Cointegration-Based Algorithmic Trading Pipeline

Thies Weel

April 16, 2025

Contents

1 Introduction

During my academic career, I have encountered a variety of mathematical and computational tools used in quantitative finance. These include, among others, machine learning, cointegration, volatility modeling, and portfolio optimization. I found these topics particularly intriguing due to their mathematical elegance; however, they were often explored in isolation. What I found missing was a broader application that brings all of these methods together into a single cohesive project. With this initiative, I aim to explore the challenges and insights that arise when combining these tools into one comprehensive pipeline.

When it comes to investing in the stock market with the expectation of long-term growth, I have observed that the average investor (myself included at this stage) is often guided by gut feeling. This emotional approach is especially prone to biases and can lead to inconsistent decisions, particularly for nonprofessionals. With this project, I aim to adopt a more quantitative and systematic approach to investing. Although I do not expect this to lead directly to live investment opportunities, I am eager to explore the practical implications and skill development that such a quantitative framework can offer.

One of the primary limitations I anticipate is the lack of direct access to real-time market data and the inability to send live buy/sell orders to an exchange. These constraints may limit the real-world application of the system, but do not detract from its educational and exploratory value.

The core of this pipeline will be based on cointegration, a concept that I find particularly elegant due to its combination of simplicity and mathematical robustness. Cointegration provides a solid mathematical foundation upon which further tools, such as error correction models, Monte Carlo simulations, and neural networks, can be layered.

The pipeline will begin by analyzing stock market data across different industries, identifying highly correlated pairs, and then testing these pairs for cointegration. When a strong cointegration relationship is found, the system will apply an error correction model to identify deviations. These deviations will be used for forecasting, supported by Monte Carlo simulation to generate expectations. A neural network will then integrate these forecasts with additional market features, such as open interest, to determine position sizing based on risk. In the final stages, this framework will be extended to the options market, where ECM-driven forecasts combined with option chain data will help identify low-risk, high-reward strategies.

This project is not only a technical exercise but also a personal challenge to unify multiple fields I've studied—mathematics, econometrics, computer science, and computational engineering—into something coherent, scalable, and potentially powerful. It is a self-initiated project pursued independently alongside my master's studies and is not affiliated with any university program.

2 System Overview

The system is designed as a modular pipeline that automates the end-to-end process of identifying cointegrated asset pairs, generating forecasts, and evaluating investment strategies. Each module serves a distinct function and contributes to a comprehensive view of statistical arbitrage opportunities in equity and derivative markets.

At a high level, the pipeline follows these stages:

2.1 Data Acquisition Strategy

The pipeline begins by loading historical stock price data for a broad set of tickers across multiple industries and sectors. However, to manage computational complexity and increase the likelihood of identifying meaningful relationships, the search for cointegrated pairs is initially limited to assets within the same industry. This industry-focused approach ensures that candidate pairs share similar structural and economic characteristics, increasing the probability of long-term equilibrium behavior. Data is initially retrieved with daily resolution via free APIs such as `yfinance`, which provides adjusted close prices, trading volume, and basic option chain data. Later in the pipeline, higher-resolution data (e.g., hourly) will be incorporated to refine the risk profile of trading opportunities. While daily data is used for detecting and modeling long-term cointegration relationships, intraday data will help assess short-term volatility, optimize entry timing, and inform position sizing decisions with greater precision.

To maintain data quality, the system performs rolling updates of the most recent 7 days of data. This ensures that late corrections—such as those caused by dividend adjustments, stock splits, or volume updates—are captured. The update cycle is executed at regular intervals during market hours (with optimal timing to be determined during the backtesting phase), overwriting the most recent local records.

Supplementary data—such as option chain information (including strike prices, bid/ask spreads, open interest, and volume) and historical volatility derived from price series—is gathered where available. While more advanced liquidity indicators may be incorporated in the future, the current implementation primarily relies on volume-based metrics, which are sufficient given the low-volume nature of the trades considered in this pipeline.

All data is stored in efficient local formats (e.g., Parquet or HDF5), ensuring fast read/write access for model training, simulation, and analysis.

2.1.1 Testing on Comprehensive Historical Datasets:

To evaluate the full potential of incorporating derivative market data (e.g., open interest, implied volatility surfaces, and Greeks), the system is also tested on static, comprehensive historical datasets that include these variables. These datasets — while not up-to-date — provide a valuable sandbox for quantifying the additional predictive power such data can offer. For instance, the use of open interest in position sizing logic or implied volatility in option strategy selection can be assessed through such a dataset. This stage helps guide the eventual integration of more advanced real-time data sources (i.e., paid APIs) and provides an empirical basis for prioritizing which variables to incorporate into the live pipeline.

2.2 Correlation and Pair Selection

The system uses a two-stage filtering process to select candidate stock pairs for cointegration analysis.

A correlation matrix is first computed within each industry group, rather than across the full ticker set. This industry-specific filtering reduces computational load and increases the likelihood of finding economically meaningful relationships. Pairs exceeding a correlation threshold based on rolling-window Pearson correlation are selected as candidates.

To further optimize resource usage, a Bayesian inference framework is then applied to estimate the probability of cointegration for each candidate pair. The model incorporates prior information, such as historical test outcomes and previous correlation strengths, and updates posterior probabilities using new data on a weekly basis. This update is performed over the weekend, when markets are closed, to minimize interference with live computations.

Only pairs with a posterior probability exceeding a given threshold are retained in a high-confidence set. During the trading week, downstream computations, such as, coientegration tests and Z-score monitoring, are restricted to this subset. This results in a more efficient, and thus more scalable, pipeline.

2.2.1 Deployment Considerations

The system is designed for remote deployment on an SSH-accessible server, allowing for continuous operation and access to greater computational resources. The architecture is modular, enabling different components of the pipeline to be run on separate schedules (e.g., price updates every few hours, Bayesian updates weekly, neural network retraining monthly?).

An optional exploratory extension involves integrating an FPGA board to offload highly parallelizable tasks such as Monte Carlo simulation. While not expected to be necessary in the early stages of development, this serves as a learning experiment with potential future benefits if the scope of analysis scales up significantly. The implementation of this component will be considered once the pipeline has matured and a sufficiently large set of cointegration candidates has been collected and evaluated.

3 Cointegration Analysis

The detection of cointegration relationships between stock pairs forms a core component of the pipeline. Cointegration allows for the identification of statistically stable long-term relationships, even between non-stationary time series, and forms the foundation of this pipeline.

3.1 Engle-Granger Two-Step Approach

For this project, the pipeline focuses exclusively on pairwise cointegration analysis using the classical Engle-Granger two-step method. Although the Johansen test is a more general method capable of identifying multiple cointegrating relationships across larger groups of assets, it is not employed here due to the project's focus on asset pairs rather than multi-asset groups. This design choice reflects both practical considerations — such as computational efficiency and execution simplicity — and the relatively rare occurrence of robust cointegration across larger groups of assets.

The Engle-Granger method involves:

1. Performing an ordinary least squares (OLS) regression between two asset price series.

2. Applying an Augmented Dickey-Fuller (ADF) test on the residuals to check for stationarity.

Only pairs with high historical correlation are considered for Engle-Granger testing to reduce computational overhead and improve the likelihood of detecting meaningful relationships.

3.1.1 Note on ETF and Constituent Group Analysis:

One promising real-world application of the Johansen test is the analysis of cointegration relationships between ETFs and their underlying constituents. Since ETFs are constructed from weighted combinations of individual stocks, they are structurally linked to their components, making them strong candidates for multivariate cointegration modeling. However, while theoretically compelling, this strategy poses significant practical challenges: it requires running the pipeline over groups of assets, which will be computationally heavy, maintaining up-to-date weightings to adjust the cointegration vector dynamically over time, and executing multi-leg trades. These factors introduce a lot of execution complexity and therefore will not be part of the scope of this project.

3.2 Multi-Timeframe Cointegration Analysis

To enhance the robustness of cointegration detection, the pipeline evaluates candidate pairs across multiple historical windows:

- **Short-term:** 60 trading days
- **Medium-term:** 120 trading days
- **Long-term:** 250 trading days

This multi-timeframe approach captures both transient and persistent relationships, aiding in the detection of structural breaks or regime shifts. The choice of these window lengths is informed by practices in financial econometrics and empirical research that utilize rolling windows to assess the stability and persistence of cointegration and beta estimates over time [**?**].

Recognizing that the selection of window size can significantly affect the sensitivity and reliability of cointegration tests, the pipeline treats window length as a tunable degree of freedom. By allowing experimentation with different window sizes.

3.3 Bayesian Confidence Framework

To combine results from multiple tests and timeframes, the pipeline uses again a Bayesian framework. Each pair receives a prior probability of cointegration based on factors like historical frequency, sector similarity, and correlation. This is updated with new test results (e.g., ADF statistics) to produce a posterior probability.

The resulting score serves as a long-term confidence metric, helping prioritize pairs during live signal processing and ECM forecasting. It also supports adaptive learning and provides a useful signal for the neural network's risk assessment.

3.4 Error Correction Modeling

Once a cointegration relationship is established between two assets, the pipeline fits an Error Correction Model (ECM) to capture short-term deviations from their long-term equilibrium. The ECM allows the system to model the dynamics of mean reversion and generate signals that can be interpreted by subsequent components of the pipeline.

3.4.1 ECM Formulation

The basic ECM for two cointegrated time series y_t and x_t is defined as:

$$\Delta y_t = \alpha(y_{t-1} - \beta x_{t-1}) + \varepsilon_t$$

where:

- $y_{t-1} - \beta x_{t-1}$ is the cointegration residual,
- α is the speed-of-adjustment coefficient,
- ε_t is a white noise error term.

The coefficient α should be negative and statistically significant to confirm mean-reversion behavior.

3.4.2 Z-Score Triggers as a Degree of Freedom

The cointegration residuals are standardized into Z-scores as follows:

$$z_t = \frac{r_t - \mu}{\sigma}$$

where r_t is the residual at time t, and μ and σ are the rolling mean and standard deviation of the residuals. Rather than using fixed entry and exit thresholds (such as ± 2 and ± 0.5), the Z-score is treated as a tunable degree of freedom. Thresholds like ± 1.5, ± 1, or even lower may be experimented with to evaluate predictive value when combined with downstream models.

3.4.3 Signal Generation and Integration

Z-score signals are not used as final trade triggers but are instead passed to a forecasting module and neural network, which are tasked with interpreting their relevance in context. This allows the system to defer judgment on marginal signals and lets the neural network incorporate additional market features such as open interest, volatility, and recent trend behavior to evaluate the signal's risk and expected return. With a sufficiently large training set, this structure allows the neural network to learn nonlinear decision boundaries around signal effectiveness, improving the robustness of the overall strategy.

4 Forecasting Framework

Once short-term deviations from equilibrium are modeled using the ECM, the pipeline leverages Monte Carlo simulation to generate a probabilistic forecast of the spread's future behavior. This forecasting framework aims to quantify the likelihood and timing of convergence back to the long-term equilibrium, which is essential for both trade selection and risk management.

4.1 Monte Carlo Simulation on ECM Output

The ECM equation serves as the generative process for simulating the future trajectory of the spread. Using the estimated adjustment speed (α), the most recent cointegration residuals, and a stochastic error term ε_t, the system produces multiple simulated paths of spread evolution. These simulations reflect a range of potential outcomes driven by the modeled behavior of ε_t, which can be drawn using different techniques.

4.2 Residual Simulation Options

Instead of relying solely on the standard assumption of Gaussian white noise, I will explore alternative methods that better capture empirical characteristics of financial time series—such as fat tails, asymmetry, and volatility clustering.

Empirical Bootstrapping: Instead of relying solely on the standard Gaussian white noise assumption, I will explore alternative methods that better capture the empirical characteristics of financial time series, such as fat tails, asymmetry, and volatility clustering.

Student-t Noise: An alternative parametric option is to model residuals using a Student-t distribution:

$$\varepsilon_t \sim t_\nu(0, \sigma^2)$$

This approach captures the fat-tailed nature often seen in financial time series, providing a balance between simplicity and realism. The degrees of freedom parameter ν can be tuned to control the severity of tail events in simulation.

Both methods are supported in the forecasting module, and the pipeline is designed to allow experimentation with different noise models to assess their impact on convergence probabilities, confidence intervals, and downstream position sizing.

4.3 Probabilistic Insight for Convergence

By analyzing the distribution of simulated paths, the system estimates the probability that the spread will revert to its mean within a given time horizon (e.g., 5 or 10 trading days). This probabilistic insight enables the identification of high-conviction setups where mean reversion is not only expected, but likely within a tradeable timeframe.

These probabilities can be used to:

- Filter out weak or slow-converging signals,
- Adjust the aggressiveness of trade sizing,
- Prioritize entry opportunities across multiple candidate pairs.

4.4 Forecast Confidence Intervals

From the ensemble of Monte Carlo simulations, the pipeline constructs forecast confidence intervals for each time step. These intervals quantify the expected dispersion of the spread and help visualize the uncertainty around the central forecast. If the confidence band remains tightly clustered around the mean, the signal is likely to be stable; if it widens, it suggests a volatile or unreliable setup.

These forecast intervals are also valuable inputs to downstream modules such as the neural network, which can incorporate them to assess signal reliability and expected risk-adjusted return.

I am excited to test and backtest these Monte Carlo simulations—both on real-market data and controlled synthetic datasets such as circular data—to evaluate the forecasting potential of the approach and determine which residual simulation methods produce the most reliable results.

5 Neural Network-Based Position Sizing and Risk Management

The pipeline incorporates two distinct neural network models to support different stages of the trading lifecycle: one for trade entry and position sizing, and another for ongoing risk assessment of open positions. This modular approach simplifies training logic, and allows each model to specialize in its respective task.

5.1 Entry Decision and Position Sizing

For trade entry, a feedforward neural network (FNN) is used. This model takes a snapshot of signal features at the time of trade setup and maps them to a position sizing decision. Input features may include:

- Current Z-score of the spread (can be computed locally from price data)
- Historical volatility (rolling standard deviation computed locally)
- Confidence intervals and convergence probability from Monte Carlo simulations (based on your ECM residual model)

The FNN outputs either a continuous position size or a trade recommendation (e.g., enter/skip). As a static input model, the FNN is efficient, interpretable, and well-suited for real-time execution decisions.

5.2 Monitoring Open Positions

To assess ongoing risk after a position is opened, a second model based on Long Short-Term Memory (LSTM) networks is employed. This model continuously processes updated market data and model outputs to evaluate whether the original trade remains valid or if risk has increased. Inputs may include:

- Time series of Z-scores since entry
- Rolling ECM forecast updates
- Changes in volatility or open interest
- Cumulative unrealized PnL

The LSTM captures temporal patterns in spread behavior, helping the system decide whether to hold, reduce, or exit a position. By training this model on sequences of past trade outcomes, it can learn complex risk trajectories that go beyond simple threshold logic.

5.3 Architecture Summary

- **NN1 – Entry Model:** Feedforward Neural Network (FNN)
- **NN2 – Monitoring Model:** Recurrent Neural Network (LSTM or GRU)

This separation of models supports clearer development, targeted training, and a more robust trading pipeline that adapts both at entry and throughout the life of a trade.

5.4 Capital Allocation Considerations

While the neural network modules provide position sizing signals on a per-trade basis, the broader capital allocation strategy across multiple simultaneous trades has not yet been finalized. The appropriate method—whether score-based allocation, risk-adjusted scaling, or more sophisticated portfolio optimization—will be determined once sufficient backtesting data becomes available. This allows the system to adapt its allocation logic based on observed signal frequency and reliability, ensuring that capital committed to one position does not unnecessarily limit participation in future, potentially better, opportunities.

6 Options Layer

The options layer of the pipeline translates equity-based signals—primarily generated from ECM forecasts and Monte Carlo convergence probabilities—into structured option strategies. The goal is to execute trades that align with forecasted direction, time horizon, and risk profile, while managing exposure to volatility and time decay through basic option structures.

6.1 Signal-to-Option Mapping

Signals identified by the pipeline are mapped to directional or volatility-based option trades depending on the nature and strength of the forecast. A strong convergence signal may translate into a directional trade such as a long call or put, or alternatively a vertical spread to limit downside risk. The expected convergence horizon, derived from Monte Carlo simulation, is used to select an appropriate expiry date for the options contract.

6.2 Strategy Selection

In the initial implementation, the following strategy types are supported:

- Long calls and puts for basic directional exposure.
- Vertical spreads (bull call or bear put) for risk-controlled directional bets, chosen for their simplicity and compatibility with limited data.

More advanced strategies such as straddles, calendar spreads, or delta-neutral structures may be implemented in later stages once more complete data (e.g., Greeks and implied volatility surfaces) becomes available. These strategies require a more detailed understanding of options sensitivity to time and volatility and often involve multi-leg setups that depend on accurate modeling of theta, vega, and IV behavior—data not typically available through free APIs.

6.3 Option Filtering and Selection

In the current implementation, option selection is based on a limited set of reliably available attributes:

- Open interest and volume thresholds to ensure sufficient liquidity.
- Strike proximity to the underlying price (e.g., ATM or slightly OTM).
- Expiry dates aligned with the expected mean-reversion window.

Without access to detailed Greeks or IV surfaces, the filtering logic emphasizes simplicity and tradeability. Contracts are ranked by liquidity and alignment with forecast parameters. If no suitable contract is found, the system defaults to the best available alternative or reverts to trading the underlying equity pair.

6.4 Position Sizing and Capital Allocation

Position sizing is determined by the output of the neural network (FNN), which returns a scalar score representing trade confidence and risk-adjusted sizing. This score is scaled by a predefined capital allocation per trade. To manage overall exposure and maintain flexibility for future signals, the system enforces a cap on the number of simultaneously active trades.

As the pipeline matures, the capital allocation strategy may be refined based on observed signal frequency, trade durations, and performance outcomes. For now, a simple allocation cap and a maximum number of open trades are used. If no suitable options meet the filtering criteria, the system can optionally fall back to executing the corresponding equity trade.

6.5 Future Enhancements

In later stages, once access to more comprehensive data is available, the following will be explored:

- Full Greek-based filtering and risk profiling (delta, theta, vega).
- Strategy selection based on IV surfaces and volatility arbitrage logic.
- Multi-leg, multi-strike strategies such as calendar spreads, straddles, or iron condors.

7 System Deployment

The pipeline is designed to run on a remote SSH-accessible server, allowing for reliable and automated execution with minimal manual intervention. The deployment architecture includes task scheduling, message alerts, logging, and persistent data storage for both historical and model-generated data.

7.1 Server Setup and Automation

A dedicated server is configured for daily and weekly pipeline execution using scheduled cron jobs. Weekly routines are used to update the Bayesian inference framework with newly observed cointegration test results and signal reliability metrics. Daily routines check for new data, update ECM models, recompute convergence probabilities, and generate trade signals.

7.2 Monitoring and Messaging

The system includes an alert service that sends compressed summaries of key actions and signal outputs to my phone via WhatsApp API. This allows for remote monitoring of system behavior and trading opportunities without requiring full manual access to logs or code execution. In case immediate intervention is needed, the system can be accessed remotely from anywhere via secure SSH, allowing for full control regardless of location.

7.3 Data Infrastructure

All data is stored in an organized structure on a dedicated server. Historical stock data, Monte Carlo forecasts, model outputs, and metadata are saved as structured DataFrames using formats such as Parquet or HDF5.

7.4 Execution Logging and Integrity Testing

Each component in the pipeline logs its execution, including timestamps, actions taken, and any signals generated. In addition, critical functions are monitored using a series of root-level tests—modular integrity checks designed to ensure key inputs, outputs, and intermediate steps behave as expected. These root tests act as health checkpoints for the pipeline and enable early detection of issues such as failed data downloads, invalid model fits, or stale input data.

8 Hardware Acceleration (FPGA)

This section explores the optional integration of an FPGA board to offload the Monte Carlo simulation component from the main pipeline. While not essential to the core functionality, this extension serves as a hands-on opportunity to gain experience with hardware acceleration and to evaluate potential improvements in latency and throughput. Monte Carlo simulations are well-suited to parallel execution and are expected to benefit from FPGA implementation. The offloading will be designed as a modular, self-contained component to allow for experimentation without affecting the stability or flexibility of the main system.

9 Risk Management and Practical Considerations

- Slippage modeling
- Transaction costs
- Model drift and retraining
- Execution reliability

10 Evaluation Metrics

- Strategy performance (PnL, Sharpe, Sortino)
- Forecast accuracy (RMSE, MAE)
- Signal quality (Precision, Recall)

11 Conclusion and Future Work

- Summary
- Next steps
- Long-term vision

A Appendix

- Definitions (ECM, cointegration, etc.)
- Software tools
- Sample tickers, results

References