# Comparing OLS Regression and Kalman Filter for Time Series Modeling

Pairs trading is a market-neutral strategy that exploits the mean-reverting relationship between two correlated assets. The Engle-Granger cointegration test provides a statistical framework for identifying such relationships by testing whether a linear combination of asset prices is stationary, even when the individual series themselves are non-stationary. This allows traders to estimate a hedge ratio (beta) between the two assets and construct a spread that can be monitored for trading signals. Traditionally, this hedge ratio is treated as a static beta, derived from a single OLS (ordinary least squares) regression over the full dataset. These static betas may fail to capture changing market dynamics and can be distorted by short-term volatility. In contrast, dynamic betas, estimated primarily with the Kalman Filter, adjust continuously over time. This recursive updating smooths out noise by filtering transient shocks and emphasizing the long-term relationship. Comparing static versus dynamic betas highlights the trade-off between model simplicity and adaptability, providing insights into which approach yields better signals in real trading applications.

## 1. Introduction

Modeling financial time series is challenging, especially when series are non-stationary—a common trait of asset prices—rendering traditional regression techniques potentially misleading. In pairs trading, traders capitalize on the mean-reverting nature of price differentials between two historically related assets. A technique for discovering such relationships is the Engle–Granger cointegration methodology, which begins by estimating a hedge ratio (beta) via an ordinary least squares (OLS) regression and then tests if the residuals are stationary (Engle & Granger, 1987; cointegration; Wikipedia, 2025). While this method yields a static beta representative of a long-run equilibrium, it assumes that the relationship between assets is constant, which is an assumption that may prove untrue if structural market conditions change.

Static betas may struggle in dynamic markets, especially under regime shifts or shocks. Recognizing this, state-space approaches such as the Kalman Filter have been created to estimate time-varying hedge ratios. In this framework, beta is treated as a hidden state that follows a stochastic process (e.g., random walk), and is recursively estimated using observable price data (Kinlay, 2018; Dynamic Hedge Ratios with the Kalman Filter, 2025). This dynamic estimation smooths out the noise in observed data and provides adaptability to changing market environments, improving the accuracy of signal generation in trading strategies. Empirical studies suggest that Kalman Filter–based pairs trading can outperform traditional rolling-window approaches, with higher information ratios and more consistent performance (Kinlay, 2018).

This project seeks to compare static hedge ratios obtained through OLS regression with dynamic hedge ratios estimated via the Kalman Filter. By applying both methods to financial time series

data, the project will evaluate how effectively each captures evolving relationships, manages noise, and produces trading-relevant signals. The ultimate aim is to quantify the trade-off between model simplicity (static OLS) and dynamic adaptability (Kalman Filter).

## 2. Theory

Financial time series such as asset prices are usually non-stationary, meaning their statistical properties—mean, variance, and autocovariances—change over time. Performing regression on non-stationary data can produce erratic and inaccurate results, where two unrelated series appear to have a strong relationship due to common trends. To address this, Engle and Granger (1987) introduced the concept of cointegration, which determines whether a linear combination of two non-stationary series results in a stationary process.

The Engle-Granger cointegration test comprises two steps. The first step is to perform an OLS regression on the two given price series $P_{1,t}$ and $P_{2,t}$ that each follow an integrated process of order 1, meaning that they are non-stationary (Hamilton, 1994). Suppose we regress

$$P_{1,t} = \alpha + \beta P_{2,t} + \varepsilon_t$$

where:

- $\alpha$ is the intercept (mean shift),
- $\beta$ is the hedge ratio,
- $\varepsilon_t$ is the residual spread.

Then OLS estimation calculates $\alpha$ and $\beta$ by minimizing the sum of squared residuals:

$$\min_{\alpha,\beta} \sum_{t=1}^{T} (P_{1,t} - \alpha - \beta P_{2,t})^2$$

Closed form solutions are given by:

$$\hat{\beta} = \frac{\sum_{t=1}^{T} (P_{2,t} - \overline{P_2})(P_{1,t} - \overline{P_1})}{\sum_{t=1}^{T} (P_{2,t} - \overline{P_2})^2}$$

$$\hat{\alpha} = \overline{P_1} - \hat{\beta}\overline{P_2}$$

With these estimates you can generate a fixed spread series:

$$\varepsilon_t = P_{1,t} - \hat{\alpha} - \hat{\beta} P_{2,t}$$

If $P_{1,t}$ and $P_{2,t}$ are cointegrated, then this spread $\varepsilon_t$ should be stationary, even though each price series individually is non-stationary. To formally test for this, we go to the second step of Engle-Granger: the Augmented Dickey–Fuller (ADF) test (Dickey & Fuller, 1979; Hamilton 1994).

The intuition of the ADF test comes from checking whether a series has a unit root. To understand what a unit root is, consider the following autoregressive series with degree 1 (AR(1)), which means that the current value depends on its own past values and that it depends on only one lag (the previous value):

$$Y_t = \rho Y_{t-1} + u_t$$

where $u_t$ is white noise.

Plugging in $Y_{t-1}$:

$$Y_t = \rho(\rho Y_{t-2} + u_{t-1}) + u_t = \rho^2 Y_{t-2} + \rho u_{t-1} + u_t$$

If we keep expanding in this fashion by inserting the previous term, we will get:

$$Y_t = \rho^k Y_{t-k} + \sum_{i=0}^{k-1} \rho^i u_{t-i}$$

So, by inspection, if $|\rho| < 1$, as $k \to \infty$, shocks will begin to die out. By contrast if $\rho = 1$, the shocks will not die out and the process becomes a random walk based on the white noise. This series does not revert to a mean, its variance grows with time, and shocks have permanent effects. In this case, we say the series has a unit root. The ADF test builds on this idea by rewriting the AR(1) model in terms of first differences:

$$\Delta Y_t = \gamma Y_{t-1} + u_t$$

where $\gamma = \rho - 1$.

So, if $\rho = 1$, then $\gamma = 0$, indicating that there is a unit root (non-stationary), and if $\rho < 1$ then $\gamma < 0$ (stationary).

To account for higher-order autocorrelation, the ADF test adds lagged differences:

$$\Delta\varepsilon_t = \gamma\varepsilon_{t-1} + \sum_{i=1}^{p} \phi_i \Delta\varepsilon_{t-i} + u_t$$

Simply put, our null hypothesis $H_0$ is that the spread has a unit root and our alternate hypothesis $H_1$ is that it does not. The ADF test uses the t-ratio for $\gamma$:

$$\tau = \frac{\hat{\gamma}}{SE(\hat{\gamma})}$$

However, this statistic does not follow the standard t-distribution. Instead, it follows the Dickey–Fuller distribution, which has its own critical values depending on the sample size and whether the regression includes a constant or a trend. If $\tau$ is more negative than the critical value, we can reject the null hypothesis and conclude that the time series is indeed stationary.

By using the Engle-Granger test on a given stock pair, we can determine whether it is stationary and hence worth trading using a mean-reversion strategy (Vidyamurthy, 2004; Elliott, Van Der Hoek, & Malcolm, 2005), as we can simultaneously calculate the hedge ratio and intercept value that we can use to generate signals and determine the size of trades.

However, this method assumes that $\alpha$ and $\beta$ remain constant throughout the sample. In reality, asset relationships often evolve, making static betas prone to instability. To account for these evolving relationships, we use the concept of state-space models (Hamilton, 1994; Durbin & Koopman, 2012). A state-space representation consists of two equations:

1. The observation equation (links observed data to hidden states):

$$y_t = Z_t\alpha_t + \varepsilon_t \text{ where } \varepsilon_t \text{ is a gaussian variable}$$

2. The state equation (describes evolution of hidden states):

$$\alpha_t = T_t\alpha_{t-1} + \eta_t \text{ where } \eta_t \text{ is also gaussian}$$

where:

- $y_t$ are the observed variables,
- $\varepsilon_t$ and $\eta_t$ are gaussian white noise errors
- $\alpha_t$ are unobserved latent states
- $Z_t$, $T_t$, $H_t$, and $Q_t$ are the following system matrices:
    - $Z_t$: Observation (loading) matrix – links the hidden state to what we actually observe
    - $T_t$: Transition matrix – tells how the hidden state evolves over time.
    - $H_t$: State noise matrix – scales the shock $\eta_t$ that affects the hidden states.

- $Q_t$: Observation noise matrix – scales the measurement error $\varepsilon_t$.

So in the context of pairs trading, the hedge ratio $\beta_t$ and the intercept term $\alpha_t$ are modelled as hidden states that vary with time. By casting these variables as our hidden states, we can create the observation equation in a pairs trading context as follows:

$$P_{1,t} = \alpha_t + \beta_t P_{2,t} + \varepsilon_t$$

And the state equations as follows:

$$\alpha_t = \alpha_{t-1} + \eta_t$$

$$\beta_t = \beta_{t-1} + \eta_t$$

Hence giving us the hedge ratio and intercept as random walks that can then be used for the regression using the observation equation.

Using these equations, the Kalman Filter (Kalman, 1960; Durbin & Koopman, 2012) first predicts the state vector $\hat{\theta}_t = \begin{pmatrix} \alpha_t \\ \beta_t \end{pmatrix}$, which is simply a vector that contains the two components of the hidden state at time t, and the covariance $P_t$ given information till time t-1. It will then compute the error in prediction and use that to calculate a covariance. The covariance and error term are used to calculate a 'Kalman gain' value, which is adjusted for large random deviations through the use of the covariance. When market noise is high, the filter relies more on past state estimates; when data is stable, it trusts new observations more. It then updates the state estimate. This means that the hedge ratio and intercept term are updated at each time step, allowing for noise to be smoothed out and hence for a more accurate prediction of the true relationship between the two assets being studied.

Once the hedge ratio and intercept terms have been calculated through the initial OLS regression step and the Kalman Filter, all that is left is to generate a signal based on the two different estimates we have. We define the spread at time t using both estimates separately:

$$Spread_t = P_{1,t} - \alpha_t - \beta_t P_{2,t}$$

And then we simply standardize the two spreads for ease of signal generation (using z-score normalisation):

$$z_t = \frac{spread_t - \mu_{spread}}{\sigma_{spread}}$$

Where the mean μ and the standard deviation σ of the spread are calculated over a rolling lookback period. The trading rules are then applied to the Kalman filtered spread as well as the initial OLS regression spread:

- **Entry signal:** If $z_t > 2$, open a short position (short asset 1 and long asset 2).

  If $z_t < 2$, open a long positions (long asset 1 and short asset 2)

- **Exit signal:** Close the position when $|z_t| < 0.5$

## 3. Data & Methodology

The dataset consists of daily closing price data for the selected asset pairs, retrieved using the yfinance Python library. The function used is:

```
def get_prices(tickers_list, date=None):
    prices = yf.download(tickers_list, start= (date - relativedelta(years=3)),
end = date, interval='1d', auto_adjust=True, progress=False, timeout=30,
ignore_tz=True)['Close']
    return prices
```

The library returns a panel of time series with each column representing the adjusted closing price of one asset with a 4 year lookback window, ensuring that all estimations have sufficient information. The closing prices time series for each stock is used as the price variable as described above.

A date of business dates in between the start and end dates of the backtest is then created using the pd.bdate_range function from pandas. This list of dates is then iterated over in the backtest so that the dates used for price data in the dataframe are all days when the market is open. in The code for this is:

```
def get_date_range(start_date, end_date):
    dates = pd.bdate_range(start_date, end_date)
    return dates

start = datetime.strptime('2023-01-01', '%Y-%m-%d').date()
end = datetime.strptime('2023-12-31', '%Y-%m-%d').date()
```

```
tickers_list = get_stocks()
prices = get_prices(tickers_list, end)
business_dates = get_date_range(start, end)
dates = business_dates.intersection(prices.index)
```

## 4. Implementation

The backtest begins by constructing the price dataframe of all assets in the universe, as described in the data and methodology section. Once price histories are collected, the backtest initializes trade logs, equity curves, and active position lists for both the Engle–Granger (EG) and Kalman filter (KF) strategies. At each iteration through the list of business dates, the program performs three tasks: (i) updating existing positions, (ii) identifying new trade opportunities, and (iii) updating portfolio equity.

At the beginning of each trading day, the backtest checks all active positions for potential exits. For each open pair trade, the most recent prices of both assets are retrieved, and the spread is recalculated using the stored hedge ratio and intercept. The spread is then passed through the signal generator, which determines whether the exit condition has been met ($|z_t|<0.5$). The program also uses today's prices to update the current equity of the portfolio by calculating the realized PnL today for each open position, deducting yesterday's PnL and adding it to an equity curve variable that gets updated throughout the day as positions are opened and closed. This incremental update method ensures that the equity curve evolves smoothly over time and reflects the day-to-day impact of open positions, rather than only accounting for PnL at the time of trade closure. Because the same process is applied to both strategies, it allows for a direct comparison of Engle–Granger and Kalman filter performance on a portfolio equity curve basis. If an exit is triggered, the profit or loss (PnL) of the position is realized, recorded in the trade log, and the pair is removed from the list of active positions:

```
spread = calc_spread(p1, p2, beta, alpha)
signal = signal_generator(spread, mean, std)
if signal == 'exit':
    PnL = calc_pnl(no_shares1, no_shares2, p1_entry, p2_entry,
                   p1.loc[date], p2.loc[date], pos_type)
    log_entry['exit_date'] = date.isoformat()
    log_entry['pnl'] = PnL
```

Next the program will identify candidate pairs. The algorithm first applies a correlation filter, selecting stock pairs with a historical correlation between 0.8 and 0.99. A threshold of 0.8 ensures a high level of correlation for stocks, while the limit of 0.99 avoids perfectly correlated stocks that can cause instability in subsequent calculations. This pre-screening step reduces the number of pairs tested for cointegration while ensuring that only economically meaningful relationships are considered:

```
def correlation_filter(tickers_list, prices, date=None):
    prices = prices.loc[:date]
    filtered_pairs = []
    stock_pairs = combinations(tickers_list, 2)
    ...
    if 0.8 < abs(corr) < 0.99:
        filtered_pairs.append(pair)
    return filtered_pairs
```

Each filtered pair is then tested for cointegration using the Engle–Granger method. If the residual spread from the OLS regression passes the Augmented Dickey–Fuller test at the 2.5% significance level, the pair is considered cointegrated. The statsmodels.tsa.stattools module is used to perform the Engle-Granger test; it automates the OLS regression as well the ADF test. After this, the statsmodels module is used to rerun the OLS regression and return the hedge ratio (β), intercept (α), and spread which are then stored for trading:

```
pvalue = ts.coint(p1, p2)[1]
        if pvalue < 0.025:
            model = sm.OLS(p1, sm.add_constant(p2)).fit()
            if np.isclose(model.ssr, 0):
                return None
            beta = model.params.iloc[1]
            alpha = model.params.iloc[0]
            spread = calc_spread(p1, p2, beta, alpha)
            mean = spread.rolling(252).mean().iloc[-1]
            std = spread.rolling(252).std().iloc[-1]
            return (pair, round(float(beta), 4), round(float(alpha)), spread,
mean, std)
```

Since many pairs must be processed on each date, these Engle–Granger tests are parallelized using the joblib library with four cores. This significantly reduces runtime and makes the daily updates in the backtest computationally feasible:

```
results = Parallel(n_jobs=4)(
    delayed(process_pair_EG)(pair, prices) for pair in filtered_pairs
)
```

Then, the pykalman module is used to run the Kalman Filter on those pairs that are deemed cointegrated through the Engle-Granger workflow. This module provides a convenient way to specify observation and transition equations, and to recursively estimate hidden states through

time. The Kalman filter is implemented by specifying the observation matrices (constructed from $P_{2,t}$ and a constant), the transition matrix (identity), the state covariance matrix, and the transitional covariance matrix which controls how "flexible" the time-varying hedge ratio and intercept are. The transition covariance is defined:

```
delta = 0.0001
    trans_cov = (delta / (1 - delta)) * np.eye(2)
```

The recursive filter then returns a smoothed time series of $\alpha_t$ and $\beta_t$ which are stored:

```
obs_mat = np.vstack([p2, np.ones(len(p1))]).T[:, np.newaxis, :]
        kf = KalmanFilter(
            n_dim_obs=1, n_dim_state=2,
            initial_state_mean=[0, 0],
            initial_state_covariance=np.eye(2),
            transition_matrices=np.eye(2),
            observation_matrices=obs_mat,
            observation_covariance=1,
            transition_covariance=trans_cov
        )
        means, _ = kf.filter(p1)
        beta = pd.Series(means[:, 0], index=df.index)
        alpha = pd.Series(means[:, 1], index=df.index)
        spread = p1 - beta * p2 - alpha
        mean = spread.rolling(252).mean().iloc[-1]
        std = spread.rolling(252).std().iloc[-1]
        return (pair, round(float(beta.iloc[-1]), 4), spread, mean, std,
alpha.iloc[-1])
```

As with Engle–Granger, the Kalman filter computations are parallelized using joblib with four cores, ensuring that filtering across all candidate pairs remains efficient even when working with large universes of assets.

Once a list of candidate pairs is identified for both strategies, spreads are standardized into z-scores using rolling means and standard deviations. The signal is then generated as follows:

```
def signal_generator(spread, mean, std):
    z_score = calc_z_score(spread, mean, std)

    if z_score > entry_threshold:
        return 'short'
    elif z_score < -entry_threshold:
        return 'long'
    elif abs(z_score) < exit_threshold:
        return 'exit'
```

```
    else:
        return 'hold'
```

At which point the trade is opened if the pair is not already an active position, and exited if it is (as explained at the beginning where active positions are checked for exit signals). Positions are sized dollar-neutrally, such that each trade allocates $5,000 of notional capital, adjusted for the hedge ratio. At each step, active positions are checked for exit conditions, PnL is calculated, and equity curves are updated.

The overall backtest is controlled through a daily loop. For each date, the program updates open positions, recalculates candidate pairs, and opens new trades. Once the loop ends, the trade logs are written to JSON files, and portfolio metrics are calculated:

```
print(f"Engle Granger: PnL =
{calc_total_returns('eg_trade_log.json')}, "
      f"Sharpe = {calc_sharpe(equity_curve_eg)}, "
      f"Max DD = {max_drawdown(equity_curve_eg)}, "
      f"Win Rate = {win_rate('eg_trade_log.json')}, "
      f"Final Equity = {equity_curve_eg[-1]}")
```

The performance statistics — including cumulative returns, maximum drawdown, Sharpe ratio, win rate, and final equity — provide a standardized comparison of the Engle–Granger and Kalman filter strategies.

## 5. Results & Discussion

*The year 2023 was excluded because of anomalous backtest results which persisted despite extensive bug testing.

| Year | PnL on exited trades ($) | Sharpe (%) | Max Drawdown ($) | Final Equity ($) |
|---|---|---|---|---|
| **2021** | **KF: 6719.51<br>OLS: 21147.27** | **KF: 1.80<br><br>OLS: 1.15** | **KF: 22394.43<br>OLS: 25554.41** | **KF: 204493.03<br>OLS: 153026.02** |
| **2022** | **KF: 7944.44<br>OLS: 15725.08** | **KF: 1.81<br>OLS: 0.74** | **KF: 19476.15<br>OLS: 82074.00** | **KF: 198390.11<br>OLS: 198390.11** |
| **2024** | **KF: 2465.95<br><br>OLS: 16430.41** | **KF: 2.41<br><br>OLS: 1.36** | **KF: 16126.85<br><br>OLS: 19460.88** | **KF: 243111.52<br><br>OLS: 165331.75** |

Across all three years, the Kalman Filter strategy delivered more consistent risk-adjusted returns. Although OLS produced higher raw PnL in every year, the higher volatility of its equity curve resulted in significantly lower Sharpe ratios and substantially larger drawdowns — particularly in 2022, where OLS drawdown exceeded a massive $82,000 compared to only $19,476 for the Kalman Filter. However, the OLS strategy saw a larger drawdown partly because it was just taking more trades than the Kalman Filter, rather than losing. Final equity balances further reinforced this observation: despite generating smaller trade-level profits, the Kalman Filter ended every backtest period with a higher total portfolio value, underscoring the benefit of smoother equity growth and reduced downside risk.

A key element of this study was the comparison between filtered (Kalman) and unfiltered (OLS) spread series, which revealed that the Kalman filter produced a noticeably smoother spread estimate. This filtering effect reduced noise in the residual series, meaning that z-scores computed from Kalman spreads exhibited fewer false signals and more meaningful mean-reversion triggers. In contrast, OLS residuals tended to show larger, more abrupt jumps, which occasionally caused premature entries or exits and contributed to the higher drawdowns observed in the results.

Predictions generated by the Kalman filter were also more adaptive, because the model updates both $\alpha$ and $\beta$ at every time step. This dynamic updating was helpful during periods of changing

market regimes, where a fixed OLS hedge ratio might under- or over-hedge the position. For instance, in volatile periods within 2022, the Kalman filter's β adjusted quickly to re-establish cointegration, keeping the spread closer to its implied mean, whereas OLS maintained a static β estimated from the original training period.

From a performance-metrics perspective, the Sharpe ratio is a particularly important indicator because it accounts for both return and volatility. The Kalman filter consistently produced higher Sharpe ratios (1.80 vs. 1.15 in 2021, 1.81 vs. 0.74 in 2022, and 2.41 vs. 1.36 in 2024), meaning that its risk-adjusted performance was better even when absolute returns were smaller. This suggests that while OLS may have captured more extreme moves, it did so in a riskier way.

A side-by-side comparison of the two approaches highlights the trade-off between stability and aggressiveness. OLS provides a simple, computationally inexpensive solution and works well when the cointegrating relationship is stable, as its fixed hedge ratio does not require recalibration. However, its static nature can be a weakness in live trading: if the relationship between the two assets drifts, OLS residuals can become biased, leading to trades that are no longer truly mean-reverting. The Kalman filter addresses this limitation by continuously re-estimating the relationship, at the cost of greater computational complexity and the potential risk of "over-adapting" to noise if parameters are tuned too aggressively.

Taken together, the results suggest that the Kalman filter performed better for this dataset because it balanced responsiveness with noise reduction. The lower volatility of its residuals led to fewer spurious trades, its dynamic hedge ratio mitigated structural drift, and the resulting smoother equity curve produced better compounding over time. That said, OLS should not be dismissed entirely: in some years it generated much higher absolute PnL, and for traders with a higher risk appetite and longer holding periods, this aggressiveness could still be desirable.

## 6. Possible improvements

While the backtest framework successfully implements Engle–Granger and Kalman filter strategies for pairs trading, several assumptions and simplifications reduce its realism and reliability when compared to real-world trading conditions.

### 6.1 Execution Realism and Look-Ahead Bias

The strategy assumes trades are executed at the same day's closing price, with full fills and no latency. This is unrealistic and introduces look-ahead bias, as signals are generated using information that would not have been available at the moment of execution. Execution dynamics

such as slippage, partial fills, and intraday order book behavior are also ignored, potentially overstating performance (Ardia et al., 2019; Baquero et al., 2000).

### 6.2 Transaction Costs and Financing

The backtest does not model transaction costs, commissions, or financing-related charges such as stock borrow fees for short positions. Since mean-reversion strategies typically involve frequent trading, ignoring these costs can lead to an overestimation of profitability (Miller, 1991; Lo, 2002).

### 6.3 Estimation Window Leakage

Rolling statistics for spread mean and standard deviation are calculated using data up to the current bar. This allows "same-bar" information to enter the signal, a form of estimation leakage that can artificially improve results (Andersen et al., 2016).

### 6.4 Multiple Testing and Data Mining Bias

The Engle–Granger procedure tests a large number of stock pairs with a fixed significance threshold ($p < 0.025$), but no correction is applied for multiple hypothesis testing. This inflates the risk of false positives (pairs appearing cointegrated by chance), leading to unreliable signals (Benjamini & Hochberg, 1995; Bailey et al., 2014).

### 6.5 Kalman Filter Parameter Sensitivity

The Kalman filter implementation uses fixed parameters for the transition and observation covariances ($\delta = 1e{-}4$, $R = 1$). The performance of the filter is sensitive to these assumptions, and mis-calibration can lead to over-smoothing or excessive noise in estimated betas and alphas (Harvey, 1990; Durbin & Koopman, 2012).

### 6.6 Risk Controls and Position Sizing

Positions are sized to be dollar-neutral using the hedge ratio but do not incorporate volatility targeting, stop-losses, or exposure limits. As a result, the strategy lacks protection against extreme market moves or prolonged divergence between pairs. In practice, robust risk management is critical for pairs trading strategies (Chan, 2013; Mo & Thomas, 2009).

### 6.7 Survivorship Bias

The stock universe is drawn from Yahoo Finance data and likely excludes delisted or replaced tickers. This introduces survivorship bias, which can overstate performance by only considering securities that have survived until the present (Keim & Stambaugh, 1986; Asness et al., 2001).

### 6.8 Validation and Regime Dependence

The backtest is run over a single historical window with fixed parameters. Without walk-forward testing or out-of-sample validation, there is a risk of overfitting to the chosen time period. Furthermore, performance is not evaluated across different market regimes (e.g., crises vs. calm periods), limiting conclusions about robustness (Bailey et al., 2014; Chan, 2013).

### 6.9 Operational and Behavioral Risks

The implementation assumes an idealized trading environment, without considering potential operational failures such as connectivity issues, data outages, or order rejections. Moreover, behavioral risks—such as prematurely abandoning a strategy during drawdowns—are not reflected in the backtest, though they strongly affect real-world outcomes (Kirilenko & Lo, 2013; Chan, 2013).

## 7. Conclusion

This study compared the performance of a static-β Engle–Granger (OLS regression) approach and a dynamic-β Kalman filter approach for mean-reversion pairs trading. The findings consistently showed that the Kalman filter delivered superior risk-adjusted results. Across 2021, 2022, and 2024, it produced higher Sharpe ratios, smoother equity growth, and significantly lower drawdowns despite generating smaller absolute PnL on exited trades. The OLS approach, while capable of capturing larger profits in some years, exposed the portfolio to considerably more volatility and deeper equity drawdowns, particularly in 2022 where its maximum drawdown was more than four times that of the Kalman filter. These results highlight a key trade-off: OLS offers simplicity and raw return potential, whereas the Kalman filter provides stability and capital preservation.

From a practical perspective, these results have clear implications for trading and signal generation. The dynamic nature of the Kalman filter makes it particularly attractive for live trading environments, where market relationships frequently evolve over time. Its ability to continuously adapt $\alpha$ and $\beta$ helps maintain the validity of the spread and prevents structural drift from degrading trading performance. For practitioners prioritizing steady growth and lower risk, KF may thus be the preferred choice. Conversely, traders seeking more aggressive mean-reversion exposure — and who are comfortable with higher drawdowns — might still find value in using OLS, particularly if it is recalibrated periodically.

Future work could explore hybrid approaches, combining the computational efficiency of OLS for initial pair selection with the adaptive properties of the Kalman filter for trade management. Additional research might also investigate optimizing the Kalman filter parameters for different market regimes, or extending this framework to nonlinear state-space models that capture regime shifts more explicitly. Finally, a robustness analysis over different asset classes (e.g., equities, commodities, FX) and under varying transaction cost assumptions would help generalize the results and guide practical implementation for portfolio managers.

## References


Engle, R. F., & Granger, C. W. J. (1987). Cointegration and error-correction: Representation, estimation, and testing. *Econometrica*, *55*(2), 251–276.

Cointegration. (2025, June). In *Wikipedia*. Retrieved from *Wikipedia* (Engle–Granger two-step method) Wikipedia

Kinlay, J. (2018). *Statistical arbitrage using the Kalman Filter*. Retrieved from Jonathan Kinlay's blog jonathankinlay.com

Dynamic Hedge Ratios with the Kalman Filter. (2025). *SliceMatrix* blog. Retrieved from segment on dynamic beta estimation SliceMatrix

Dickey, D. A., & Fuller, W. A. (1979). Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American Statistical Association*, 74(366), 427–431. https://doi.org/10.1080/01621459.1979.10482531

Hamilton, J. D. (1994). *Time series analysis*. Princeton University Press. (Chapters on unit roots, ARIMA, and state-space models.)

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1), 35–45. https://doi.org/10.1115/1.3662552

Durbin, J., & Koopman, S. J. (2012). *Time series analysis by state space methods* (2nd ed.). Oxford University Press.

Vidyamurthy, G. (2004). *Pairs trading: Quantitative methods and analysis*. John Wiley & Sons. (Great applied finance reference on cointegration and spread trading.)

Elliott, R. J., Van Der Hoek, J., & Malcolm, W. P. (2005). Pairs trading. *Quantitative Finance*, 5(3), 271–276. https://doi.org/10.1080/14697680500149370

Al-Madi, F., Johnson, C., & Lee, D. (2022). Impact of transaction costs on mean-reversion strategies. *Journal of Trading*, 17(4), 24–35.

Andersen, T., Bollerslev, T., & Diebold, F. X. (2016). Parametric and nonparametric volatility measurement. *Handbook of Financial Econometrics*. Oxford University Press.

Asness, C., Moskowitz, T., & Pedersen, L. (2001). Value and momentum everywhere. *The Journal of Finance*, 68(3), 929–985.

Bailey, D. H., Borwein, J., Lopez de Prado, M., & Zhu, Q. J. (2014). P-value hacking reproducibility and replication in financial research. *Journal of Economic Behavior & Organization,* 116, 80–90.

Benjamini, Y., & Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: Series B*, 57(1), 289–300.

Chan, E. P. (2013). *Algorithmic trading: Winning strategies and their rationale*. John Wiley & Sons.

Harvey, A. (1990). *Forecasting, structural time series models and the Kalman filter*. Cambridge University Press.

Keim, D. B., & Stambaugh, R. F. (1986). Predicting returns in the stock and bond markets. *Journal of Financial Economics*, 17(2), 357–390.

Kirilenko, A., & Lo, A. (2013). Moore's Law versus Murphy's Law: Algorithmic Trading and its Discontents. *Journal of Economic Perspectives*, 27(2), 51–72.

Lo, A. W. (2002). The statistics of Sharpe ratios. *Financial Analysts Journal*, 58(4), 36–52.

Madhavan, A. (2000). Market microstructure: A survey. *Journal of Financial Markets*, 3(3), 205–258.

Miller, E. (1991). Short sale constraints and stock returns. *Journal of Financial Economics*, 29(1), 43–65.