asset when there is a sell signal and (3) stop when there is an error. Any further actions by the system are developed based on this core foundation. Thus while we develop an algorithmic trading system we consider key system design questions, including– (1) what assets are to be traded, (2) what modules are to be developed, (3) how to query for the data and generate signals, (4) how to optimise trade execution, and (5) how to effectively manage risks.

In addition to the ability to trade at high frequency, an algorithmic trading system also has other advantages. If the system is developed accurately, then it often will execute a trade at the best possible price. Furthermore, trade order placement is instant with a high chance of execution at the desired levels. However, if a trade is not executed quickly enough, it may result in missed opportunities or losses. It can also allow for simultaneous automated checks on multiple market conditions. Moreover, as this negates the human traders' tendency to be swayed by emotional and psychological factors, it can greatly reduce the risk of human error. Additionally, an algorithmic trading system can incorporate back testing based on historical data. (Seth, Investopedia, 2023)

In this project, the aim is to develop a trading system that trades cryptocurrency spot and perpetuals, by exploiting funding rates of cryptocurrency perpetual. We aimed to investigate the process of constructing a trading system from the ground up, with the objective of understanding the various components and their interactions in a real-world setting. In particular, our interest was motivated by numerous anecdotal accounts of funding rate arbitrage strategies being deployed in individual portfolios to capitalize on inefficiencies within the rapidly burgeoning cryptocurrency market.

Funding rate arbitrage between spot and perpetual contracts refers to simultaneously executing two trades in opposite directions, with equal quantities and offsetting profits and losses, in both spot and perpetual contracts. Since perpetual contracts require periodic funding payments between long and short positions to keep prices anchored to the spot market, traders can potentially earn a steady return by holding the side that receives the funding. (“What Is Funding Rate Arbitrage?”, CoinGlass, 2025). The system is designed to automate the identification and execution of funding rate arbitrage opportunities while incorporating mechanisms to manage the associated risks. As the strategy is inherently delta-neutral with respect to the price movements of the underlying asset, it seeks to generate returns independently of directional market trends. A detailed discussion of the strategy's design, implementation, and potential limitations is presented in subsequent sections of this report.

From a market microstructure perspective, this strategy reflects how retail traders can respond to pricing inefficiencies, liquidity constraints, and incentive structures inherent in the cryptocurrency markets. Funding rates serve as a dynamic market mechanism that adjusts trader behaviour by redistributing payments between long and short positions, thereby incentivizing reversion to spot. As retail participants, by taking offsetting positions in spot and perpetuals, the strategy acts as a liquidity consumer in both legs. Additionally, this strategy effectively exploits the incentives designed to attract order flow and balance market pressure. The success of this strategy also hinges on minimizing adverse selection and execution costs. (Makarov and Schoar,2020)

1.2. Funding rate arbitrage

The funding rate arbitrage strategy is particularly of interest with recent academic attention on arbitrage in decentralized and crypto-native market structures. Makarov and Schoar (2020) investigated how inefficiencies between crypto exchanges, capital frictions and fragmented liquidity lead to persistent arbitrage opportunities. Similarly, Auer and Claessens (2018) highlight how funding costs and leverage constraints amplify pricing dislocations in crypto markets (Auer and Claessens, 2018). These studies support the hypothesis that funding-based arbitrage strategies can be both profitable and persistent under certain market conditions. This project seeks to develop a rule-based system that explores and execute the strategy in real time.

Funding rates are periodic payments made between traders who are long (buyers) and those who are short (sellers) in a perpetual futures market where funding rates is positive. Perpetual are futures contracts without any expiry date and settles every 8 hours. The funding rate can be either positive or negative, depending on the relationship between the perpetual contract price and the spot price.

Perpetual futures contracts typically require funding payments every 8 hours to align futures prices with the spot market prices. This creates opportunities to capture funding rate payments while remaining market neutral. The presence of futures basis also creates potential for capital gains when the gap converges. This gap convergence theoretically happens because when the price of a perpetual futures contract moves away from the underlying spot price, the funding rate will encourage traders to open positions in the opposite direction to receive the funding fees. When the funding rate is positive, long positions pay a funding fee to short positions. When the funding rate is negative, short positions pay a funding fee to long positions. This behaviour of traders brings the price back to the underlying spot price in efficient markets.

1.2.1. Funding rate calculation

The funding rate on Binance is calculated using the formula:

$$Funding\ Rate\ (F)\ =\ Premium\ Index\ (P)\ +\ clamp$$

$$clamp = Interest\ Rate\ -\ Premium\ Index\ (P);\ 0.05\%, -0.05\%.$$

- Positive Funding Rate: Longs pay shorts
- Negative Funding Rate: Shorts pay longs

This mechanism ensures that the funding rate remains within a capped range, helping to align the perpetual futures price with the spot market. When the futures price consistently trades above the spot price, the funding rate tends to be positive, meaning short positions receive payments from long positions. The expected profit from such a trade has two components: first, funding payments, where being short in a positively skewed market yields income through funding; and second, spread mean reversion, as futures and spot prices naturally tend to converge over time, potentially generating capital gains from price alignment.

$$Expected\ Returns = Funding\ Rate \times Notional + the\ expected\ contraction\ of\ the\ spread.$$

The simplest arbitrage approach is to exploit the price discrepancy between Futures and Spot is to trade both of them together and earn the funding rates (Fig 1). The arbitrageur anticipates that despite the price discrepancy the prices will realign due to funding mechanisms—specifically, that

$$|Futures - Spot| - E[spread]$$

will eventually converge.

A positive arbitrage can occur when Futures are trading at a premium (Futures > Spot Price), and the funding rate is positive: So we can sell Futures and buy Spot in a 1:1 ratio and remain delta neutral. Then we hold until the funding payment is collected or the spread compresses. A negative arbitrage is when Futures are trading at a discount (Futures < Spot Price), and the funding rate is negative. We then buy Futures and sell Spot in the same 1:1 ratio and collect funding payments and spread reversion.

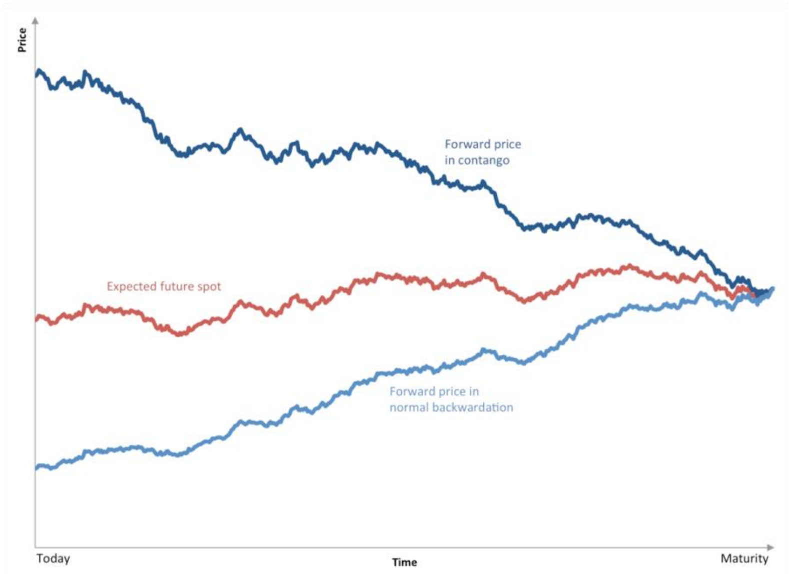


Fig 1: Shows forward price in contango (dark blue), expected future spot price (red line) and forward price in backwardation (light blue).

1.3. Binance Testnet

This strategy is run on Binance. As we are only testing it out we run it on the mock version of binance called Testnet. The trading system connects to the testnet via API. The testnet replicate many features of the live mainnet—such as API structures, order types, and market interfaces—but they do so in a simulated environment . Unlike the mainnet, where trades involve real capital and carry financial risk, testnets operate with valueless "paper funds." This allows us to safely and rigorously test our strategy and trading system stability without any monetary exposure.

However, this separation introduces critical limitations. For instance, the liquidity and order book behaviour in the testnet are artificial and do not reflect the actual depth, volatility, or slippage of live markets. This can lead to misleading assumptions about execution quality or latency. Moreover, certain edge cases not be accurately simulated in the testnet environment, especially for funding rate arbitrage strategy. The limitations of testnet and it how the discrepancy with the mainnet might affect our trading system have been discussed later.

Furthermore, on Binance Spot and Futures have separate testnet environment which must be taken into consideration while building the trading system. However, in the mainnet spot and futures are traded in one unified system. So any trading system built and tested on testnet should also be able to flexibly adjust to the unified architecture of the main net. This poses its own unique challenges. Another difference between the Binance testnets is that the Futures Testnet (Fig 2) provides a full exchange-like interface. In contrast, the Spot Testnet does not offer any graphical interface or market visualization tools. Hence, all trading and monitoring must be conducted through APIs, making it less user-friendly.

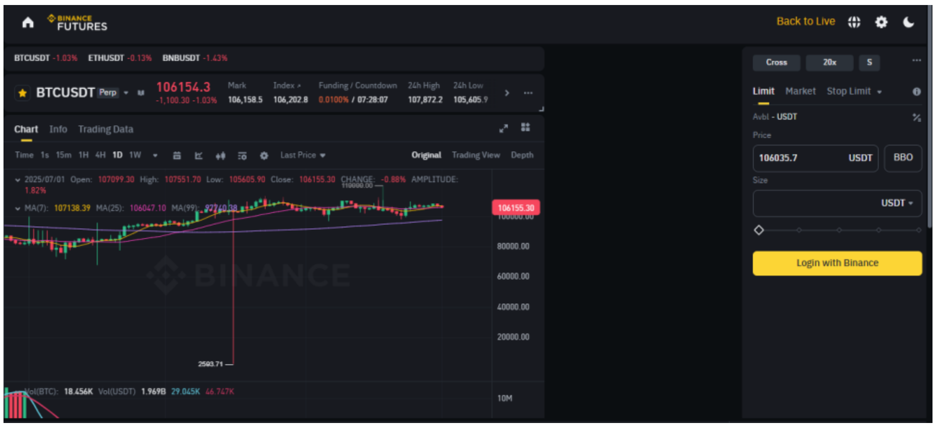


Fig 2: Binance Futures Testnet

2. Methodology

2.1. Execution overview

The trading system is built to execute a funding rate arbitrage strategy between the spot and perpetual futures markets on Binance. Developed in Python, it connects to the exchange via REST and WebSocket APIs. Its core logic is centered on identifying funding rate discrepancies and entering opposing long and short positions across the two products. This allows the strategy to profit from funding flows in a market-neutral way, limiting directional risk while exploiting inefficiencies in perpetual futures pricing.

On a high-level the system operates as a continuous, asynchronous loop that runs every second, enabling real-time response to market conditions (Fig 4).

It continuously processes real-time data, executes trading logic, monitors open positions, and prints system status. This structure enables efficient concurrency and fast responsiveness. The loop integrates real-time data ingestion, order placement, exit condition checks, and account updates in a single cohesive workflow.

To ensure compatibility with Binance's exchange constraints, the system rigorously applies rounding logic to both prices and quantities. Trade prices are adjusted based on the instrument's tick size using one of three methods: floor, ceil, or round. Order quantities are similarly rounded down to the nearest valid step size. These adjustments ensure all submitted orders conform to Binance's requirements, avoiding slippage or rejection due to invalid formatting. This design choice also simplifies downstream logic by removing the need for redundant error handling and corrections.

Capital allocation and position management are governed by a strict risk control framework. The system sets a fixed cap on the number of concurrent positions (maximum 20% of total capital can be allocated to each position) and evenly divides available capital across these slots. Before executing any trade, it checks both the number of open positions and available margin to ensure adherence to this constraint. This approach guards against over-leveraging, ensures margin sufficiency, and acts as a throttle on trading frequency, enabling controlled and sustainable deployment of capital.

Trade execution begins by identifying the most favourable arbitrage opportunity based on a computed edge metric, which accounts for funding rate imbalances and market conditions. Once a candidate opportunity is selected, it passes through a series of validation filters, including funding regime, market depth, and execution feasibility. If the opportunity meets all criteria, the system calculates an appropriate order size based on capital allocation, adjusts the trade parameters for exchange compliance, and prepares the order legs.

The system supports two primary trade directions: shorting perpetual futures and buying spot assets when the funding rate is positive or the inverse when the funding rate is negative. Entry prices for both legs are determined and adjusted to match valid tick sizes. Limit orders are then submitted to the exchange, with all relevant trade metadata—including fees, position size, entry price, and capital used—recorded for monitoring and evaluation.

Beyond immediate execution, the system also considers anticipated funding regime shifts and may take early positions in expectation of favourable future changes. This predictive component enhances strategic depth by allowing the system to position ahead of known funding windows, capturing edge before market adjustments occur.

Following execution, all open positions are logged and tracked in a portfolio system that records mark-to-market PnL in real time. This portfolio serves as the foundation for monitoring exposure, calculating returns, and making exit decisions. Exit logic is integrated into the main loop and is continuously evaluated based on predefined profit targets, loss thresholds, and funding rate decay.

The overall trading flow is represented as a continuous pipeline (Fig 3) that begins with strategy formulation and opportunity identification. This phase incorporates real-time market data, historical analysis, and funding rate projections to guide decision-making. It flows into execution optimization, where leverage, slippage, fees, and margin requirements are factored into trade construction. Execution and post-order logic handle trade placement, confirmation, and initial monitoring. This closed feedback loop supports iterative improvements and adaptation to changing market conditions, while also accounting for operational risks such as exchange connectivity or API failures. The logic of each step is discussed further in detail in the subsequent subsections.

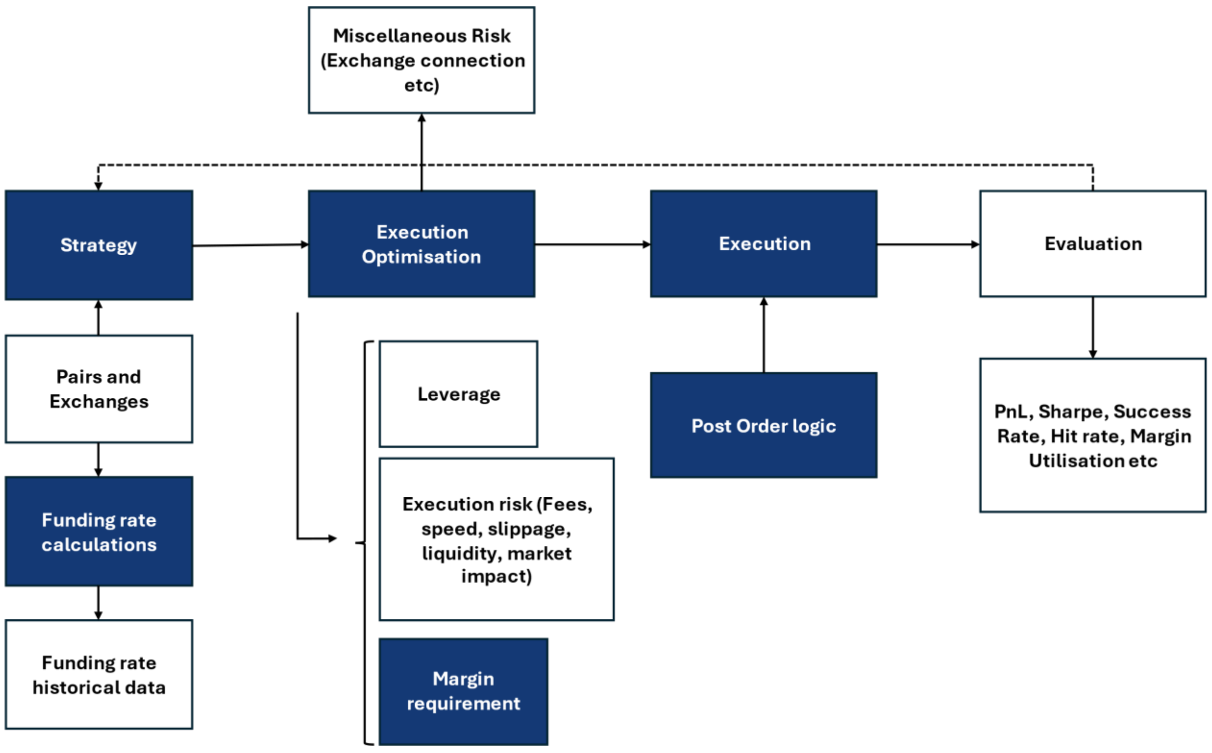

Miscellaneous Risk (Exchange connection etc)
Strategy
Execution Optimisation
Execution
Evaluation
Pairs and Exchanges
Leverage
Post Order logic
PnL, Sharpe, Success Rate, Hit rate, Margin Utilisation etc
Execution risk (Fees, speed, slippage, liquidity, market impact)
Funding rate calculations
Funding rate historical data
Margin requirement

Fig 3: Trading system architecture and logic flow (already implemented – blue ; yet to implement – white)

2.1.1. Data retrieval and management

This project aims to develop a trading system that can connect to both Binance's Spot and Futures Testnet environments to exploit the funding rate. The system is designed for efficient and scalable data handling across multiple trading symbols, with a strong emphasis on modularity. Functionality is separated between Spot and Futures, facilitating independent debugging, testing, and maintenance. Since the Spot and Futures Testnet environments are distinct, they require separate API connections. This is distinct to testnet environment due to limitations of the system. In reality, one can trade both Futures and Spot on the same API.

API access for both Spot and Futures Testnet endpoints is provided by Binance. Authentication is securely handled via HMAC-SHA256 signatures generated using secret keys and timestamps. For Futures market data, we utilize the official Binance Futures Python SDK, while Spot market interactions are constructed manually using authenticated REST API calls. API credentials and configuration parameters are externalized into a secure .env or dedicated configuration file, to ensure secure communication. All credentials and keys are initialized at the start of the script, while endpoint URLs and request headers are organized into a modular structure. This ensures that sensitive details remain secure and that components are reusable across API requests. Furthermore, rate limits are considered and mitigated through the use of combined WebSocket streams, thereby reducing the reliance on REST API calls.

Once connections to both Testnet environments are established, we query account balances for both Futures and Spot markets. Two independent modules were developed to retrieve margin balances for perpetual contracts and spot pairs, respectively. These balances are used to set the initial trading capital by selecting the minimum of the two values. This conservative allocation approach ensures that the algorithm adheres to capital constraints and prevents overexposure due to uneven balances.

Real-time processing is achieved using asynchronous functions that enable non-blocking operations, allowing concurrent management of multiple data streams. A dedicated module monitors several trading pairs simultaneously. Trading symbols are dynamically loaded from a CSV file (binancespot.csv), parsed, and used to construct WebSocket URLs for real-time data acquisition. The system retrieves spot book ticker data, perpetual futures order book data, and mark price streams, including funding rate information. Persistent WebSocket connections are implemented with automatic reconnection logic in the event of network disruptions. The system maintains a real-time store of key market information, including the latest bid and ask prices for spot and futures, the current mark price, funding rate, and time remaining until the next funding interval. Real-time logging functionality tracks both data flow and potential runtime errors.

Additionally, the system continuously extracts trading constraints directly from Binance's exchange metadata. These include price and quantity filters such as tick size, step size, and minimum notional quantity. By referencing these filters, the system ensures that submitted orders conform to exchange rules, thereby reducing the risk of rejections. The smallest allowable price increment for each trading symbol is retrieved and applied dynamically to all pricing logic. This level of validation enhances both order formatting and execution reliability.

The use of WebSocket streams for real-time updates minimizes latency compared to polling-based REST requests. Moreover, by accounting for exchange-level constraints such as tick size and step size, the system mitigates risks related to order rounding errors, slippage, and execution mismatches. Accurate alignment of funding timestamps and mark prices ensures the system reacts in synchrony with other market participants, preserving informational fairness and avoiding execution lag.

Despite the system's current robustness, there are several future improvements. Enhanced error-handling mechanisms such as retry limits, exponential backoff strategies, and more granular exception logging should be integrated to improve overall system resilience. Monitoring WebSocket latency and measuring response time variability will help detect synchronization issues between spot and futures feeds.

In real-world deployments, data integrity and latency fairness are especially crucial in crypto markets, where fragmented infrastructure and differing network paths may create informational asymmetries. Retail traders are particularly vulnerable to these effects, as even slight data delays can lead to missed arbitrage windows or mistimed orders. To address this, future versions of the pipeline should implement checksum validation, message sequence tracking, and clock synchronization with Binance's server. These enhancements will help preserve market parity and reduce susceptibility to adverse selection in fast-moving order books.(Budish, E., Cramton, P., & Shim, J., 2015)

To further improve scalability under higher throughput, WebSocket listeners can be distributed across multiple threads or enhanced with asyncio queues for prioritized processing. These changes will allow the system to scale with minimal performance degradation, particularly when monitoring dozens of trading pairs simultaneously.

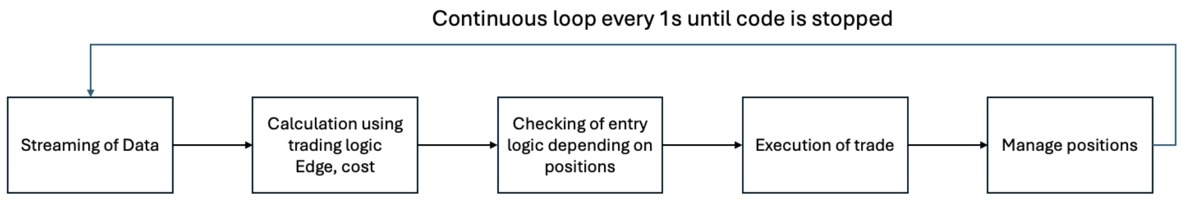

Continuous loop every 1s until code is stopped
Streaming of Data
Calculation using trading logic Edge, cost
Checking of entry logic depending on positions
Execution of trade
Manage positions

Fig 4: high level overview of the execution logic and data retrieval and management

2.1.2. Setting up the execution modules

The execution system is composed of two primary components: the ‘Futures Trader’ module, which interfaces with the Binance Futures Testnet, and the ‘Spot Trader’ module, which connects to the Binance Spot Testnet. These modules are implemented using Python and adhere to object-oriented programming principles, allowing for clean abstraction, extensibility, and testability. Their primary functions are automated order execution, and portfolio and position monitoring.

The ‘Futures Trader’ class is responsible for crypto-currency perpetual contracts, while the Spot Trader class manages execution in the spot market. Each class was designed to encapsulate both REST API interactions and WebSocket listeners for live feedback, thereby reducing latency and enhancing responsiveness to real-time market data. The modular separation between the two enables independent debugging and efficient resource scaling, especially under high-frequency trading conditions.

During initialization, both the Futures Trader and Spot Trade class connect to their respective Binance Testnet accounts using the API details. Both classes use Python’s logging module to display and record messages when connections are successfully established or orders are successfully placed. Conversely, if an API call fails due to invalid input, insufficient funds, or connectivity issues, the error is logged in detail, including the exception message from Binance. The exception handling ensures system continuity, and aids in debugging, without interrupting the trading workflow.

Users can place market or limit orders through these classes. For market orders, the required parameters are the trading symbol (e.g., BTCUSDT), the order side (BUY or SELL), and the quantity. If the API call is successful, the order is executed and logged; otherwise, the failure is caught and logged, and None is returned. While placing limit orders, in addition to the three orders for market order, user also need to define a specific price at which they are willing to buy or sell an asset.

Furthermore, the user is recommended to set Immediate or Cancel (IOC) condition. In volatile crypto markets with wide spreads, IOC is particularly useful to minimize slippage and avoid passive exposure. Moreover, the use of IOC orders helps minimize adverse selection by ensuring that only aggressively priced orders are filled, reducing exposure to latency arbitrage (Budish et al., 2015). Both order methods are wrapped in try-except blocks for error handling, ensuring that any failure in order submission is logged and does not cause the program to crash unexpectedly.

In addition to the advantages of IOC mentioned we considered other execution types (Table 1)

Order Combination	**Pros**	**Cons**
All Fill and Kill Limit Orders	Minimizes trading fees Able to get desired price No gap risk	High risk of partial fills, leaving unhedged exposure
All Market Orders	Guarantees full execution	High slippage risk, especially in volatile conditions
Hybrid: Limit Order + Market Order	Balance cost and execution certainty	Requires latency optimization, high risk of partial fills on the second leg

Table 1: Execution types considered other than IOC and their advantages and disadvantages

For funding rate arbitrage strategy when there is a signal generated for execution of a trade position, it should be done immediately due to the relatively high frequency that we are operating in. Thus using an IOC limit order to prevents the orders from staying on the order book. This ensures that we are getting into both legs of the position simultaneously when a signal is generated, preventing erroneous trades.

The Future Trade class is also able to extract real-time insight into open futures positions. The method iterates through each trading position and filters out empty ones (positions with zero quantity). For active positions, it extracts the trading symbol, the amount of the position, the direction (long or short), the entry price, the current mark price, and the unrealized PnL. By actively monitoring unrealized PnL and position sizes, the system implicitly supports inventory risk management. This data is packaged into a structured list of dictionaries, which can be easily processed or displayed in dashboards.

An interesting enhancement to this would be a user-friendly interface or dashboard that could visualize positions, balances, and PnL metrics for both perpetuals and spot trade simultaneously. Furthermore, logging execution prices, order submission timestamps, and bid-ask spreads would allow for the implementation of transaction cost analysis (TCA), making it possible to evaluate order quality and improve execution algorithms (Almgren &

Chriss, 2001). However, Binance's spot testnet infrastructure is lacking, and a similar view of ongoing metrics cannot be shown.

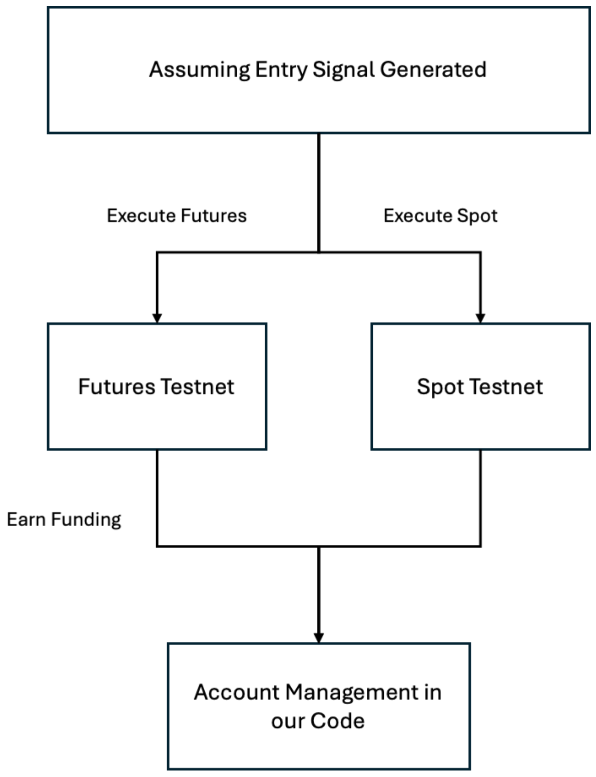


Fig 5: Execution logic

2.1.3. Signal generation

The funding rate arbitrage strategy is grounded in the principle of market neutrality. Simultaneous positions in the perpetual and spot markets are used to extract value from pricing inefficiencies, rather than from directional price predictions. While portions of this section may overlap with earlier discussions in section 1.2.1, they are included again for emphasis and contextual clarity, particularly as they relate to the logic behind signal generation and trade entry conditions.

At its core, the system will continuously monitor the price relationship between the Binance Spot and Futures markets for a given trading pair (e.g., ETH/USDT). The trading decision is fundamentally based on a direct comparison between the Futures Bid and Spot Ask prices. A situation where the Futures contract is priced higher than the Spot (i.e., Futures > Spot) typically suggests a market expectation of future price increases. However as explained earlier, in the context of perpetual swaps, such discrepancies often normalize due to arbitrage and the funding mechanism that incentivizes price alignment.

The funding rate itself is derived from Binance's official formula:

$$F = P + clamp(r - P, +0.05\%, -0.05\%)$$

Here, F represents the funding rate, P is the Premium Index—which reflects the average difference between the perpetual Futures and Spot prices over time—and r is the Interest Rate, typically set at 0.01%. The clamp function ensures the adjustment remains within a ±0.05% band, thus stabilizing the funding mechanism against extreme swings.

Under standard market conditions, when $P \in [-0.04\%, 0.06\%]$ the funding rate defaults to approximately 0.01%. Outside this band, the funding rate becomes more reactive and is adjusted upward or downward to reflect the relative market pressure in the Futures book. This rate is updated every eight hours and becomes the dominant factor in arbitrage strategy.

When the Futures price exceeds the Spot and the funding rate is positive, this forms an ideal arbitrage condition. Upon detecting such a price differential, the system will evaluate the current funding rate. If the funding rate is positive, meaning longs pay shorts, the system employs a cost-benefit framework. This framework quantifies expected returns against incurred trading expenses. For illustration, consider a unit trade in ETH/USDT with entry prices x (Futures) and y (Spot).

From a profit perspective, there are two primary sources of expected gain:

1. Funding Payments: If the position is held through a funding interval, the trader receives a payment equal to the funding rate multiplied by the notional value of the Futures short position, i.e.,

$$Funding\ payments = \ Funding\ Rate\ \times\ notional.$$

2. Price Realignment (Convergence Profit): The assumption of market efficiency suggests that the price discrepancy between Spot and Futures will eventually converge. If this occurs before the funding payment, the system can exit the position early and profit from the narrowing spread. The net expected profit from this

realignment is $|x - y| - E[spread]$, where E[spread] reflects the average bid-ask slippage encountered when closing the position in both books.

The trading cost component is driven by Binance's fee structure. For taker trades, the Spot market incurs a 0.1% fee, while the Futures market incurs a 0.05% fee. Therefore, entering the position results in a cost of $0.1\% \times y + 0.05\% \times x$. If the position is closed at prices x' *and* y', an additional exit cost of $0.1\% \times y' + 0.05\% \times x'$ is incurred. These costs are estimated at the time of entry using projected future prices or mid-market assumptions.

If the cumulative expected profit (from funding payments and/or price convergence) exceeds these transaction costs, the system proceeds with the trade. Otherwise, it abstains, preserving capital and maintaining trade discipline. This ensures that any arbitrage attempt results in a net positive return after costs are deducted.

Only when the funding rate exceeds the total cost threshold does the system take this as a signal to enter a trade. It then opens long position in the Spot market and a short position in the Futures market. This structure is neutral to directional price changes but positively exposed to the funding payment. In simpler words the short Futures position accrues the funding payment. If the funding rate is insufficient to cover the trading costs, no trade signal will be generated.

The signal generation logic can be expanded to also account for negative funding rates to generate additional trading signal. An exploratory idea is that if the funding rate is negative, so the shorts pay longs, the system should not automatically ignore the opportunity. Instead, it should conduct a relative analysis comparing the current negative funding rate to the worst-case historical funding rate observed over a specific time horizon, adjusted by the transaction fees. If the current rate still offers a favourable return compared to this benchmark, the system may execute the trade under the same long Spot/short Futures structure. This additional signal generation will allow the system to remain responsive and opportunistic under varying market conditions, rather than relying on a simplistic positive-rate filter.

Nonetheless, complications arise when the Futures price is at a premium while the funding rate is still negative. This suggests that the Futures market had recently traded at a discount and the funding rate has not yet adjusted. If this mispricing persists into the next update cycle, the funding rate is likely to flip to a positive value.

To capture such latent opportunities, the system employs a time-to-funding filter. If the Futures price exceeds Spot but the funding rate is negative, the system checks the time remaining until the next funding update. If this duration is sufficiently long—empirically defined as more than six hours—the system may choose to enter the trade pre-emptively. This anticipatory behaviour is based on the assumption that the futures premium is a leading indicator of a forthcoming positive funding rate. The same logic applies in reverse for scenarios involving a Spot premium and positive funding rate, where the trader may act ahead of a likely reversal.

While it is common to assume that the position should be held until the funding is paid out, this is not always the most efficient approach. The system incorporates an adaptive exit strategy to maximize returns while minimizing risk exposure.

One primary reason for early exit is price convergence before the funding window. If the price spread between Spot and Futures narrows significantly soon after trade entry, the arbitrage opportunity effectively disappears. At this point, holding the position exposes the trader to additional market risk without guaranteed returns. Therefore, the system opts to close the position and realize the spread profit immediately.

Another factor influencing early exits is execution risk, particularly around funding intervals. As the time to funding approaches, order book conditions may change. Liquidity often becomes fragmented or thins out due to increased trading activity or strategic behaviour by other arbitrageurs. This can make it difficult to exit the position at favourable prices. The system therefore monitors order book depth, spread width, and execution latency as part of its real-time risk management module.

When liquidity deteriorates or volatility increases beyond certain thresholds, the system prioritizes immediate closure over waiting for the funding pay-out. This risk-averse posture reduces the likelihood of slippage or adverse fills and is especially important when managing larger position sizes or operating in lower-liquidity altcoin markets.

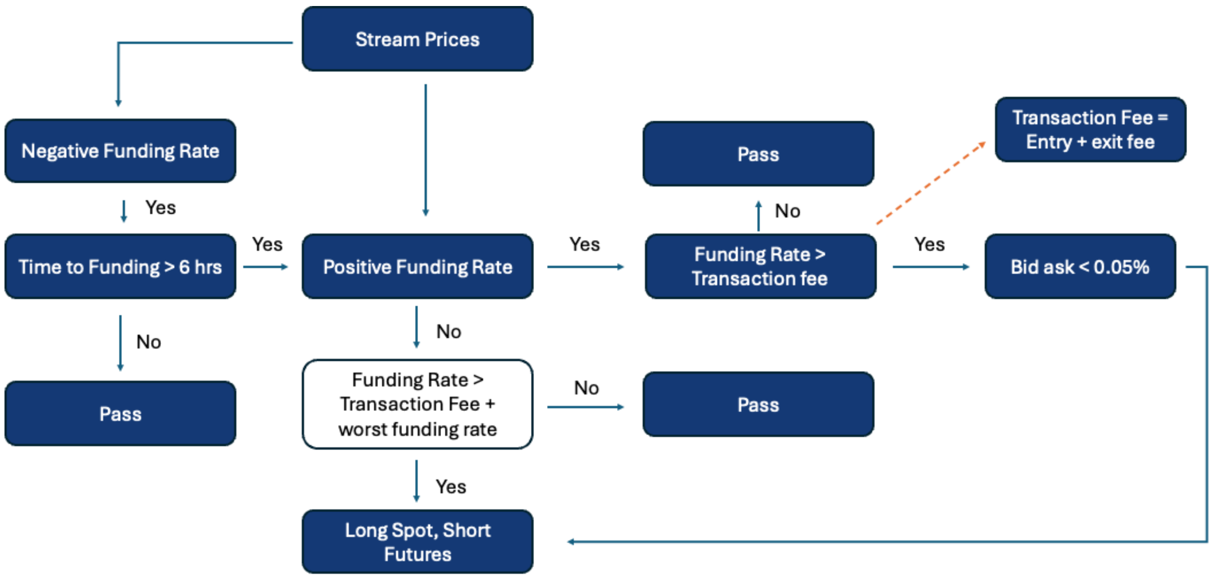


Fig 6: Signal Generation Logic

2.1.3.1. Signal generation system

The discrepancy between spot and future prices are called 'edges.' These edges arise due to momentary inefficiencies or liquidity gaps in either market. For trading signal generation, we created an edge detection engine that evaluates trading opportunities based on these discrepancies between spot and futures prices. The engine continuously analyses the bid and ask prices for both spot and perpetual futures pairs. It also incorporates trading costs such as taker fees in its computations, with a spot fee rate of 0.1% and a perpetual futures taker fee of 0.05%. These fees are applied to calculate net executable prices. This ensures that the estimated arbitrage potential closely reflects the actual profit or loss a trader would realize in live market conditions.

It is to be noted that the taker fee for Spot and Futures is considerably higher than the maker fee. This is to incentivise participants to be price makers and add to the order book with limit orders and increase liquidity. However, for the purpose of our project, we intentionally used the taker fee as the criteria due to acknowledgement of limited infrastructure and latency. As a result, we compensated for this limitation with a higher barrier of entry, giving us a larger margin of safety.

- The edge is calculated as the Bid of Futures – Offer of Spot after accounting for fees, still remains positive with positive funding rate.

$$Edge = Bid(Futures) - Offer(Spot) - Fees.$$

- Conversely, $Edge = Bid(Spot) - Offer(Futures) - Fees$ for Futures after accounting for fees, yields negative rate.
- The futures–spot basis is also expressed as a percentage of the spot price:
  $$Basis\ (\%) = (Futures\ Price - Spot\ Price)\ /\ Spot\ Price \times 100,$$
  providing a normalized measure of the spread's relative magnitude.

The system’s primary analytic function is executed repeatedly in an infinite asynchronous loop. Each iteration of this function begins by filtering out any symbols with incomplete market data. If any of the required prices—spot bid, spot ask, perpetual bid, or perpetual ask—are missing, the corresponding symbol is immediately excluded from analysis. This prevents system from generating false signals due to partial or stale data feeds. In the first step of filtering it avoids duplicate positions in the same symbol. Furthermore, it incorporates a dynamic filtering mechanism that discards symbols exhibiting excessively wide spreads. If either the spot or perpetual bid-ask spread exceeds 0.5% of the mid-price, the symbol is considered too illiquid or volatile for reliable arbitrage execution and is consequently skipped. These allow to maintain the integrity of the signals and avoiding slippage-induced losses.

The 0.5% spread threshold used in liquidity filtering was determined empirically by observing historical market behaviour across spot and perpetual instruments. Wider spreads often correlate with low liquidity or heightened volatility—both of which undermine arbitrage reliability. However, this threshold could be dynamically tuned based on market regime classification (e.g., during high-volatility events), allowing the system to adaptively balance between aggressiveness and conservativeness in edge selection.

Once a symbol passes the initial data quality checks, the system calculates the mid-price for both the spot and perpetual instruments. These mid-prices represent the theoretical "fair value”. The mid prices are derived by averaging the bid and ask prices. If there is limited liquidity, for instance only bids are present, then the mid-price falls back to the available quote. This ensures that the engine can continue operating even in sparsely populated order books, although such cases are rare in liquid markets.

A key component of edge analysis is the fee-adjusted price calculation. Spot ask and bid prices are adjusted upward and downward, respectively, to incorporate spot trading fees.

Similarly, perpetual bid and ask prices are adjusted for taker fees. Using these adjusted prices, the system evaluates two arbitrage directions:

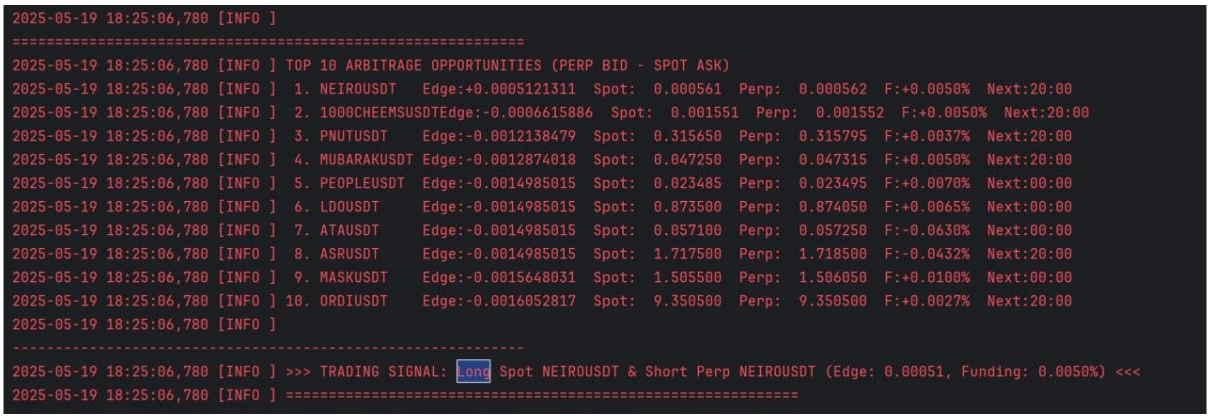
- Sell perpetual and buy spot (basis trade when $perp > spot$)
- Buy perpetual and sell spot (reverse basis trade when $perp < spot$)

The system calculates the edge as the relative difference between these net execution prices, divided by the cost basis, and retains the larger of the two values. This ensures that only the most profitable trade direction is selected in real time. The pairs are sorted in descending order by edge magnitude. The pair with the highest edge is logged along with details about the funding rate and when the next funding event will occur. This output provides a real-time snapshot of the most attractive arbitrage opportunity available at that moment.

Exiting an open position in a funding rate arbitrage strategy is governed by two key triggers: whether the market edge has reverted—signalling that the arbitrage opportunity has dissipated—and whether the anticipated funding payment has been collected. When both conditions are satisfied, the system promptly exits by placing market orders to close both legs of the position and logs the outcome. While it is common to assume that positions should be held until the funding fee is paid (typically every eight hours), this is not always optimal. Price convergence between futures and spot markets can occur well before the funding event, allowing traders to lock in profits early without waiting. For instance, if the futures contract was trading at a premium and the spread narrows shortly after entry, the opportunity effectively closes, making an early exit advantageous. Additionally, holding a position longer increases exposure to execution risks and potential liquidity issues. As market depth fluctuates, especially near funding times, exiting at favourable prices can become more difficult, reinforcing the need for a dynamic, condition-driven exit strategy.

Parallel to this edge detection engine, the system's real-time data ingestion layer, powered by asynchronous WebSocket clients, continuously streams spot order books, futures order books, and futures mark price feeds from Binance. The data from these streams are referenced by the edge computation logic. The asynchronous nature of the WebSocket management ensures that data ingestion operates concurrently with edge computation. Thus minimizing latency and maximizing responsiveness to changing market conditions.

The main execution function combines these components in an event loop. It begins by initiating all WebSocket subscriptions, retrieving the market data into the system. After a brief time lapse, it enters an infinite loop where it checks for edge (trading opportunities) every second (Fig 7). This maintains high temporal resolution and can respond to fleeting arbitrage windows. A manual interruption signal by the users leads to system cancelling any outstanding tasks and can effectively be used as a kill switch.



```
2025-05-19 18:25:06,780 [INFO ]
============================================================
2025-05-19 18:25:06,780 [INFO ] TOP 10 ARBITRAGE OPPORTUNITIES (PERP BID - SPOT ASK)
2025-05-19 18:25:06,780 [INFO ]  1. NEIROUSDT   Edge:+0.0005121311  Spot:  0.000561  Perp:  0.000562  F:+0.0050%  Next:20:00
2025-05-19 18:25:06,780 [INFO ]  2. 1000CHEEMSUSDTEdge:-0.0006615886  Spot:  0.001551  Perp:  0.001552  F:+0.0050%  Next:20:00
2025-05-19 18:25:06,780 [INFO ]  3. PNUTUSDT    Edge:-0.0012138479  Spot:  0.315650  Perp:  0.315795  F:+0.0037%  Next:20:00
2025-05-19 18:25:06,780 [INFO ]  4. MUBARAKUSDT Edge:-0.0012874018  Spot:  0.047250  Perp:  0.047315  F:+0.0050%  Next:20:00
2025-05-19 18:25:06,780 [INFO ]  5. PEOPLEUSDT  Edge:-0.0014985015  Spot:  0.023485  Perp:  0.023495  F:+0.0070%  Next:00:00
2025-05-19 18:25:06,780 [INFO ]  6. LDOUSDT     Edge:-0.0014985015  Spot:  0.873500  Perp:  0.874050  F:+0.0065%  Next:00:00
2025-05-19 18:25:06,780 [INFO ]  7. ATAUSDT     Edge:-0.0014985015  Spot:  0.057100  Perp:  0.057250  F:-0.0630%  Next:00:00
2025-05-19 18:25:06,780 [INFO ]  8. ASRUSDT     Edge:-0.0014985015  Spot:  1.717500  Perp:  1.718500  F:-0.0432%  Next:20:00
2025-05-19 18:25:06,780 [INFO ]  9. MASKUSDT    Edge:-0.0015648031  Spot:  1.505500  Perp:  1.506050  F:+0.0100%  Next:00:00
2025-05-19 18:25:06,780 [INFO ] 10. ORDIUSDT    Edge:-0.0016052817  Spot:  9.350500  Perp:  9.350500  F:+0.0027%  Next:20:00
2025-05-19 18:25:06,780 [INFO ]
------------------------------------------------------------
2025-05-19 18:25:06,780 [INFO ] >>> TRADING SIGNAL: Long Spot NEIROUSDT & Short Perp NEIROUSDT (Edge: 0.00051, Funding: 0.0050%) <<<
2025-05-19 18:25:06,780 [INFO ] ============================================================
```

Fig 7: System searching for potential trading opportunities (high edge)

2.1.3.2. Stop loss strategy

Although the strategy is market-neutral, it is not risk-free. Adverse market conditions—such as delayed convergence, unexpected funding regime shifts, or deteriorating liquidity—can erode expected returns or cause losses. To mitigate such risks, the system needs to incorporate a stop loss mechanism that dynamically responds to both price divergence and market conditions**.**

Currently the stop loss is triggered under the following condition:

1. <u>Realized Loss Breach After Accounting for Fees:</u>
   The system monitors the mark-to-market unrealized PnL in real time. If the cumulative loss (including estimated exit costs) exceeds a predefined percentage of capital allocated to the trade (20% of the total capital per position) the position is closed immediately to preserve capital.

$$UnrealizedPnL - ExitFees < StopLossThreshold$$

However, we want to also incorporate the following logic and conditions for stop loss in future enhancements:

1. Adverse Spread Divergence Beyond Expected Thresholds:
   If the absolute spread between the Futures and Spot positions widens beyond a statistically defined bound (e.g., beyond historical 95th percentile spread), the position is flagged for exit. This condition protects against scenarios where convergence fails to materialize and divergence intensifies, leading to mark-to-market losses.

$$| Pfutures - Pspot | > MaxExpectedSpread$$

2. Volatility and Execution Risk Escalation:
   When the system detects rising volatility or declining order book depth—e.g., widened bid-ask spreads, lower market depth at top-of-book, or increased latency—the stop loss condition may be pre-emptively triggered to prevent slippage or poor execution. This is particularly important during the approach to funding intervals or in thinly traded markets.
3. Funding Outlook Deterioration:
   If the system entered a position in anticipation of a future funding rate flip (e.g., negative to positive), but market indicators (such as basis compression or declining futures premium) begin to contradict that thesis, the trade is unwound early. This protects against being caught on the wrong side of a funding adjustment cycle.
4. Liquidity Exit Failures:
   If multiple attempts to unwind a position result in partial fills or failed executions due to insufficient liquidity, the system may enforce a full stop loss using market orders, accepting temporary cost impact in exchange for risk containment.

2.1.4. Trade ledger

Post signal generation and trade execution, the trades are logged. The Trade Ledger module acts as an internal logbook for tracking executed and attempted trades. This module is crucial for maintaining a reliable and structured record of all trading activity. It is vital not only for performance monitoring but also for tasks such as auditing, debugging, and evaluating strategies.

This module stores the trading symbol entry price, trade direction, stop-loss level, take-profit level, the time the trade was executed, and the "edge". New trades are also logged along with these details. The system uses Python's logging method to also store the timestamp of when the trade was executed and recorded. This assists during post-trade analysis or when unexpected results occur.

Users can generate either the full trade history or filter out by a specific pair or instrument. This is particularly helpful when trying to evaluate the performance of a single asset within a larger trading strategy. The subsystem can also retrieve the most recent trade entry, returning None if no trades have been recorded yet. This allows the system to make decisions based on the outcome or details of the most recent trade, such as avoiding repeated entries or adjusting strategy parameters in response to recent execution results.

Furthermore, the subsystem also generates a summarised and concise report of the trading history. It displays time of execution, trading symbol, direction of the trade, entry price, stop-loss and take-profit values, and the edge score. It is created to generate a quick snapshot of the records and trading history. It can reveal patterns such as frequent stop-loss activations or unusually low edge values, which in turn can help developers identify problems or opportunities for improvement in their trading logic. Although the current version prints directly to the console, this method could easily be adapted to output to a log file or display in a web-based dashboard.

2.1.5. Portfolio Management and PnL tracking system for arbitrage trading

In addition to arbitrage detection, we also monitored the accounts for both spot and futures. A portfolio management system was designed to facilitate account and position management, risk control, and performance evaluation in a systematic and scalable way.

The portfolio management component oversees the open arbitrage positions. It regulates the exposure across the trading strategy by enforcing constraints on both the number of active trades and the capital allocated to each position. It takes in total initial capital as the main parameter and an optional parameter defining the maximum number of concurrent open positions allowed. From these inputs, it computes the maximum notional allocation permitted per position, ensuring that capital is distributed evenly and risk is contained.

On the backend, it tracks open positions and logs the position to a corresponding trading symbol and its associated trade details. These details typically entails both legs of the arbitrage position. The manager provides interfaces to add new positions, remove existing ones, and query the state of individual trades. These allows for dynamic position management.

In addition to these, another key function of the portfolio management module is its ability to evaluate whether a new position can be opened under the current portfolio conditions. It enforces both position count and notional value limits. If the signal is constantly recurring for the same pair, it will still prohibit the execution of the trade, in order to prevent overexposure. Additionally, the overall rule ensures that the number of active trades does not exceed the configured maximum, and that the proposed notional value of a new position remains within the allowable budget. This safeguard is to prevent overexposure, particularly during periods of high signal density.

Furthermore, this system also provides a real-time estimation of unrealized profit and loss (PnL) for any open position. This function retrieves the relevant trade details for a given symbol and calculates the spot and futures PnL separately. The spot leg's PnL is determined by comparing the current market price—either bid or ask, depending on the trade direction—with the execution price, then multiplying the result by the quantity held. The same process is applied to the futures leg. The function indicates trade directionality by flipping the PnL sign for short positions, ensuring that gains and losses are computed correctly regardless of the market side. The total PnL is returned as a rounded sum of the spot and futures components.

$$Exit - Entry + Funding = Profit\ or\ Loss$$

$$Entry\ Cost\colon Spot\ ask * 1.1\% - \ Futures\ Bid * 1.05\%$$

$$Exit\ Cost\colon Futures\ Ask * 1.05\% - \ Spot\ Bid * 1.1\%$$

Formulas for short Futures Long spot Trade.

Complementing the portfolio manager is the Trade ledger system which records all the historical trade activity. As mentioned previously this system records both all trades ever executed and trades that have been closed. It also logs new trades as they are opened and tracks them throughout their lifecycle. Each trade recorded in the log contains full metadata about the spot and futures legs, including side, entry price, quantity, and associated symbol.

It also allows for manual or automatic closure of positions by providing the final spot and futures prices at the time of closure. Upon locating the corresponding trade in the history that has not yet been marked as closed, the method computes final realized PnL. Similar to the real-time estimator, the spot and futures components are calculated independently, accounting for whether the initial trade was long or short. The total final PnL is then stored in the records, along with a timestamp marking the closure event. The trade is then moved from the open to the closed records.

The relationship between the two systems is symbiotic. The portfolio manager system enforces forward-looking constraints and manages real-time exposure, while the supplementary system acts as a backward-looking ledger that captures the outcomes of strategic decisions. For instance, risk metrics such as drawdown, Sharpe ratio, or position heatmaps can be integrated into the supplementary system , while trade sizing algorithms or automated liquidation rules can be embedded within the main system. However, for the purpose of this project, as the focus was on execution of algorithm, we refrained from running this over prolonged periods and recording such metrics. Moreover, since the Spot and Futures Testnet have different interfaces, there is a slight inaccuracy on this part. Nonethless, the subsequent addition of these should create a comprehensive framework for managing, monitoring, and evaluating arbitrage trading strategies in a disciplined and methodical manner.

2.1.5.1. Account position

As a foundational element of this framework, real-time visibility into account positions is essential for effective PnL tracking. For the futures wallet balances, available margin, and unrealized profit and loss across all positions are fetched from the Binance Futures TestNet. Similarly, we retrieve account information of our Spot Testnet. We query for the account balances for a specified asset and reports the free, locked, and total balances. Together, these account monitoring tools provide essential visibility into the system's financial state, enabling informed decisions and automated capital checks in future extensions.

The system retrieves detailed information about the account status for the futures and spot account. It focuses on USDT (Tether) as the margin currency and reports several key metrics, including the available balance, any funds currently locked as collateral, the level of margin usage, and the unrealized profit or loss from open positions.

Monitoring these values serves multiple purposes. Firstly, it confirms that sufficient capital is present in the account to support trading activities. Secondly, it allows the user to identify any inconsistencies—such as missing funds, unresolved orders, or issues caused by failed internal transfers or delayed synchronization with the exchange's servers. Such discrepancies, if left unnoticed, can lead to failed trades or incorrect portfolio valuations.

The system can also monitor the profit and loss associated with currently open futures positions. This involves accessing a list of all positions that have non-zero exposure. For each of these positions the size of the position, the entry price at which it was initiated, the current market price, and the resulting unrealized and realized PnL are examined. Fetching on the non-zero exposure enables the user to confirm what assets are present in the spot and futures account. They are also able to verify how capital is distributed between available and locked funds. If the request fails, the script logs the corresponding error code and message, providing immediate insight into what went wrong and where corrective action may be needed.

This allows to assess trading performance and identifying any trades that may need to be closed or adjusted. Furthermore, if this portion of the script returns no data—or if the values are inconsistent with known activity—this could indicate a connectivity issue, an internal bug, or a problem with the communication between the script and the exchange.

In addition to wallet balances, the system also fetches detailed metadata such as the account's permission status, fee structures, and information about whether the account is linked to margin or futures products.

To ensure the script executes its diagnostic function seamlessly, it includes a mechanism that automatically triggers the main wallet-checking function when the script is executed directly. This eliminates the need for a manual call and allows for convenient integration into automated monitoring pipelines.

3. Risk Management

Effective risk management is paramount in any trading strategy, and it is especially critical in a funding rate arbitrage system, where positions are often delta-neutral but still exposed to multiple trading pairs, operational, and systemic risks. Although the core principle of the funding rate arbitrage strategy is to capture low-risk yield from funding payments, prudent safeguards must be in place to ensure capital preservation, avoid margin call, and handle edge cases.

The central idea of funding rate arbitrage is to maintain a hedged position. This delta-neutral construction protects against directional price risk under normal conditions. However, perfect neutrality is difficult to achieve due to factors such as position sizing mismatches, tick size constraints, slippage, and funding cycle timing. To mitigate residual exposure, the system enforces strict position sizing based on available wallet balances and notional value equivalency. Additionally, the system checks margin levels before entry, ensuring there is adequate collateral to support the hedge.

Furthermore, proper position sizing is critical in managing leverage and exposure. The trading system implements a sizing engine that adjusts the trade size to remain within predefined capital allocation thresholds (20% of account balance), accounting for both spot and futures liquidity and balance. To prevent overexposure, a limit is enforced on the maximum allowable position size per pair. The account balances, volatility, and the funding rate edge used are to set this limit.

The system uses real-time funding rate data and calculates the expected yield adjusted for time until the next funding interval. This expected edge must exceed a minimum threshold before a trade is considered. However, funding rates are not static—they can fluctuate rapidly in response to market conditions. To mitigate entry risk, the system includes IOC condition. This ensures that only positions with favourable risk-reward profiles are executed.

Execution slippage can erode arbitrage profits or create imbalances in the hedge. To counter this, the system uses tick and step size rounding to ensure orders conform to Binance's market rule. It also includes checks to validate that both sides of the trade (spot and futures) can be executed near simultaneously and at prices within acceptable deviation from current market quotes. If one leg fails, the system avoids execution entirely to prevent un-hedged exposure.

As the strategy relies on Binance APIs and real-time order execution, any interruption in connectivity, API rate-limiting, or exchange errors pose operational risk. To address this, the bot includes exception handling routines and logs trade attempts through the Trade Ledger subsystem. Regular polling and logging of account status is conducted to ensure that wallet balances and positions remain in sync and that margin levels are maintained. This is to ensure that we are not margin called. Failures or inconsistencies are logged and can trigger emergency halts or alerts. Additionally, a secondary endpoint is used to manually query spot account positions using custom HMAC-authenticated REST calls, adding redundancy to the default Binance SDK methods. This dual-approach monitoring enhances reliability and ensures robust situational awareness.

Even though the strategy avoids directional exposure, the short leg on the perpetual futures contract can face liquidation risk if left unmanaged, especially during periods of rapid price movement. The system monitors available margin and unrealized PnL. It enforces automatic stop-loss exits or position reductions if margin levels fall below safe thresholds. In addition, the system does not use cross-margin by default, preferring isolated margin for containment of risk within individual positions.

Moreover, all trades are recorded which facilitates not only post-trade analysis but also aids in identifying anomalies, such as frequent premature exits, funding edge misestimates, or consistent underperformance by asset class. The summary function of the ledger offers a high-level view of ongoing risk and execution quality.

Furthermore in high-frequency market environments, especially during periods of extreme volatility, a rapid sequence of signals can lead to overtrading or multiple redundant entries within a short time frame. While this is less common in funding rate arbitrage, where valid opportunities are naturally sparse, speed bumps can be useful in throttling execution during periods of rapid market movement. Thus, the system implements a minimum time interval (time-lapse) between trades to prevent crowding into trades that might no longer offer the required edge by the time orders are executed. This feature acts as a self-regulating mechanism to dampen overreaction to microstructural noise and reduce the likelihood of slippage-driven losses during fast-moving markets.

While the current system provides a solid foundation for basic risk control, several enhancements could significantly improve its robustness and resilience under dynamic market conditions. Introducing automated exit rules for stale or underperforming positions would help prevent capital from being tied up in unproductive trades and reduce exposure to adverse market movements. Implementing circuit breakers to halt trading during periods of extreme funding rate volatility would serve as a safeguard against sudden, systemic shocks. Finally, integrating dynamic risk scoring models would allow the system to adapt its behaviour based on evolving market conditions, enabling more intelligent capital allocation and position sizing.

Risk management in a funding rate arbitrage strategy extends beyond mere market hedging. It encompasses a comprehensive framework that includes position sizing, execution control, funding edge evaluation, monitoring of system health, and real-time account integrity. The trading system presented here implements these principles in a disciplined manner, striking a balance between automation and safety. With continuous improvement, this foundation can be scaled into a robust institutional-grade arbitrage system capable of operating across multiple assets and exchanges with confidence.

4. Discussion

4.1. Intended future extensions

Due to time constraints and limited resources, we were unable to implement all planned features in the current system. This section outlines the additional components that were considered during development and are intended for future implementation.

4.1.1. System enhancements for robustness

To ensure that the trading system remains resilient, and adaptive in live deployment environments, we aim to scale the system further to include several additional modules and enhancements. These components aim to refine trade selection, risk control, capital efficiency, and real-time monitoring.

1. Dynamic Leverage Adjustment
   Incorporating a dynamic leveraging module would allow the system to adjust its leverage ratios based on real-time market volatility, available margin, and portfolio exposure. Rather than relying on static leverage values, the system could scale position sizes up or down based on current risk appetite and opportunity strength. This would improve capital allocation efficiency while preserving downside protection.
2. Execution Risk Estimation
   To avoid hidden trading costs, the system should implement logic to quantify execution risks such as adverse selection (being picked off by more informed traders), short borrow costs (especially in altcoins or during high demand), market impact (due to aggressive order sizes), speed and latency sensitivity, trading fees, and expected slippage based on real-time liquidity. Incorporating these factors into the decision-making process will improve the system's ability to identify truly profitable trades.
3. Live Performance Monitoring
   A performance analytics module should be developed to calculate and display live metrics such as Sharpe ratio, hit rate (percentage of profitable trades), win/loss ratio, and margin utilization. These metrics allow for real-time strategy evaluation, enabling the system to dynamically assess its effectiveness and adapt operational parameters as needed.

4. Historical funding rate data

   The system should also be able to query for historical funding rate data for backtesting, parameter stress testing and for additional trade signal generation. As mentioned in Section 2.1.3, the signal generation logic can be extended to incorporate historical funding rate data for more nuanced trade signal generation. Specifically, when the funding rate is negative (i.e., shorts pay longs), the system should not discard the opportunity outright. Instead, it should compare the current funding rate to the worst-case historical funding rate observed over a defined time window. If the current rate is still relatively favourable, the system can proceed with a long spot / short futures position under the same arbitrage logic. This extension allows the strategy to remain responsive and opportunistic under a broader range of market conditions, rather than relying solely on a simplistic positive funding rate filter.
5. Comprehensive Performance Evaluation

   Beyond real-time metrics, overall strategy performance should be evaluated holistically using cumulative profit/loss, Sharpe ratio, drawdowns, success rate, and capital and margin utilization.

4.1.2. Risk management

The system currently has put in place multiple risk management mechanisms to ensure viable trades are put through and the system works accordingly. However, we want to further expand this to allow for further dynamic error handling and also limit the amount of human monitoring required.

4.1.2.1. Automated kill switch

In this system checks for losses, API error, rate limitations have already been put in place. Furthermore, we have allowed for a manual kill switch that allows user to terminate the program if any errors occur.

However, this requires continuous active and live monitoring. Thus we want to implement an automatic kill switch that won't require live monitoring. So the system should stop trading if it encounters any of the following conditions:

- Exceedingly large orders that exceed our position limits.
- API errors or authentication failures: Repeated failures to connect to exchange endpoints (REST or WebSocket), invalid API credentials, or authentication rejections.

- Position mismatchIf a discrepancy is detected between internal position tracking and actual account holdings on Binance the system will exit all positions and suspend further operations.
- Rate limit violations or trading bans**:** If the system encounters exchange-enforced rate limits, banned IP flags, or circuit breakers, it will suspend all trading activity until human intervention occurs.

4.1.2.2. Alarm system

In addition to a kill switch we want to implement a system that doesn't terminate all trades immediately, but rather puts the system on hold and then alert user. This allow for more effective error handling. An alarm system should be integrated into the architecture to alert operators in real time about critical events, anomalies, or system health warnings. Key alarm triggers could include similar conditions as the kill switch but as a pre-warning. It should also notify us of anomalies such as slippage spikes, failed executions, or breaches of defined risk thresholds. Furthermore, it should also dynamically alert us if there are possibilities of margin call.

Alarms can be configured to send alerts via multiple channels such as email, Telegram, or Slack using webhook integrations. This ensures that any required human intervention can occur promptly, even in unattended deployments.

4.1.2.3. Handling errant algorithms and unanticipated behaviours

While the current system incorporates multiple fail-safes there remains a residual risk of unforeseen behaviours that could arise under edge-case scenarios. These include logic branches that may not trigger under typical conditions or rare market states that the system was never explicitly coded to handle. To mitigate this, a dedicated layer of defensive programming should be introduced. This would involve setting hard boundaries on all input parameters, introducing redundancy checks at each critical decision point, and implementing real-time trade sanity validators. These measures would serve as guardrails against silent failures or rogue executions.

4.1.2.4. Risk measure monitoring

The trading system can currently monitor risk using PnL and open positions. We want to extend it further to account for risk measures like VaR and maximum drawdown. It should be able to calculate them live and automatically, and then manage the risk accordingly.

4.1.2.5. Mitigating systemic risk amplification

Despite the strategy being market-neutral, there is a broader implication of many similar bots acting on the same arbitrage signal poses potential systemic risks. This is further amplified in illiquid or crowded pairs. If multiple systems simultaneously enter or exit trades based on the same or similar funding rate conditions, they may inadvertently cause rapid price dislocations or liquidity vacuums. This is primarily because as mentioned previously funding rate arbitrage opportunities are rare and few. To mitigate this, the system should incorporate basic crowding awareness metrics, such as sudden increases in order book imbalance or funding rate spikes that could indicate over-participation. In addition, we want to explore randomized execution windows or signal weighting based on market saturation. These measures help ensure the strategy does not amplify instability during periods of stress.

4.1.3. Backtesting and parameter stress testing

To strengthen confidence in the strategy's robustness, a comprehensive backtesting module should be developed. This framework would simulate the system's behaviour across historical spot and futures market data, including funding rate time series and fee models. This is to evaluate profitability, margin usage, and slippage impact. Additionally, parameter stress testing should be employed to evaluate how sensitive the strategy is to changes in variables such as funding rate thresholds, execution timing, and fee assumptions. It should also include stress tests for how the system performs under different market condition.

4.1.4. Live order status

There should be a system to display the live order status for effective position management. It should show the entire order lifecycle and flag any errant orders. It should in the trade ledger mark and store any orders that couldn't be executed, was rejected or had expired.

4.2. Limitations

While the system demonstrates foundational capabilities, it is important to acknowledge its current limitations, which arise from both infrastructure constraints and the inherent characteristics of the trading strategy. This section outlines these limitations in detail.

4.2.1. Limitations of the strategy

The funding rate arbitrage strategy, by its nature, comes with several risks and constraints that are difficult to eliminate entirely. These limitations are intrinsic to the arbitrage mechanism itself and can impact the reliability and profitability of the system. The following points highlight key challenges and structural weaknesses associated with the strategy.

4.2.1.1. Exposure to margin call risk

One of the inherent limitations of funding rate arbitrage is the risk of adverse liquidation or margin calls before profitability is realized, particularly when executed across a wide range of trading pairs including lower-cap or highly speculative tokens. In scenarios where the strategy shorts a perpetual futures contract on a low-quality or highly volatile asset, a sudden and extreme downward price movement can rapidly erode available margin. If the price crashes to near-zero levels, the short position may be forcibly liquidated, resulting in significant losses despite the initial arbitrage signal being valid. This further adversely affects our profit and loss when the funding payment is not received in time to offset the loss or prevent liquidation.

A key challenge lies in identifying and filtering out such high-risk assets in advance. While trading volume and liquidity filters may help exclude some of the most illiquid instruments, they do not fully capture market sentiment or the potential for rapid revaluation. At times, speculative tokens may exhibit temporarily high volume or positive sentiment, misleading the system into treating them as viable trading candidates. Moreover, the market-neutral assumption may fail when idiosyncratic risks arise—such as smart contract exploits, hacks, or token delistings—which can disproportionately affect the price of the targeted coin and invalidate the arbitrage position. At present, the system does not include a robust mechanism for dynamically assessing these tail risks.

4.2.1.2. Scarcity of signals

We continuously stream all Binance trading pairs that have both Spot and Perpetual Futures markets at approximately 100-millisecond intervals. Under normal market conditions, the perpetual futures bid price typically remains below or, at best, equal to the spot ask price, leaving no arbitrage opportunity to exploit. However, on rare occasions, the futures bid momentarily exceeds the spot ask, creating a short-lived positive spread.

These events are transient—usually lasting no more than 0.5 seconds—and almost always occur during periods of sudden, high-volume activity, such as sharp upward price movements (pumps) or abrupt market sell-offs (rug-pulls). During these moments, liquidity on the perpetual futures order book is rapidly consumed before the spot market has time to react. As a result, arbitrage bots often capitalize on the discrepancy within milliseconds, erasing the spread almost immediately.

From live monitoring, we observed that by the time a potential signal is detected, it often vanishes within half a second, indicating that the mispricing has already been arbitraged away. After accounting for transaction fees and slippage, the opportunity is typically no longer profitable. This suggests that positive spread conditions represent anomalies that are both rare and extremely short-lived, generally arising only during periods of extreme market volatility.

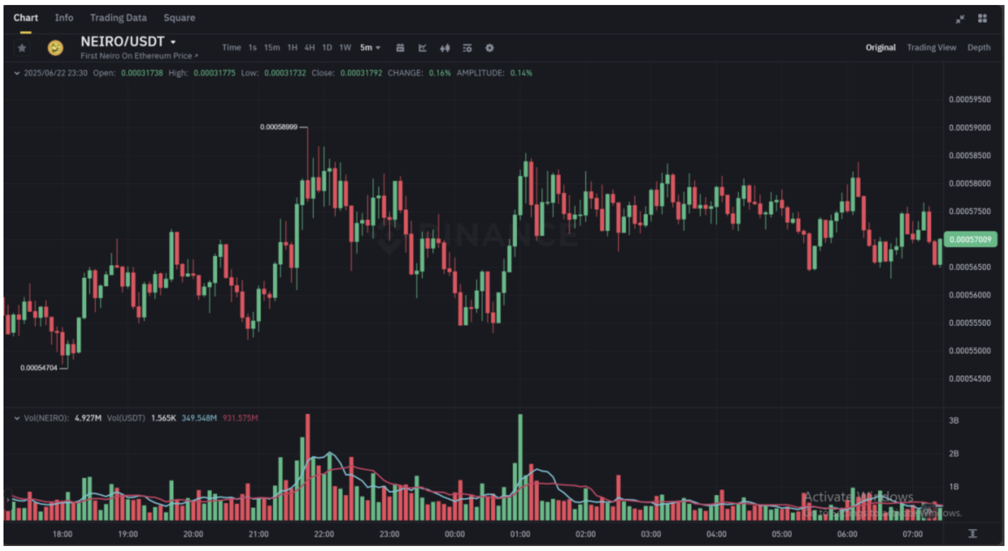


Fig 8: Sample signal for NEIRO/USDT spot from mainnet.

4.2.2. Limitations of the infrastructure

The system currently operates on Binance's testnet, which differs from the mainnet in terms of latency, liquidity, and execution behaviour, limiting real-world accuracy. Additionally, using Python imposes speed constraints.

4.2.2.1. Testnet Discrepancies vs. Live Environment

One significant limitation encountered during testing is the discrepancy between Binance's testnet and the live production environment. The testnet infrastructure differs from the live Binance platform in both behaviour and data accuracy. For example, the margin systems for spot and futures operate independently in the testnet, which does not reflect the integrated margin dynamics of the live platform. This mismatch introduces challenges when simulating real capital constraints and can result in misleading assumptions about trade feasibility or risk exposure. Additionally, testnet price feeds have, at times, exhibited unrealistic anomalies—such as futures prices being disproportionately higher than corresponding spot prices by several magnitudes (e.g., 100x)—which are not observed in the live Binance market. These discrepancies lead to requiring more robustness checks to ensure that the system performs as intended during live deployment.

4.2.2.2. Performance Constraints of Python

The system has been developed entirely in Python, for ease of prototyping and integration with Binance testnet APIs. However, Python's interpreted nature and single-threaded execution model limit its performance under latency-sensitive conditions. Given that funding rate arbitrage opportunities are often short-lived the response time of the trading system is critical. In high-frequency scenarios, execution delay caused by Python's overhead can result in missed opportunities or adverse fills. Moving forward, consideration should be given to re-implementing the system components in a compiled, high-performance language such as C++ or Rust. This transition would help reduce execution lag, improve concurrency handling, and enhance the overall robustness of the trading infrastructure.

4.3. Considerations for scaling to institutional AUM

To scale this trading system to a production-grade system, special attention must be given to the infrastructural, executional, and risk-related constraints that arise when managing larger assets under management (AUM). Operating at scale demands not just greater capital efficiency, but also highly optimized systems to ensure speed, reliability, and security in volatile markets. This section focuses on the key considerations across infrastructure, execution, and risk domains, along with recommendations for future development.

4.3.1. Infrastructure considerations

To support larger AUM and higher trade throughput, substantial upgrades to system infrastructure are necessary. One of the most critical steps is server colocation, ideally in proximity to Binance's matching engine, which is physically located in Tokyo. Hosting the trading system on servers within the same region can drastically reduce latency, ensuring that orders are transmitted and acknowledged faster than competitors operating from distant geographies. Ideally it should be placed in the same building as the server on the same connection line. A further consideration should be crowdedness of the server. We ideally want to be connected to the least crowded server to enhance speed of execution.

Additionally, a dedicated high-speed network connection such as a leased line or direct optical connection should replace conventional Wi-Fi or consumer-grade internet. Such connections provide guaranteed bandwidth, lower packet loss, and consistent latency profiles, which are essential for receiving real-time data streams and submitting orders without delay.

Lastly, the current system architecture runs on a consumer-grade machine with limited threading capacity (4 threads). To manage higher trading volume and perform concurrent tasks—such as real-time risk checks, order monitoring, and portfolio updates—multi-threaded operations and server-grade computing infrastructure will be required. This involves deploying the system on more powerful machines or distributed computing clusters with advanced resource scheduling and memory management capabilities.

4.3.2. Execution and data pipeline considerations

Scaling the system requires an enhancement of the execution logic and data ingestion pipeline. Currently, the system relies on polling for market data, however switching to a streaming data will allows us to handle the data more efficiently and at higher frequency. This would allow the system to respond instantly to any trading opportunity.

In addition, the execution strategy must become more sophisticated. For large trades, simply crossing the spread with market orders can introduce significant slippage. To mitigate this, the system should dynamically consider execution algorithms. It would be interesting to explore how based on market impact, adverse selection and slippage the system can alternate between execution algorithm such as VWAP (Volume-Weighted Average Price), TWAP (Time-Weighted Average Price), or PVOL (Participation Volume)**.**

Furthermore, a fundamental question remains: Is this strategy scalable with larger AUM? Funding rate arbitrage opportunities are typically small and time-sensitive. As the capital deployed increases, it may become more difficult to find sufficient liquidity in both the spot and perpetuals books to absorb large positions without distorting the market or eroding expected returns. Therefore, simulations and capacity testing must be conducted to understand how AUM growth affects strategy profitability.

4.3.3. Risk management considerations

Another area for enhancement is the implementation of an automated hedging module, capable of adjusting or unwinding positions dynamically in response to changes in market exposure, margin levels, or volatility. It should also be able to update its risk management strategy and parameters based on market conditions.

4.3.4. Additional Considerations

There are several other considerations when trying to deploy this on a larger AUM.

1. <u>Compliance and Reporting:</u> At scale, regulatory considerations become more prominent. The system must generate accurate trade logs, mark-to-market valuations, and capital utilization reports to satisfy audit and compliance requirements.
2. <u>Disaster Recovery and Redundancy</u>**:** A robust disaster recovery plan, including database backups, failover servers, and redundant connectivity, must be in place to ensure system continuity during outages.

4.4. AI integration

With the emergence of Artificial Intelligence (AI) as a transformative force across industries, its application particularly in algorithmic and arbitrage trading has grown increasingly promising. It will be particularly interesting to explore how AI can be integrated into our system.

As a traditional rule-based system our system often relies on hard-coded thresholds for capturing funding rate arbitrage opportunities. These rules and assumptions may fail under changing market conditions or during regime shifts. AI offers a dynamic way to shift beyond static decision-making by enabling systems to learn from data, adapt to new patterns, and optimize operations. AI can be used not only to enhance signal generation but also to support execution logic, risk management, portfolio optimization, and even the interpretation of unstructured external data.

AI could be integrated with our signal generation system to make our trade entry logic more sophisticated. Instead of relying purely on funding rate thresholds and basic price differentials, supervised learning models such as XGBoost, LightGBM, or even deep learning architectures like recurrent neural networks (RNNs) and temporal convolutional networks (TCNs) could be trained on historical trade data. These models would learn to predict the probability of a profitable arbitrage opportunity materializing within the next funding interval, incorporating multidimensional inputs such as price momentum, volatility regimes, bid-ask depth imbalance, historical funding curves, and time-to-event features. Over time, such models could adapt to changing market structures and identify latent, non-obvious signals that are difficult to capture with deterministic rules. Thus, these could be used to predict future trading opportunities.

Moreover, AI can significantly improve execution quality, targeting fleeting arbitrage opportunities. Reinforcement learning agents, for example, could be trained in simulated environments to decide how to enter and exit positions in a way that balances slippage, market impact, and latency constraints. Execution algorithms like TWAP and VWAP could be enhanced with real-time order book prediction models, enabling the system to dynamically choose between limit orders, market orders, or a hybrid strategy based on current and forecasted liquidity. For higher capital deployment, this becomes essential to avoid adverse selection and reduce hidden costs that erode arbitrage profitability. AI can also enable smart order routing if the strategy is expanded to multiple exchanges, optimizing where and how orders are placed across fragmented liquidity pools.

In terms of risk management, AI can be deployed for continuous monitoring of position exposure, market volatility, and counterparty risk in real time. Unsupervised learning methods, such as clustering and anomaly detection algorithms, can flag unusual trading patterns, latency spikes, or margin depletion events before they evolve into major failures. Furthermore, the AI layer should be able to autonomously halt trading or send alerts during outlier events. Moreover, AI-powered portfolio optimization can ensure that capital is intelligently allocated across multiple trade opportunities, instruments, or time zones.

In addition an AI system could explore sentiment analysis to search for potential risks using natural language processing (NLP). Crypto markets are heavily influenced by news flow, regulatory updates, project-specific developments, and social media sentiment. An AI system capable of ingesting Twitter feeds, Reddit threads, Discord announcements, and GitHub activity could run a sentiment analysis and provide early signals for asset-specific risk. It could alert us of potential rug pulls, exchange hacks, smart contract vulnerabilities, or delisting events. This would allow the arbitrage system to also dynamically de-risk positions in assets with deteriorating sentiment or heightened idiosyncratic exposure.

However, the AI integration has its own challenges. Models can overfit, particularly in non-stationary and adversarial environments like as crypto markets. Data quality and labelling, particularly in decentralized exchanges or altcoins with sparse trading histories, can also limit model robustness. Therefore, any AI integration must be carefully validated through rigorous backtesting, stress testing, and performance monitoring.

In summary, AI can be a beneficial extension to the existing trading system by making it smarter, faster, and more resilient. As the system matures and begins handling larger AUM and more complex market environments, the integration of AI could mark a transition from a deterministic, rule-based approach to a self-improving, data-driven engine capable of sustaining alpha in increasingly efficient markets.

5. Conclusion

Through this project we took a significant step towards building a foundational, robust trading system capable of operating in dynamic cryptocurrency markets and leveraging statistical arbitrage opportunity.

At its core, the system effectively retrieves live data from both spot and futures testnet environments, allowing for real-time trading. The current implementation relies on periodic polling to fetch market data and it has led to quite robust results. However, moving forward we would like to test transitioning to streaming data architecture as we believe it will help ensure that the system can respond instantaneously to market movements. Thus, be able to capturing fleeting arbitrage opportunities more effectively.

Leveraging the polling data feed, the system generates live trading signals. It incorporates multiple layers of error handling designed to maintain system integrity and execution accuracy. These safeguards include checks for network connectivity, verification of data stream accuracy, validation of trade order parameters (such as compliance with required tick sizes), and confirmation of successful trade execution. Together, these mechanisms reduce the risk of system failures, erroneous trades, or unexpected behaviour in volatile market conditions.

On the operational side, the system features robust account and risk management controls. It enforces capital allocation limits to prevent overexposure and includes critical risk mitigation tools such as stop loss functionality and a manual kill switch. These features ensure that positions can be exited promptly to limit losses under adverse market conditions or if manual intervention is required. Complementing this, live trade monitoring and comprehensive post-trade record keeping enable transparency and facilitate performance tracking.

Additionally, risk management extends beyond trade execution to include considerations such as margin calls and explicit trading costs, which are factored into decision-making. However, a current limitation lies in the system’s static treatment of costs. We would like to scale the system further to include more sophisticated modules that dynamically account for market impact, slippage, and other adverse cost factors.

Performance-wise, the system scans for trading opportunities per second and executes trades swiftly. This allows the system to capture high frequency, fleeting arbitrage opportunities. Nonetheless, there is considerable room for improvement in execution speed and scalability. To scale this system further to handle higher AUM or operate at a higher frequency, it is vital to improve infrastructure. This could include incorporating multithreading or parallel processing, deploying system clusters, or rewriting critical components in lower-level, high-performance languages. This would substantially reduce latency and increase throughput. Additionally, while the system presently relies on manual oversight, plans to implement automated monitoring features—such as a dynamic kill switch and an alarm system—would allow for more autonomous and resilient operation.

Looking ahead, several key features remain to be developed. A comprehensive backtesting and stress-testing frameworks should be implemented. They are crucial to evaluate strategy robustness across diverse market scenarios and historical data. Similarly, integrating detailed performance analytics—including Sharpe ratio, compound annual growth rate (CAGR), maximum drawdown, and other risk-adjusted metrics—will provide deeper insights into the strategy's effectiveness and risk profile.

Furthermore, as mentioned we have considered how the system can be scaled to manage higher AUM. However, this presents additional challenges. It requires careful considerations which include but are not limited to: multithreaded execution, capital scalability, stable and low-latency connections, potential colocation with exchanges, optimized order execution, and the integration of automated hedging mechanisms to manage portfolio risk dynamically.

Furthermore, it would be interesting and relevant to explore integration of artificial intelligence and machine learning techniques. These could further augment the system's ability to adapt to evolving market conditions and uncover new trading signals.

Despite the progress made, the funding arbitrage strategy and the testnet environment have their own inherent limitations. The scarcity of trading signals can limit profit opportunities and affect consistency. The inclusion of potentially illiquid coins in the portfolio introduces additional risk, including increased exposure to liquidation risk, slippage and margin calls. Moreover, the Binance testnet environment differs from live market conditions in terms of pricing, liquidity, latency, and infrastructure reliability, which may impact the strategy's real-world performance.

Nonetheless, we have attempted to implement and demonstrate a risk-conscious trading system. Furthermore, we have considered many possible extensions, limitations and scalability considerations. Our current system despite requiring manual intervention and monitoring is able to automatically find trading opportunities, execute trades and manage risks. Additionally, the framework is flexible enough to evolve with ongoing improvements and scaling requirements. Moreover, it is also worth noting that a lot of checks and balances we have places are because Testnet Spot and Futures are different, such is not the case on the mainnet.

Finally, we recognize that deploying such a system in live markets involves not only technical challenges but also regulatory considerations. Compliance with relevant financial regulations—such as the Markets in Financial Instruments Directive II (MiFID II) in Europe and analogous frameworks globally—must be ensured. This includes adhering to standards around risk controls, trade reporting, client protection, and operational resilience.

In conclusion, this project reflects a comprehensive and thoughtful approach to constructing a sophisticated trading system. It integrates critical components necessary for live trading, risk management, and operational control, while also acknowledging areas for future enhancement. In future, we hope to deploy this system live and implement the improvements considered here.

6. References

1. Auer, Raphael, and Stijn Claessens. "Regulating Cryptocurrencies: Assessing Market Reactions." *Papers.ssrn.com*, 1 Sept. 2018, papers.ssrn.com/sol3/papers.cfm?abstract_id=3288097.
2. Binance Academy. "Testnet." *Binance Academy*, 2025, academy.binance.com/en/glossary/testnet. Accessed 29 June 2025.
3. "Binance Spot Test Network." *Binance.vision*, 2025, testnet.binance.vision/. Accessed 29 June 2025.
4. Budish, E., Cramton, P., & Shim, J. (2015). The High-Frequency Trading Arms Race: Frequent Batch Auctions as a Market Design Response. Quarterly Journal of Economics
5. Makarov, Igor, and Antoinette Schoar. "Trading and Arbitrage in Cryptocurrency Markets." *Journal of Financial Economics*, vol. 135, no. 2, July 2019, https://doi.org/10.1016/j.jfineco.2019.07.001.
6. Seth, Shobhit. "Basics of Algorithmic Trading: Concepts and Examples." *Investopedia*, 21 Mar. 2023, www.investopedia.com/articles/active-trading/101014/basics-algorithmic-trading-concepts-and-examples.asp.
7. "What Is Funding Rate Arbitrage? | CoinGlass." *Coinglass*, 2025, www.coinglass.com/learn/what-is-funding-rate-arbitrage. Accessed 29 June 2025.