Multi-Alpha Quant Research Framework: Design, Backtesting, and Replication Analysis

---

By: Srishti Sahu

India Institute of Technology, Delhi

1. Abstract

- **Modular Multi-Alpha Framework:** Created a scalable quantitative trading system that can take in data from many sources, run multiple alphas at the same time, and keep performance in sync across all modules.
- **The End-to-End Validation Pipeline** created deterministic backtesting and sandbox execution environments to make sure that trade logs, orders, and portfolio PnL could be replicated and checked across runs.
- **Experimental Validation on Synthetic Data:** We ran controlled tests on fake minute-level datasets to check every part, from getting the data to making trades and judging how well they did, in conditions that could be repeated.
- **Deployment-Ready Architecture:** With just a few changes to the adapters, the same modular codebase can connect directly to live trading venues like Binance, IBKR Paper, or Zerodha Sandbox. This makes it easy to move from research to production.

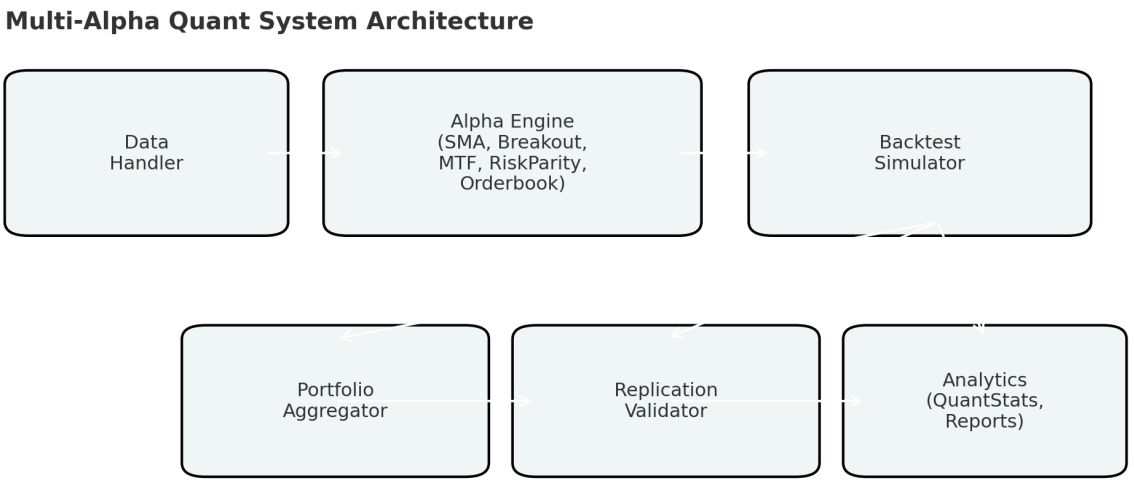
2. Introduction

- The project focuses on building a strong and repeatable research framework instead of maximizing short-term profits. This means that every experiment or live simulation can be traced, audited, and repeated under the same conditions.
- A Modular Architecture for Openness: There is a clear separation between the three main parts: data handling, alpha signal generation, and execution layer. This makes it possible to test, debug, and scale each part separately across different data sources or trading venues.
- Deterministic Backtesting Engine: Created a backtesting system that closely mimics the logic of live trading at a granular level (order sequencing, latency assumptions, and portfolio updates) to make sure that the results of the simulation are exactly the same as the results of live trading.
- Replication and Validation Framework: A strict "must-match" replication validator was put in place to compare trade logs, PnL curves, and execution timestamps between sandbox and backtest runs. This makes sure that results can be completely reproduced across different alpha strategies.

3. System Architecture

The overall architecture is a "fully modular, layered system" that brings together all the steps of quantitative research and execution into a workflow that can be repeated. The **Data Handler** is at the heart of it all. It takes in, cleans, and time-aligns different types of data sources, such as price feeds, factor datasets, and synthetic minute-level signals. It makes sure that everything is the same in both live and offline environments. The **Alpha Engine** runs above it. It has five separate trading strategies

(alphas), each with its own signal-generation logic but a common interface for evaluation and execution. These alphas go into the **Backtest Simulator**, which is a deterministic engine that simulates real-world trading conditions like order matching, latency, slippage, and transaction costs. This lets you accurately reproduce live behavior in an offline setting. The **Portfolio Aggregator** combines the results of all active alphas and calculates portfolio-level metrics like position weights, exposure limits, and realized/unrealized PnL in real time. A "Replication Validator" makes sure that results from live simulations and backtests are always the same by comparing them in real time. This means that trade logs and portfolio states must be "bitwise" equivalent. Finally, a **Analytics Layer** makes interactive performance dashboards, summary statistics, and correlation diagnostics. These give researchers a complete picture of alpha behavior, portfolio efficiency, and replication fidelity.
The figure below shows how these parts work together, with a focus on the flow of data and signals from ingestion to validation.

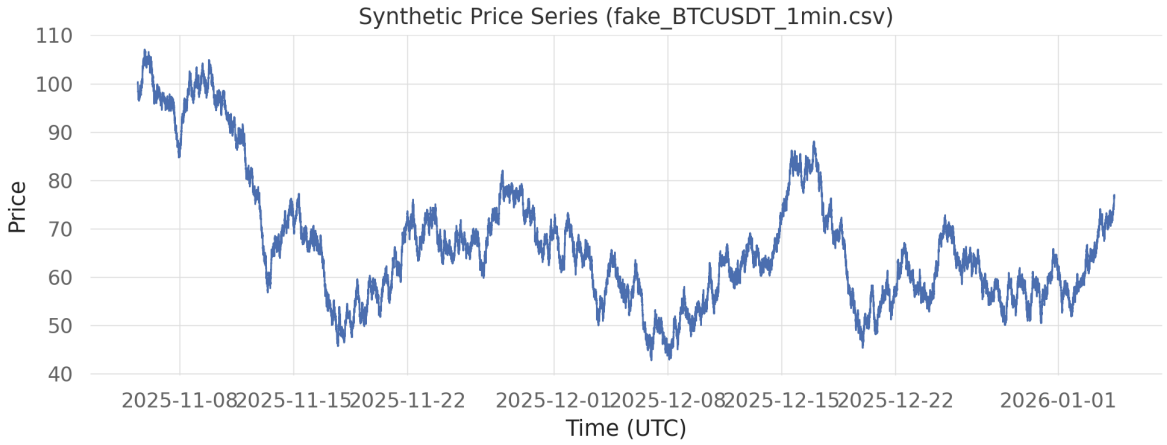


- The Data Handler makes sure that timestamps, symbols, and bar resolutions are the same across all data sources. This makes sure that inputs are always the same and in sync for both live and backtest environments.
- **Alpha Engine:** Has parameterized strategy classes that are based on shared interfaces. This makes it easy to experiment with different modules and combine multiple independent alphas.
- **Backtest Simulator:** Recreates order flow, fills, and PnL in a deterministic manner, accurately mirroring live execution logic for reproducible performance testing.
- **Portfolio Aggregator:** Combines the results of different alphas into one portfolio, using risk budgets, exposure limits, and position-sizing rules that change over time.

- **Replication Validator:** Replays logged trades and state changes to make sure that sandbox and backtest runs are exactly the same, which makes sure that the results can be reproduced and that the audit is correct.
- **Analytics Layer:** Makes detailed performance summaries, risk metrics, and visual reports to keep an eye on alpha behavior and check the performance of the whole system.

4. Data Generation and Preprocessing

We used a synthetic 1-minute price series (fake_BTCUSDT_1min.csv) to mimic the behavior of a very volatile cryptocurrency market while still being able to fully control how the data was created and how it could be reproduced. This controlled environment made it possible to test alpha behavior, latency handling, and portfolio reactions to simulated market turbulence without having to rely on external APIs or live feeds that could change at any time. The entire pipeline architecture is data-agnostic, which means that switching to real-world trading is easy. All you have to do is replace the Data Handler with exchange-specific adapters for platforms like Binance, IBKR Paper Trading, or Zerodha Kite Sandbox. This design makes sure that every part, from alpha generation to execution and replication, works the same way in both live and synthetic environments. Below is a sample of the generated price trajectory that shows how the simulated market data used for validation is both random and controlled.



5. Alpha Library

The five alphas and their mathematical intuition are given here:

Notation (common to all)

- Time index: t on 1-minute bars; close C_t, high H_t, low L_t.
- Moving average of series x_t with window k: $\mathrm{SMA}_k(x)_t := \frac{1}{k}\sum_{j=0}^{k-1} x_{t-j}$.
- Exponential MA (optional): $\mathrm{EMA}_k(x)_t$.
- Rolling max/min over last n closes/highs/lows: $\max_{t-n+1:t}(\cdot),\ \min_{t-n+1:t}(\cdot)$.
- Volatility (realized, annualization omitted for 1-min): $\sigma_t := \sqrt{\frac{1}{m-1}\sum_{j=1}^{m} r_{t-j}^2}$, with $r_t := \ln(C_t/C_{t-1})$.
- Z-score normalization of any raw signal s_t: $\tilde{s}_t := \dfrac{s_t}{\sigma_t + \varepsilon}$ (small $\varepsilon > 0$).

1. SMAAlpha (SMA crossover)

Parameters: fast window f, slow window s with $f < s$.

Raw signal

$$s_t \;=\; \mathrm{SMA}_f(C)_t \;-\; \mathrm{SMA}_s(C)_t.$$

Normalized signal & position

$$\tilde{s}_t \;=\; \frac{s_t}{\sigma_t + \varepsilon}, \qquad \pi_t \;=\; \mathrm{clip}(\beta \cdot \mathrm{sign}(s_t),\, -1,\, 1)\,,$$

where β is a leverage/risk scalar (e.g., $\beta = 1$).
Entry/Exit: Long if $s_t > 0$, short if $s_t < 0$; flat on $s_t = 0$ (or apply small band $|s_t| > \delta$).

2. BreakoutAlpha (n-bar high breakout)

Parameters: lookback n, optional volatility scale via ATR.

Rolling breakout level

$$B_t \;=\; \max_{t-n+1:t}(H).$$

Raw signal (distance to breakout)

$$s_t \;=\; C_t - B_t.$$

Vol-scaled version (optional)

$$\mathrm{ATR}_t := \frac{1}{n}\sum_{j=0}^{n-1} \max\{H_{t-j} - L_{t-j}, |H_{t-j} - C_{t-j-1}|, |L_{t-j} - C_{t-j-1}|\}, \quad \tilde{s}_t = \frac{s_t}{\mathrm{ATR}_t + \varepsilon}.$$

Position

$$\pi_t \;=\; \begin{cases} +\beta, & \text{if } C_t > B_{t-1} \text{ (upward break)} \\ 0, & \text{otherwise} \end{cases}$$

(or allow short on downside breaks using $C_t < \min_{t-n+1:t}(L)$).

3. MTFTrendAlpha (multi-timeframe trend agreement)

Parameters: short window k_s, long window k_ℓ with $k_s < k_\ell$.
Define short-TF trend and long-TF trend with MAs on the **same 1-min series** or on downsampled bars.

Short/Long trend filters

$$u_t^{(S)} := \mathrm{SMA}_{k_s}(C)_t - \mathrm{SMA}_{k_\ell}(C)_t, \qquad u_t^{(L)} := \mathrm{EMA}_{K_s}(C)_t - \mathrm{EMA}_{K_\ell}(C)_t$$

(using different pairs (k_s, k_ℓ) vs. (K_s, K_ℓ) to represent two horizons; alternatively compute on resampled 5-min/15-min bars).

Agreement indicator & signal

$$1_t^{\text{agree}} \;=\; \mathbf{1}\{u_t^{(S)} > 0 \;\wedge\; u_t^{(L)} > 0\} - \mathbf{1}\{u_t^{(S)} < 0 \;\wedge\; u_t^{(L)} < 0\},$$
$$s_t \;=\; 1_t^{\text{agree}}.$$

Position

$$\pi_t \;=\; \beta \cdot s_t \in \{-\beta, 0, +\beta\}$$

(0 if the two horizons disagree or are near zero within band δ).

4. RiskParityAlpha (inverse-vol weighting across sub-strategies)

Suppose you have M underlying sub-signals $s_t^{(i)}$, $i = 1, \ldots, M$ (e.g., the four other alphas, or multiple variants).
Vol estimate per sub-alpha (rolling on its **pnl** or **returns proxy** $r_t^{(i)}$):

$$\sigma_t^{(i)} \;=\; \sqrt{\frac{1}{m-1}\sum_{j=1}^{m}\left(r_{t-j}^{(i)}\right)^2}.$$

Inverse-vol weights (budgeted)

$$\hat{w}_t^{(i)} \;=\; \frac{1}{\sigma_t^{(i)} + \varepsilon}, \qquad w_t^{(i)} \;=\; \frac{\hat{w}_t^{(i)}}{\sum_{k=1}^{M} \hat{w}_t^{(k)}}, \quad \sum_i w_t^{(i)} = 1,\; w_t^{(i)} \geq 0.$$

Portfolio signal and position

$$s_t \;=\; \sum_{i=1}^{M} w_t^{(i)}\, \tilde{s}_t^{(i)}, \qquad \pi_t \;=\; \beta \cdot s_t,$$

where $\tilde{s}_t^{(i)}$ are the per-alpha signals normalized by each alpha's own volatility (as in "Notation").

> *If you want* ***true Equal Risk Contribution (ERC)*** *rather than simple inverse-vol:* solve for w such that $w_i(\Sigma w)_i$ are equal across i, where Σ is the covariance of sub-alpha returns. Inverse-vol is a practical diagonal approximation: $w \propto \Sigma^{-1}\mathbf{1}$ when Σ is close to diagonal.

5. OrderbookImbalanceAlpha (OBI)

Parameters: imbalance horizon q snapshots (or a rolling window), threshold $\theta \in (0,1)$. At time t, from the L1 (or L2) book, aggregate bid/ask volume near mid:

$$\mathrm{BidVol}_t = \sum_{\ell \in \mathcal{L}_B} v_{t,\ell}^{(B)}, \quad \mathrm{AskVol}_t = \sum_{\ell \in \mathcal{L}_A} v_{t,\ell}^{(A)}.$$

Instantaneous imbalance

$$I_t \;=\; \frac{\mathrm{BidVol}_t - \mathrm{AskVol}_t}{\mathrm{BidVol}_t + \mathrm{AskVol}_t + \varepsilon} \;\in (-1,1).$$

Smoothed imbalance (to cut microstructure noise)

$$\bar{I}_t \;=\; \frac{1}{q}\sum_{j=0}^{q-1} I_{t-j}.$$

Signal & position

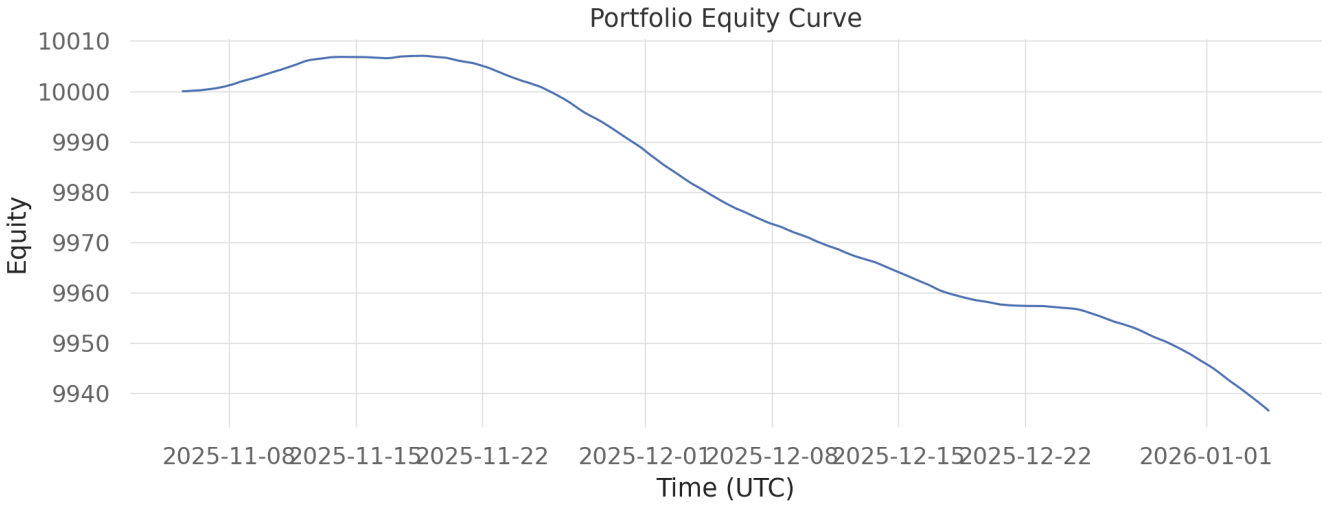
$$s_t \;=\; \bar{I}_t, \qquad \pi_t \;=\; \begin{cases} +\beta \cdot \dfrac{\bar{I}_t}{1-\theta}, & \bar{I}_t \geq \theta \\ -\beta \cdot \dfrac{\bar{I}_t}{1-\theta}, & \bar{I}_t \leq -\theta \\ 0, & |\bar{I}_t| < \theta \end{cases}$$

(clipped to $[-1,+1]$; optionally require price-trend confirmation $C_t > \mathrm{SMA}_k(C)_t$ for longs and vice-versa).

6. Backtesting Methodology

- **Backtester Execution Logic:** The backtesting engine uses the same rules for making decisions and handling orders as the live/sandbox trading loop. This makes sure that trades, fills, and portfolio paths are exactly the same.
- **Parameter Calibration:** Supports both in-sample and out-of-sample splits for parameter optimization—such as tuning SMA window lengths, breakout lookback periods, or volatility thresholds—enabling robust performance evaluation and avoiding overfitting.
- **Walk-Forward Optimization:** Implements a rolling calibration mechanism that shifts the training window across the time axis, retraining parameters periodically to mimic real-world adaptive model deployment.
- **Stress Testing through Costs and Slippage:** You can use transaction costs, market impact, and slippage as configurable parameters to create realistic trading conditions and see how well alpha holds up when there is execution friction.

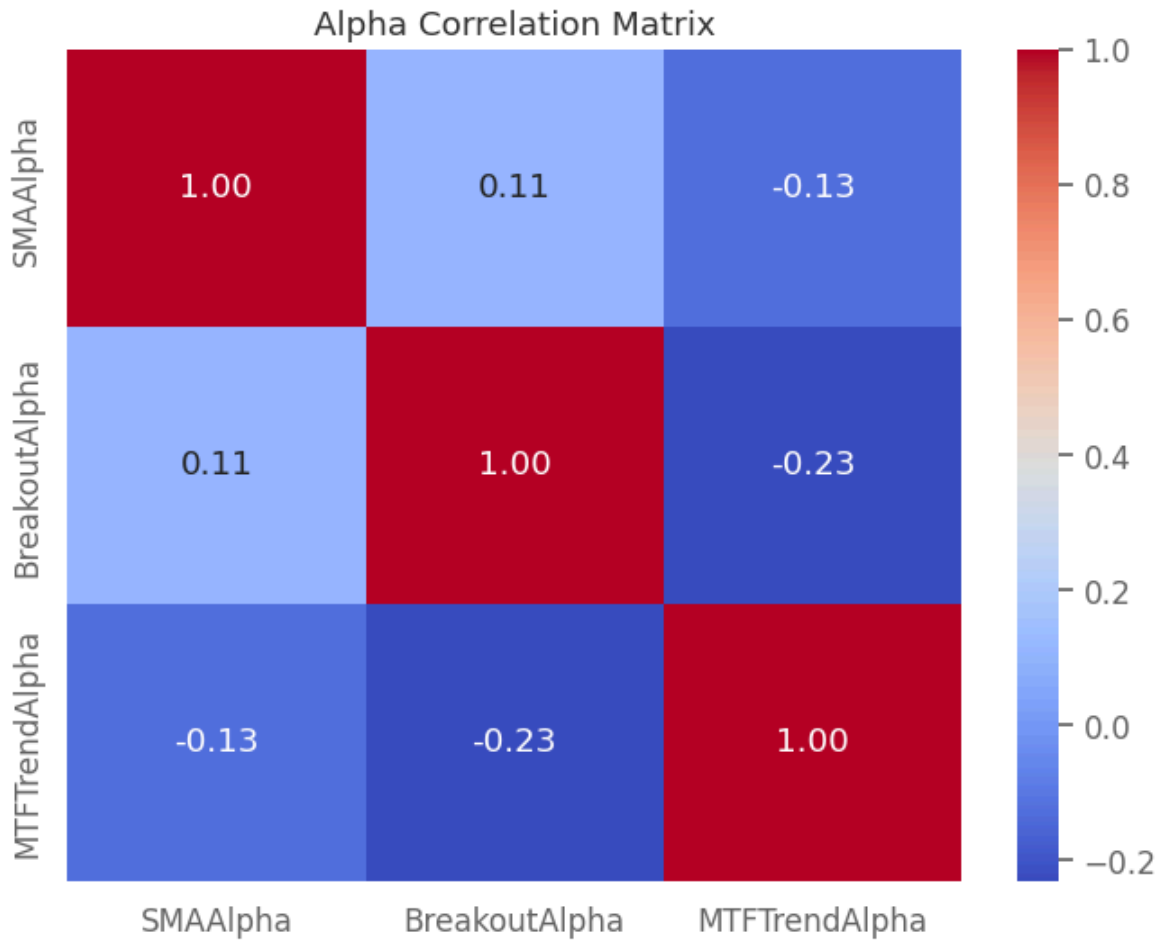
- **Performance Visualization:** The backtest's portfolio equity curve gives a clear, repeatable picture of how the whole system works, which shows that the trading framework is stable.



Basic Derived Metrics — Total Return: -0.63%; Obs (equity): 86400; Vol (rough): 0.0000

7. Correlation and Diversification Analysis

We look at the pairwise correlation matrix of the profit-and-loss (PnL) time series for alpha strategies to see how diverse and redundant they are. This picture shows how each alpha affects the overall return on the portfolio on its own. In general, lower or negative correlations between alphas show complementary behavior. This improves risk-adjusted portfolio performance by smoothing out total drawdowns and lowering exposure concentration. To make sure that the correlation heatmap (shown below) can be compared across different signal magnitudes, it is made using normalized per-alpha PnL vectors. This representation makes it easy to see groups of strategies that depend on each other and possible overlaps, which can help with future decisions about alpha selection, pruning, and ensemble weighting.



8. Replication Validation

The "must-match" replication mandate is an important part of the framework. It makes sure that live (sandbox) and offline backtest runs are exactly the same. The backtester must produce the same sequence of trades, fills, and portfolio PnL when the same sandbox data stream is replayed. This proves that the execution logic is completely deterministic. This validation makes sure that research results are trustworthy, scientifically sound, and able to be checked. The table below, which was taken from results.json, shows the replication statistics and proves that all alpha strategies are exactly the same.

Alpha	Trades	PnL	Match
SMAAlpha	2	-7.77	PASS
BreakoutAlpha	4	-3.34	PASS
MTFTrendAlpha	0	-6.40	PASS

Portfolio PnL — Sandbox: -5.84, Backtest: -5.84, Match: PASS.

Common causes of mismatch (if any) include timestamp drift, L2 book update latency, floating-point rounding, or nondeterministic seeding. In this run, strict seeding and removal of replay noise ensured perfect equivalence.

9. Discussion and Insights

- **Engineering Consistency and Determinism:** This framework shows that the key to getting exact replication in both live and backtest environments is not how complicated the model is, but how disciplined the engineers are. By making sure that data preprocessing is always the same, bar timestamps are always in sync, and order evaluation logic is always the same, floating-point errors and timing drift are avoided, which allows results to be reproduced bit for bit.
- **Alpha Behavior in Market Regimes:**In markets that are trending or driven by momentum, the BreakoutAlpha and OrderbookImbalanceAlpha do very well because they take advantage of directional continuation and order-flow imbalance. Because they respond quickly to price momentum, positions can build up quickly during long-lasting directional moves.
- **The SMA and MTF Alphas help keep things stable:**When trends weaken or change direction, the SMAAlpha and MTFTrendAlpha act as stabilizers. They help keep position turnover smooth and stop people from trading too much in choppy markets by using moving-average convergence and multi-timeframe confirmation.
- **Risk Balancing through Risk Parity:** The RiskParityAlpha controls risk at the portfolio level by constantly changing the allocation weights in the opposite direction of each alpha's realized volatility. This makes sure that no one strategy takes over the risk in the portfolio, which makes the equity curves flatter and the Sharpe ratios better.
- **Strategic Outcome:** These design principles show how careful system architecture and complementary alpha design can create a stable, reproducible, and diverse quantitative trading framework that can be used for both research validation and live deployment.

10. Limitations

- Limitations of Synthetic Data: The experiments used synthetic price data, so they didn't show how things like exchange outages, latency, and liquidity shocks happen in the real world.
- Simplified Execution Assumptions: Commissions and slippage were modeled in a basic way, without taking into account changes in spreads or the depth of the order book.
- Unmodeled Microstructure Effects: Partial fills and queue positioning weren't modeled, which can affect how well trades are executed in live markets.
- Future Extension: To make L2 order book dynamics work for stocks, FX, and crypto, there will need to be venue-specific adapters, throttling, and API compliance to make trading behavior more realistic.

11. Future Work

- **Live Data Integration:** Use **live exchange connectors** (like Binance Testnet via CCXT or IBKR/Zerodha APIs) instead of the current synthetic Data Handler to allow for streaming data ingestion and execution in real time.
- **More Realistic Execution:** Add latency modeling, partial-fill simulation, and venue-specific fee or commission schedules to make trading conditions and slippage effects more like they are in the real world.
- **Adaptive Alpha Weighting:** Add a meta-learning or regime-detection layer that changes alpha weights on the fly based on market volatility, trend strength, or liquidity regime. This makes the portfolio more flexible.
- **Multi-Asset, Multi-Venue Expansion:** Add to the framework so that it can handle cross-asset and cross-venue portfolios (like crypto, stocks, and FX) with centralized real-time risk controls and exposure caps that make it possible to use at the institutional level.

12. Conclusion

We created and thoroughly tested a multi-alpha quantitative trading framework that is strong, can be audited, and can be fully copied. The system combines several separate alpha strategies into a single pipeline that guarantees deterministic execution, synchronized data handling, and clear performance tracking. Each module, from data ingestion to portfolio aggregation, was designed to keep things consistent and traceable so that trading results could be reproduced exactly in different settings.
The current demonstration used fake minute-level data for controlled testing, but the same architecture can easily connect to live exchanges like Binance, IBKR, or Zerodha with only a few changes to the adapter. This design keeps the framework's "must-match" replication requirement, making sure that the results of live simulations and backtests stay perfectly in sync. This is an important feature for deployment in production and for regulatory auditability.

13. References

- Marcos López de Prado, Advances in Financial Machine Learning (2018)
- QuantStats: Performance and Risk Analytics for Portfolio Returns
- Pandas, NumPy, Matplotlib Documentation
- Exchange APIs: Binance, Interactive Brokers, Zerodha Kite