Cointegration Aware Pairs Trading with Reinforcement Learning Based Optimal Stopping

Tudor Stefan Pagu[1]
Supervisor(s): Fenghui Yu[1]**, Frans A. Oliehoek**[1]
[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 21, 2026

Name of the student: Tudor Stefan Pagu
Final project course: CSE3000 Research Project
Thesis committee: Fenghui Yu, Frans A. Oliehoek, Neil Yorke-Smith

Abstract

Pairs trading is a type of algorithmic trading strategy that exploits temporary divergences between assets that tend to follow each other, which we describe as cointegrated. As a special case of statistical arbitrage, it has long been studied by both practitioners and academics. We hypothesize that a common failure of existing pairs trading strategies is their behavior when the cointegration relationship is not constant over long periods of time, which is often the case in practice. We show that given future knowledge of the cointegration relation, a strategy can yield dramatically better returns, up to 16% annualized during cointegration periods. This finding motivates a data-driven approach for estimating the cointegration regime. To solve this problem, we propose a GRU model that tracks the cointegration regime better than chance, though its reliability varies substantially by pair. We trained an RL model to exploit cointegration periods using synthetic data, and experimented with limiting trading to only periods of predicted cointegration. We tested this on three commonly used pairs and found it outperformed the risk-free rate, with Sharpe ratios of 0.30–0.65. Our work shows the potential of cointegration-aware approaches through an oracle analysis, proposes a way to approximate it in a realistic strategy, and identifies current limitations of the model.

1 Introduction

The usage of machine learning in quantitative finance has long been studied in an attempt to find profitable strategies and better understand the optimal allocation of capital in the stock markets [21, 2, 4, 26, 18]. In this paper, we focus on pairs trading, which is a particular case of statistical arbitrage. We apply machine learning techniques such as reinforcement learning and a recurrent neural network in order to increase the robustness of an algorithmic trading strategy.

The first step of pairs trading consists in identifying two (or more) assets which the practitioner believes to be highly correlated (for example, imagine Albert Heijn and Jumbo). The two main methods for choosing pairs are distance based [12] and cointegration based [10]. Since we expect such stocks to revert to a common price (up to a linear coefficient adjusting for relative size), the strategy is to wait until they diverge, take a long position in the undervalued asset and a short position in the overvalued one, and close both once they converge. As long as we enter when the assets are far apart and exit when they are close, we profit, ignoring transaction costs. The risk is that the cointegration relationship breaks down: the assets may never reconverge, causing losses, or reconverge too slowly to beat the risk-free rate.

The second step, once a pair is selected, is deriving a strategy for when to enter and exit the position. The most basic approach computes the "spread" or difference in price between the two stocks, and uses the z-score of the current spread as a signal [12, 10] against predetermined fixed thresholds. For example, the strategy may enter when the absolute value of the z-score exceeds 2 and exit when it falls below 0.5. While simple and often surprisingly effective, this can suffer from not being adaptive enough. Reinforcement learning has been proposed to address this: Ning and Lee [20] applied Q-learning to the entry/exit decision, and Kim and Kim [16] used deep Q-learning to select among predefined threshold sets.

We evaluated several RL approaches for optimal stopping in pairs trading. We found mixed results, with many strategies not out-performing the risk free rate. However, they also did not lose money, often reaching yearly returns of around 4%. We hypothesise that the reason why the strategies sometimes failed is that the cointegration relationship would often break down across different long term periods. For example, one of the pairs we

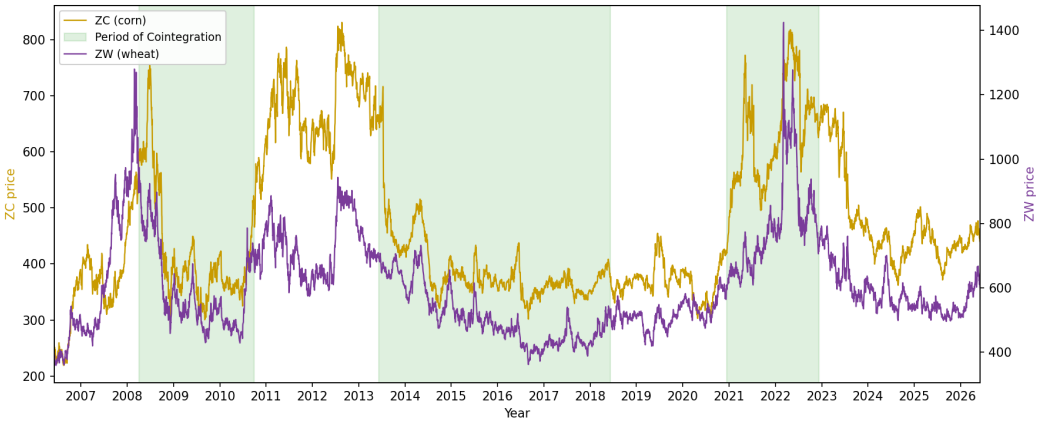


Figure 1: Periods of cointegration for the ZC/ZW pair (corn and wheat futures). Green windows are 2 year periods in which the spread passed all three criteria for being cointegrated: ADF rejected its unit-root null ($p < 0.05$), KPSS failed to reject its stationarity null ($p > 0.05$), and the half-life of mean reversion fell between 5 and 60 days.

tested (corn and wheat futures) had 3 distinct periods of cointegration between 2008-2011, 2014-2018, and 2021-2023 (See Figure 1). This suggested that we should only expect a pairs trading strategy to be consistently profitable if it was trading throughout these periods. This motivated experimenting with a RL agent that only trades during these periods, as if given a *cointegration oracle*. The concept of benchmarking against an idealized trader with perfect model knowledge has precedent in pairs trading: Chiu and Wong [6] define an oracle strategy in which the true parameters of the cointegration model are assumed known, serving as an upper bound for classical stochastic control approaches. We extend this spirit to a different notion of oracle — one that knows the periods over which cointegration holds — and apply it to evaluate RL strategies. To our knowledge, this is the first work considering the impact of a long term cointegration oracle of this kind for a reinforcement learning strategy.

We found that the oracle-informed agent significantly outperformed an agent without the oracle, despite the oracle providing no information about future prices — only whether the current moment fell within a cointegrated period during which trading was permitted. This suggests that identifying periods of cointegration is key to the performance of a pairs trading strategy. However, the approach is not directly applicable to live trading, as the oracle is offline: it uses future data to determine whether a given moment is cointegrated.

We applied the insights from the oracle experiments to develop an online trading strategy that does not rely on future data. We trained a Gated Recurrent Unit (GRU) [7] to predict, using only past market data, whether the pair is currently in a period of cointegration, with periods labelled according to the cointegration oracle. We found that the GRU-informed agent outperformed the cointegration-unaware RL baseline and consistently beat the risk-free rate. As expected, it did not match the oracle-informed agent, which serves as an upper bound.

Research Questions

The research questions this work aims to answer are:

- **RQ1.** How does reinforcement learning perform compared to fixed threshold methods for pairs trading optimal stopping, accounting for transaction costs?
- **RQ2.** Using offline analysis of the full time series, can we identify the periods over which a pair is cointegrated, and does restricting an RL agent to trade only during these periods improve performance?
- **RQ3.** Can we approximate the cointegration period detection from **RQ2** using only data available up to the present moment, making the strategy viable for live trading? How does this online detection affect RL agent performance compared to the oracle case?

Thesis Outline

The remainder of this thesis is structured as follows. Section 2 presents related literature on pairs trading and reinforcement learning applications in finance. Section 3 gives an overview of background concepts related to this thesis, touching on relevant financial concepts, pairs trading, reinforcement learning ideas such as Double DQN, and Gated Recurrent Units. Section 4 presents our methodology in depth, explaining the model architecture, training parameters, and data selection or generation. Section 5 shows the experiments we performed and the results we observed. Section 6 gives a wider discussion of our results, answering our original research questions. Section 7, "Responsible Research", covers the ethical implications of our work. Section 8 concludes the thesis and proposes future work.

2 Related Work

Classical pairs trading. Pairs trading as a strategy dates back several decades, having long been practiced by traders. Gatev et al. [12] present a seminal methodology for identifying pairs via a distance-based method which minimizes the sum of squared deviations, using static z-score thresholds to trigger trades. One drawback of this approach is that it assumes a 1:1 relationship between assets, ignoring that they may co-move at different scales. Vidyamurthy [23] address this by introducing a cointegration-based framework, in which OLS is used to estimate a hedge ratio, and the stationarity of the resulting residual is tested to confirm a long-run equilibrium relationship.

Reinforcement learning for trading. Ning and Lee [20] replaced static thresholds with reinforcement learning, modelling the state space as discretized spread movements over the past 4 timesteps and the action space as entering, holding, or exiting the position, applying Q-learning to outperform fixed thresholds. Kim and Kim [16] used DQN not to determine entry and exit directly, but to select from a set of predefined fixed thresholds for an entire episode, combining the reliability of static thresholds with the adaptability of reinforcement learning. Yang et al. [26] presented a more general framework for RL-based algorithmic trading using an ensemble of PPO, A2C, and DDPG, without restricting the agent to a pairs trading structure. Han et al. [13] applied a bidirectional GRU with a temporal attention mechanism in order to enhance the RL-based approach.

Optimal stopping. Becker et al. [2] tackled optimal stopping using a deep learning method in an offline setting, training a separate network at each timestep and working

backwards through the time series to estimate the value of continuing. Although the offline assumption makes this inapplicable to live trading, it motivates our use of an oracle setting as an upper bound before considering the online case.

Structural breaks in pairs trading. Lu et al. [18] addressed the problem of structural breaks—periods where the cointegration relationship breaks down—by training an LSTM to detect them from labelled intraday minute data. Our work extends this concern to daily data over horizons of 1–2 years, where cointegration may hold only during distinct multi-year subperiods.

3 Background

In this section, we give an overview of the background needed for our method: pairs trading, transaction costs, optimal stopping, reinforcement learning, and recurrent neural networks.

3.1 Pairs Trading

A pairs-trading strategy assumes that two or more stocks are *cointegrated*; we describe the two-stock case. Let A_t, B_t be the stock price series. If they are cointegrated, then by definition there exist $\alpha, \beta \in \mathbb{R}$ such that the *spread*

$$S_t = \log(A_t) - \beta \log(B_t) + \alpha \tag{1}$$

is stationary and mean reverting: its distribution is constant over time and it reverts to its mean faster than a random walk. Log prices are used because price movements are naturally multiplicative, and the logarithm transforms this into the additive relationship that cointegration assumes. We estimate the intercept α and hedging coefficient β by ordinary least squares; β always refers to this hedging coefficient. The *z-score of the spread* is

$$z_t = \frac{S_t - \mu}{\sigma} \tag{2}$$

where μ and σ are the mean and standard deviation of S_t over a trailing window.

The goal of pairs trading is to take a simultaneous long/short position when the stocks diverge and close it when they converge. Entering at a larger absolute spread than we exit guarantees a profit, ignoring transaction costs. The strategy is *market neutral*: we bet only on the spread reverting to its mean, not on the direction of either stock.

Kalman Filter. A standard technique for estimating β is the Kalman filter [15, 10]. It models the hedging coefficient as a latent state following a random walk, observed noisily through the log-prices, and alternates between predicting the state and correcting it with each new observation. This gives a causal, time-varying estimate of β_t that adapts as the relationship between the assets drifts, using only information available up to time t.

3.2 Transaction Costs and Liquidity Assumptions

We impose a flat 0.1% transaction cost on every action: buying \$100 of an asset costs \$100.1. We also assume unlimited liquidity: any quantity can be bought or sold at the daily price without moving it, subject only to available capital. Both simplifications are standard in the literature [26, 25, 16] and let us focus on the strategy itself.

3.3 Optimal Stopping

We frame our pairs trading strategy as an optimal stopping problem: our goal is to find optimal moments to enter and exit our position, entering when the spread is high and exiting when it is low. As in [20, 16], we constrain ourselves to entering or exiting our entire position at once, assuming the trade executes instantly and completely, and we hold a position until we exit without re-hedging.

3.4 Reinforcement Learning and Deep Q-Learning

We model the environment as a Markov decision process (MDP) [20]. At each step, the agent observes a *state* $S_t \in S$, selects an action $A_t \in \mathcal{A}$, and receives a reward $r \in \mathbb{R}$, transitioning to a new state determined by the environment. The goal is to find a policy π that maximises the expected cumulative reward over an episode. Q-learning estimates the value of taking action a in state s via the Bellman equation,

$$q_*(s,a) = \mathbb{E}[r + \gamma \max_{a'} q_*(s',a') \mid s,a],$$

where γ is the discount factor. Deep Q-learning [19] replaces the tabular q_* with a neural network trained on $(s,a) \to (s',r)$ samples using an *epsilon-greedy* policy. Double DQN [22] improves on this by decoupling action selection from evaluation: an online network Q selects actions while a periodically-frozen target network Q' evaluates them, reducing the value over-estimation that harms standard DQN. We use Double DQN throughout this work.

3.5 Recurrent Neural Networks and Gated Recurrent Units

Recurrent Neural Networks [11] carry information forward across time steps, but classic RNNs struggle to learn long-term dependencies due to vanishing or exploding gradients [3]. Long Short-Term Memory (LSTM) networks [14] address this with gating mechanisms that let the network selectively retain or forget information. Gated Recurrent Units (GRUs), introduced by Cho et al. [7], simplify the LSTM by merging the forget and input gates into a single update gate and combining the cell and hidden states, giving fewer parameters while retaining the ability to capture long-term dependencies. GRUs often perform comparably to LSTMs despite their reduced complexity [8], which motivates their use in our cointegration detector, as their reduced size should lead to less overfitting on our limited training data.

4 Pairs Trading With Cointegration Aware Reinforcement Learning

In this section, we present our proposed method for pairs trading with cointegration aware reinforcement learning.

4.1 Reinforcement Learning Model

We model our optimal stopping problem as an MDP, where a given state is made up of the following observations:

$$s_t = \left[\ \tilde{s}_t,\ \ z_t^{(10)},\ z_t^{(45)},\ z_t^{(100)},\ z_t^{(200)},\ \ p_t\ \right] \in \mathbb{R}^6, \tag{3}$$

where:

- $\tilde{s}_t = \log P_t^A - \beta_t \log P_t^B + \alpha_t$ is the *spread* between the two legs, with β_t and α_t the hedge ratio and intercept estimated by the Kalman filter. (The method for determining α and β will be discussed below).

- $z_t^{(w)} = \dfrac{\tilde{s}_t - \mu_t^{(w)}}{\sigma_t^{(w)}}$ for $w \in \{10, 45, 100, 200\}$ are the *normalised deviations* (z-scores) of the spread, where $\mu_t^{(w)}$ and $\sigma_t^{(w)}$ are the rolling mean and standard deviation of the spread over the trailing w trading days. The multiple horizons let the agent perceive both short-term dislocations and the longer-term level of the spread;

- $p_t \in \{-1, 0, +1\}$ is the agent's *current position* (short, flat, or long the spread), which makes the policy aware of its own holdings and of the transaction cost it would incur by trading.

The set of possible actions is $\mathcal{A} = \{\text{long}, \text{short}, \text{neutral}\}$. Being *long* the spread corresponds to buying stock A and shorting stock B, being *short* the spread is the reverse, and being neutral refers to owning no stock and keeping the money in a risk-free bank account. In our model, the money kept in the bank account accrues interest daily, equivalent to a yearly risk-free interest rate of 5%.

The reward is defined as the per timestep change in total account wealth:

$$r_t = W_{t+1} - W_t \tag{4}$$

Account wealth is computed as cash plus current market value of assets owned minus transaction costs:

$$W_t = c_t + n_t^A P_t^A + n_t^B P_t^B - \kappa V_t. \tag{5}$$

where:

- c_t is the cash balance (earning the daily risk-free rate while flat);

- n_t^A and n_t^B are the signed unit holdings of assets A and B (negative when short);

- P_t^A and P_t^B are their prices;

- $\kappa = 0.1\%$ is the proportional transaction-cost rate, and V_t is the notional traded at step t (zero when the position is unchanged).

4.2 Training the RL Model

Train/Test Splits

For all pairs, we split the pair's historical data into a train half followed by a test half. The train half is used for all calibration of hyper-parameters, such as the β and α used for synthetic data generation. The test split is only used for evaluation, and is not used for any training or fitting of parameters. For the remainder of the text, this is what we refer to as *train* or *test* data (See Figure 2 for a visualization of this).

Figure 2: ZC/ZW (Corn/Wheat futures) plotted with a train/test split at the middle. Spread is computed with a static beta using OLS on the train half (this is for simplicity, but other methods are used in the rest of the paper for β).

Training with Synthetic Data

In order to avoid overfitting on the limited real data we have, we rely on synthetic data to teach the RL agent how to profit from a mean reverting spread, then evaluate it on real data. (See Appendix A for the impact of real vs synthetic training data on our method).

To generate the synthetic data, we fit an AR(1) model to the training windows from the real data, yielding persistence parameter $\hat{\phi}$ and innovation variance $\hat{\sigma}_\varepsilon^2$. We then generate $N = 50$ independent synthetic series of length T by simulating

$$\xi_t^{(i)} = \hat{\phi}\,\xi_{t-1}^{(i)} + \varepsilon_t^{(i)}, \qquad \varepsilon_t^{(i)} \sim \mathcal{N}(0, \hat{\sigma}_\varepsilon^2), \tag{6}$$

and constructing $\log S_0^{(i)}$ as a random walk with drift paired to $\xi_t^{(i)}$ via Eq. (1). At each episode reset the environment samples one of the N series uniformly at random, providing diverse mean-reverting dynamics. See Figure 3 for a visualization of such a spread.

We train the agent on a total of 200 episodes of length 378 steps each, which corresponds to 1.5 years of daily data (which approximately corresponds to the long term time horizon we are aiming for). Figure 4 shows the training curves as DDQN learns.

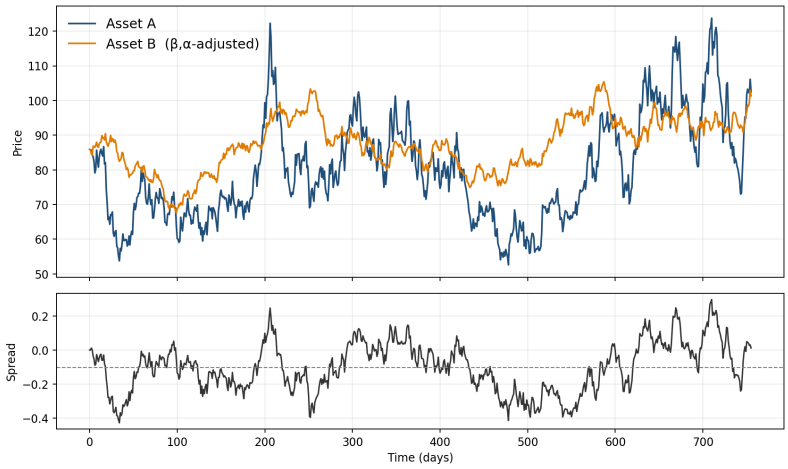


Figure 3: Example of synthetic spread. Top plot shows the generated stock prices themselves, while the bottom plot shows the spread between them.

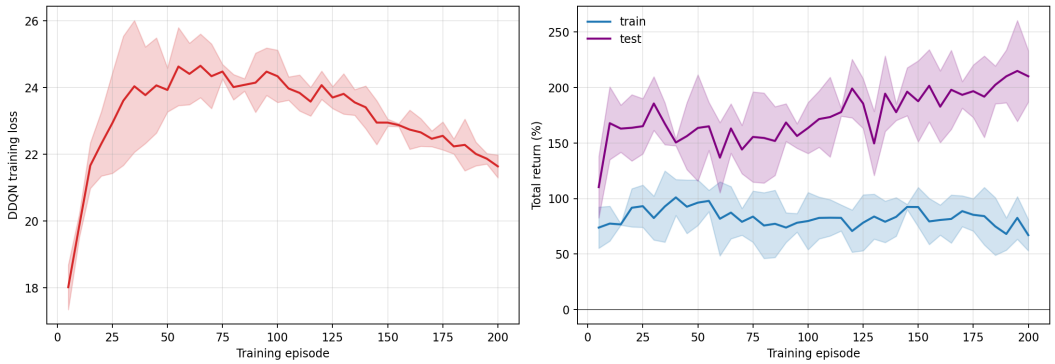


Figure 4: Training loss and accuracy curve of DDQN RL agent. Left graph shows the DQN online network's Huber loss over training. Right graph shows the total reward (total return over an episode) as training progresses, evaluated on both the train and the test (unseen) set. Data is aggregated from 3 runs with random seeds on the ZC/ZW pair.

4.3 Cointegration Awareness and the Oracle Informed RL Agent

A key contribution of this work is our method for determining periods of long-term cointegration and their impact on the performance of the RL agent. Our hypothesis is that pairs go through long-term periods of cointegration which periodically break. It is these breaks that can cause pairs trading strategies to under-perform. By restricting our RL agent to trade only in periods where the pairs are indeed cointegrated, we would expect a significant gain in returns.

To determine whether a spread is stationary within a given window, we combine two complementary stationarity tests. The Augmented Dickey–Fuller (ADF) test [9] takes non-stationarity as its null hypothesis: it tests for the presence of a unit root, so rejecting the null

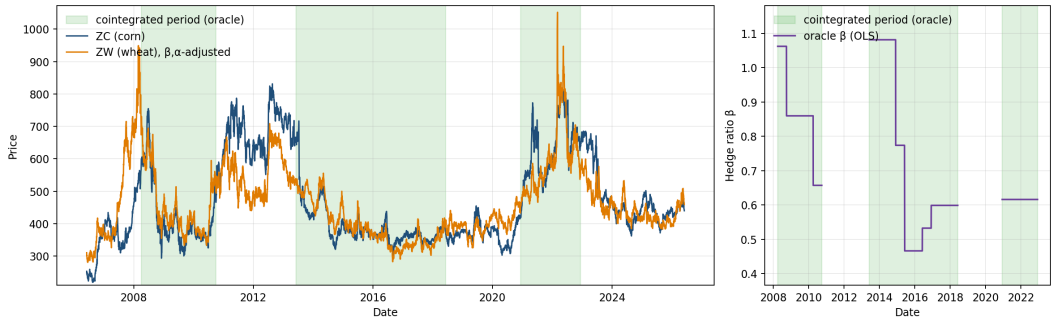


Figure 5: Output of cointegration oracle. The left plot shows periods where both the ADF and KPSS tests pass in green. The right plot shows the β computed as the mean of OLS betas across those windows.

($p < 0.05$) provides evidence that the spread is stationary and therefore mean-reverting. The Kwiatkowski–Phillips–Schmidt–Shin (KPSS) test [17] is constructed the other way around, taking stationarity as its null; here we want to *fail* to reject ($p > 0.05$). Because the two tests reverse the roles of the null and alternative hypotheses, requiring both to agree—ADF rejecting and KPSS not rejecting—yields a more robust determination of stationarity than either test alone, guarding against the low power that each can exhibit in isolation [17]. We moved a 2-year sliding window across the history of our pair, moving it forward by 2 months each time. For each window, we ran an ADF and KPSS test, marking the window as cointegrated only if both tests succeeded. This resulted in clear periods of cointegration with breaks in between, which sometimes themselves lasted for years (See Figure 5).

Half-life of mean reversion. To quantify how fast a spread reverts to its mean, we fit a first-order autoregression $S_t = \varphi S_{t-1} + c + \varepsilon_t$ by OLS, where $\varphi \in (0, 1)$ is the day-to-day persistence of a deviation. The corresponding half-life [5] is $h = -\ln 2 / \ln \varphi$, the number of days for a deviation to decay by half. We retain only windows with $5 \leq h \leq 60$ days, excluding spreads that revert too quickly to trade or too slowly to be useful.

In order to estimate the hedging coefficient β, we ran an OLS (Ordinary Least Squares linear regression) on each cointegrated window, and averaged these betas for time-steps that were part of multiple windows.

Oracle Informed RL Agent

We tested the RL agent described in Section 4.2, with the only addition that we give it the β from the oracle instead of the Kalman filter β, and we *force* it to only trade during periods the oracle says are cointegrated. That is, we are forced to keep our money in the bank account when the pairs are not cointegrated, and only defer to the RL agent during periods of cointegration. We found this **significantly** improved performance, serving as a theoretical upper bound.

Unfortunately, this method is not directly applicable to trading, as the oracle relies on future information to determine whether a given current moment is in a cointegration period or not and for computing the β. Therefore, the remainder of our work was focused on estimating the output of the oracle given only previous information, which can be approached as a supervised learning problem with labels generated by the oracle.

4.4 Gated Recurrent Units for Cointegration Awareness

We utilized a Gated Recurrent Unit (GRU) network to approximate the oracle's output using only information available up to the current time step. Since the oracle provides a binary cointegration label at every time step, this reduces to a supervised sequence-classification problem: at each time t, predict whether the pair is currently in a cointegrated regime, using the oracle's labels as ground truth.

Inputs. At each time step the network receives a vector of seven *causal* features, computed over a trailing window of $W = 128$ trading days:

- the p-values of the ADF and KPSS stationarity tests on the spread;
- the estimated half-life of mean reversion (Computed as in Section 4.3);
- the rolling correlation between the two log-price series;
- the rolling standard deviation of the spread;
- the Hurst exponent of the spread; and
- the log-price spread level $\log P_t^A - \log P_t^B$.

These features were selected as plausible indicators of the current cointegration regime.

Architecture. The network is a single-layer, unidirectional GRU with a hidden state of size 32, followed by a linear layer and a sigmoid activation that outputs, at each step, the probability $\hat{p}_t \in [0, 1]$ that the pair is cointegrated. Because the sequence is processed in a single forward pass, $\hat{p}_t$ depends only on inputs up to time t, making the network deployable as a real-time gate; a cointegrated regime is declared when $\hat{p}_t > 0.5$.

Training. The network is trained on the training half to match the oracle labels $y_t \in \{0, 1\}$, using a binary cross-entropy loss augmented with a total-variation penalty that encourages temporally smooth, non-flickering predictions:

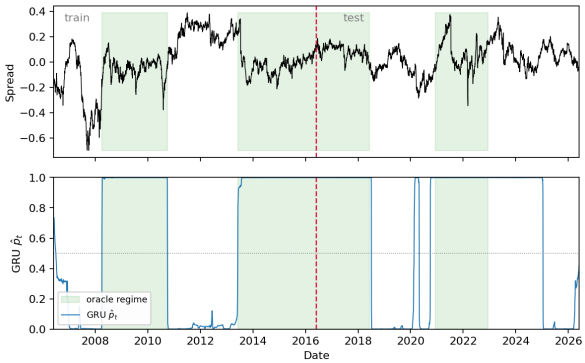
$$\mathcal{L} = \frac{1}{T} \sum_t \mathrm{BCE}(\hat{p}_t, y_t) \; + \; \lambda \frac{1}{T} \sum_t \left| \hat{p}_t - \hat{p}_{t-1} \right|, \qquad \lambda = 2. \tag{7}$$

We optimise with Adam (learning rate 3×10^{-3}) for 300 epochs, processing the sequence with truncated backpropagation through time.

Figure 6 reports the GRU's agreement with the oracle labels and visualizes its predictions on one pair. On average the detector is better than chance (mean AUROC 0.53–0.68), but its skill is modest and, more importantly, highly sensitive to the random seed: on V/MA and ZC/ZW individual runs range from well below chance (≈ 0.28) to strong (≈ 0.82). This instability of the learned detector is a key limitation of our current method.

4.5 Cointegration Aware RL Agent

Finally, our proposed model (See Figure 7) utilizes the GRU model to predict whether or not we are in a period of cointegration, then, if we are, it trades according to the RL model described in Section 4.2, with a spread computed with β from a Kalman filter. One detail is



	V/MA	ZC/ZW	EWA/EWC
AUROC	0.53 ± 0.16	0.59 ± 0.22	0.68 ± 0.05
Accuracy	0.73 ± 0.16	0.64 ± 0.09	0.68 ± 0.04
Recall	0.86 ± 0.15	0.51 ± 0.36	0.55 ± 0.09
Precision	0.82 ± 0.08	0.53 ± 0.17	0.78 ± 0.14
Overlap (IoU)	0.73 ± 0.16	0.34 ± 0.22	0.47 ± 0.04

Figure 6: **Left:** GRU regime predictions for an example pair (ZC/ZW); the shaded bands mark the oracle's cointegrated periods and the dashed line the train/test split. The network's probability $\hat{p}_t$ shows the probability of being in a cointegration area as predicted by the GRU model. **Right:** agreement between the GRU gate ($\hat{p}_t > 0.5$) and the oracle labels over the test period, as mean $\pm$ standard deviation across five seeds. AUROC is the probability the gate scores a cointegrated day above a non-cointegrated one (0.5 = chance); accuracy is the fraction of days labelled correctly; recall is the fraction of truly cointegrated days the gate flags; precision is the fraction of flagged days that are truly cointegrated; overlap (IoU) is the intersection over union of the flagged and true cointegrated periods.

that once we enter a period of cointegration (as predicted by the GRU model), we fix the β that the Kalman filter predicts at the beginning of the window, and keep this β until we exit the cointegration window. This is because we experimentally found that the Kalman filter β at the start of the window was a good enough approximation, and constantly changing the β led to worse results due to higher transaction costs and smaller divergences in the spread (See the table in Appendix B).

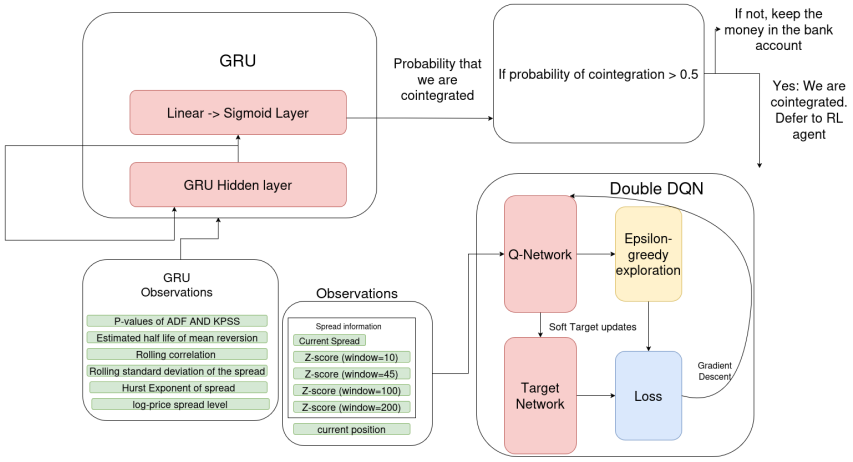


Figure 7: Overview of the architecture of the cointegration aware RL strategy

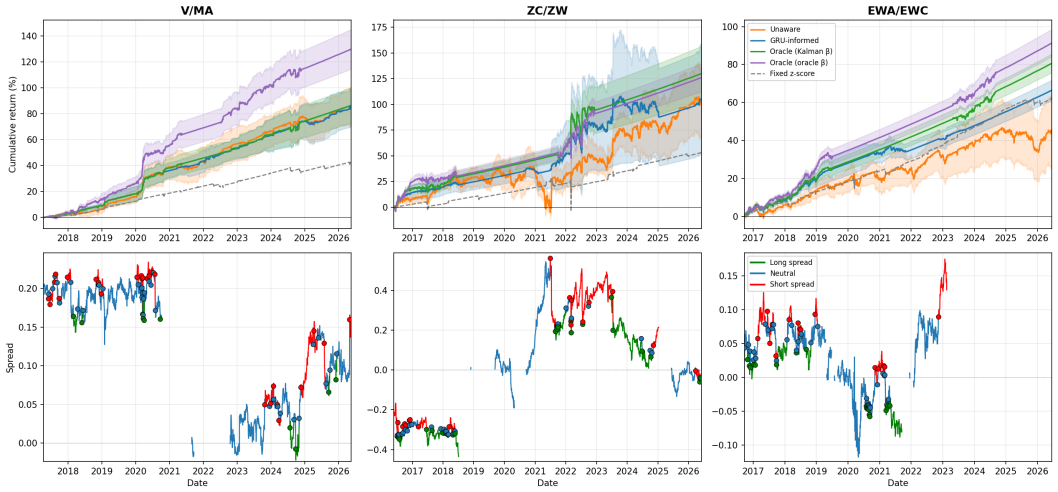


Figure 8: Wealth progression comparison of all methods and actions visualization. Each column corresponds to one pair. The top row corresponds to wealth progression for each strategy, with shaded regions showing std deviation. The bottom row shows the actions the GRU informed RL (our main method) does on some arbitrary seed. The colors green/blue/red correspond to being long/neutral/short the spread. The colored dots correspond to the RL agent making trades.

5 Experiments and Results

The five methods compared are:

- **Z-score baseline:** OLS-fitted β and thresholds over the train period, traded over the full test period.
- **Cointegration Unaware RL:** RL agent trading over the full test period.
- **GRU-Informed RL:** RL agent restricted to GRU-predicted cointegration periods, using Kalman Filter β.
- **Oracle-Informed RL (Kalman β):** As above, but cointegration periods determined by the oracle.
- **Oracle-Informed RL (Oracle β):** As above, but also using oracle-determined β (See Section 4.3).

For evaluation, we selected pairs from the same sector with similar profiles, as these are likely to exhibit periods of cointegration. Specifically, we use **Visa/Mastercard (V/MA)**, a standard equity pair in the literature; **Corn/Wheat futures (ZC/ZW)**, which are subject to similar weather and supply conditions; and **Australia/Canada ETFs (EWA/EWC)**, whose economies share similar industrial profiles (mining, etc.).

All experiments were run on 5 different seeds, with this informing the standard deviation figures.

We see in Table 1 that the Oracle informed RL performed very well, consistently outperforming the risk free rate by $3-5\%$. The Kalman β Oracle was slightly worse but still

Table 1: Comparison of the five methods, mean ± std over 5 seeds. Ann. (%) refers to annualized returns. Sharpe ratio is the average return minus the risk free rate, normalized by its std deviation. The Active-only column computes these statistics only during periods where the RL is allowed to trade in the first place (areas of cointegration), so it is left empty for methods that are not cointegration aware. The risk-free rate is 5%.

		All-days		Active-only (cointegrated)	
Pair	Method	Ann. (%)	Sharpe	Ann. (%)	Sharpe
V/MA	Fixed z-score (non-RL)	3.98	-0.55	–	–
	Cointegration-unaware	6.90 ± 0.94	0.47 ± 0.22	–	–
	GRU-informed	6.91 ± 0.93	0.59 ± 0.29	7.33 ± 1.11	0.65 ± 0.32
	Oracle (Kalman β)	7.08 ± 0.86	0.73 ± 0.24	7.93 ± 1.22	0.87 ± 0.28
	Oracle (oracle β)	9.59 ± 0.80	1.07 ± 0.19	11.47 ± 1.14	1.27 ± 0.22
ZC/ZW	Fixed z-score (non-RL)	4.36	-0.05	–	–
	Cointegration-unaware	7.30 ± 1.74	0.21 ± 0.09	–	–
	GRU-informed	7.16 ± 2.96	0.24 ± 0.33	9.23 ± 9.96	0.34 ± 0.66
	Oracle (Kalman β)	8.64 ± 1.32	0.50 ± 0.18	16.70 ± 4.33	0.88 ± 0.31
	Oracle (oracle β)	8.50 ± 0.67	0.47 ± 0.09	16.18 ± 2.19	0.82 ± 0.17
EWA/EWC	Fixed z-score (non-RL)	5.09	0.05	–	–
	Cointegration-unaware	3.86 ± 1.40	-0.16 ± 0.23	–	–
	GRU-informed	5.38 ± 0.35	0.17 ± 0.14	6.23 ± 1.22	0.30 ± 0.28
	Oracle (Kalman β)	6.28 ± 0.30	0.47 ± 0.10	8.00 ± 0.70	0.72 ± 0.15
	Oracle (oracle β)	6.90 ± 0.42	0.65 ± 0.16	9.49 ± 1.00	0.99 ± 0.24

clearly second best. The GRU informed RL (our main method) outperformed cointegration-unaware RL significantly on EWA/EWC, was similar in V/MA, and had significantly more variance in ZC/ZW, slightly under-performing on average. Looking at annualized returns in the "Active-only" column, we see that GRU-informed does in fact outperform the risk free rate significantly with a mean of 9%, but with a lot of variance ±9.96, which is not uncommon for higher returns. This may still make the cointegration aware RL preferable, as it requires the capital to be used for active trading far less frequently.

Figure 8 visualizes the wealth progression and actions of our main method (GRU informed RL). We see that our strategy is indeed learning a reasonable pairs trading strategy, where its consistently shorting the spread (red areas) when it is too high and buying it when it is too low (green areas), meaning it has learned to profitably exploit its mean reversion.

6 Discussion

We now revisit the three research questions posed in Section 1 in light of our results.

RQ1: RL versus fixed thresholds. Across all three pairs, the cointegration aware RL agents outperformed the fixed z-score baseline on both annualized return and Sharpe ratio, even after accounting for the 0.1% transaction cost. The fixed-threshold baseline achieved negative Sharpe ratios on two of three pairs (V/MA and ZC/ZW), whereas even the cointegration-unaware RL agent reached positive Sharpe ratios on those pairs. This

indicates that learning when to enter and exit, rather than relying on static thresholds, yields a more robust strategy. On EWA/EWC, the cointegration-unaware RL actually under-performs the fixed z-score baseline, which shows the value of our method.

RQ2: Offline cointegration detection and the oracle. Restricting the RL agent to oracle-identified cointegration periods improved performance on every pair. The improvement was most pronounced on EWA/EWC, where annualized returns rose from 3.86% (cointegration-unaware) to 6.90% (oracle β), and on ZC/ZW, where the active-only return during cointegrated periods reached 16.70%. This confirms our central hypothesis: pairs trading underperforms largely because of trading through structural breaks, and simply identifying and avoiding those breaks recovers most of the lost return—without any knowledge of future prices. The oracle thus serves as a meaningful upper bound on what cointegration-awareness can offer.

RQ3: Online approximation via the GRU. The GRU-informed agent, which detects cointegration causally using only past data, partially closes the gap to the oracle. It outperformed the cointegration-unaware RL baseline on EWA/EWC, matched it on V/MA, and showed higher variance on ZC/ZW. During its active (predicted-cointegrated) periods it beat the risk-free rate, with a mean active-only return of roughly 9%, while requiring capital to be deployed far less frequently than the always-on baseline. However, it did not reach the oracle's performance, as expected. The main obstacle is the instability of the GRU detector itself: its agreement with the oracle ranged from well below chance to strong depending on the random seed (Section 4.4). This makes the online strategy viable in principle but unreliable in its current form, and points to the cointegration detector—rather than the RL agent—as the bottleneck for live deployment.

7 Responsible Research

Our research is ethical as it involves no human subjects, or personally identifiable data. All price data used in this study is publicly available historical market data sourced from financial data providers [1, 24], and no proprietary or non-public information was accessed. The work is purely academic in nature and does not constitute financial advice or a live trading system. While algorithmic trading strategies in general carry potential for market impact, the scope of this research is confined to backtesting on historical data, and no capital was deployed. The study poses no risk of harm to individuals, communities, or the broader financial system.

The strategy of Pairs Trading, as explored, is ethical and responsible as it does not involve market manipulation or any other unethical method of gaining profit. In general, arbitrage helps markets be more efficient by reducing discrepancies by enforcing the "Law of One Price" [12].

We ensured our results are reproducible by running all experiments across 5 independent seeds (chosen seeds were 0,1,2,3,4) and reporting the mean and standard deviation of the aggregated results. We ensured that our methods did not train on the "test" half of the data, and we did not choose the seeds or hyper parameters in order to optimize performance on the "test" half. Our code is public and freely accessible [1].

[1] https://github.com/tudor-pagu/rp

Statement on AI Usage. AI was used to assist with the implementation and research aspect of this project. Code generated by AI was reviewed by me. The ideas and general direction of the project were not inspired by AI, but by my own intuition and research of prior work. AI was not used to directly create any figure or table, but only to assist with generating code that created the figures or tables directly from the real results of the experiments. AI was also used for proofreading and adjusting certain phrases in the paper, but it was not used to generate large passages from scratch or to determine the ideas of any paragraph. AI was used to refine and increase readability of passages that were originally written by me. In particular, AI was used to help typeset equations and determine readable mathematical notation.

8 Conclusions and Future Work

In conclusion, we trained a cointegration-aware Reinforcement Learning model, using Double DQN, to profitably run a pairs trading strategy on unseen real data, while only training it on simulated data. The cointegration awareness is provided by a GRU model trained on real data, labeled using the KPSS and ADF statistical tests for cointegration, with the model acting as a gatekeeper which only allows the RL agent to trade during periods of cointegration.

Our key finding is that selectively running the RL agent only during periods of cointegration significantly increased results (i.e. 3.86% $\rightarrow$ 6.9% Ann. returns EWA/EWC), and that this effect can be partially reproduced causally (without future knowledge) by using a GRU estimator. Our main limitation is that we found our GRU model to have a high variance and be highly sensitive to initialization. One likely cause is the limited amount of training data. More research is needed in order to consistently identify periods of cointegration in real time.

Our work shows the potential of long-term cointegration estimation on RL pairs trading. To our knowledge, no previous work showed the impact of cointegration structural breaks over multi-year trading horizons on a reinforcement learning pairs trading strategy. Future work should explore alternative methods for labeling cointegrated windows, while prioritising more robust and consistent models for real-time cointegration detection.

References

[1] Ran Arroyo and contributors. yfinance: Download market data from yahoo! finance's api. `https://github.com/ranaroussi/yfinance`, 2025. Accessed: 2025-06-02.

[2] Sebastian Becker, Patrick Cheridito, and Arnulf Jentzen. Deep optimal stopping. *Journal of Machine Learning Research*, 20(74):1–25, 2019.

[3] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[4] Hans Buehler, Lukas Gonon, Josef Teichmann, and Ben Wood. Deep hedging. *Quantitative Finance*, 19(8):1271–1291, 2019.

[5] Ernest P. Chan. *Algorithmic Trading: Winning Strategies and Their Rationale*. Wiley, 2013.

[6] Mei Choi Chiu and Hoi Ying Wong. Robust dynamic pairs trading with cointegration. *Operations Research Letters*, 46(2):225–232, 2018.

[7] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics, 2014. doi: 10.3115/v1/D14-1179.

[8] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[9] D. Dickey and Wayne Fuller. Distribution of the estimators for autoregressive time series with a unit root. *JASA. Journal of the American Statistical Association*, 74, 06 1979. doi: 10.2307/2286348.

[10] Robert J. Elliott, John van der Hoek, and William P. Malcolm. Pairs trading. *Quantitative Finance*, 5(3):271–276, 2005.

[11] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[12] Evan Gatev, William N Goetzmann, and K Geert Rouwenhorst. Pairs trading: Performance of a relative-value arbitrage rule. *The Review of Financial Studies*, 19(3): 797–827, 2006.

[13] Weiguang Han, Jimin Huang, Qianqian Xie, Boyi Zhang, Yanzhao Lai, and Min Peng. Mastering pair trading with risk-aware recurrent reinforcement learning. Papers 2304.00364, arXiv.org, Apr 2023. URL `https://ideas.repec.org/p/arx/papers/2304.00364.html`.

[14] Sepp Hochreiter and JÃŒrgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 11 1997. doi: 10.1162/neco.1997.9.8.1735.

[15] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960. doi: 10.1115/1.3662552.

[16] Taewook Kim and Ha Young Kim. Optimizing the pairs-trading strategy using deep reinforcement learning with trading and stop-loss boundaries. *Complexity*, 2019(1):3582516, 2019. doi: https://doi.org/10.1155/2019/3582516. URL `https://onlinelibrary.wiley.com/doi/abs/10.1155/2019/3582516`.

[17] Denis Kwiatkowski, Peter C.B. Phillips, Peter Schmidt, and Yongcheol Shin. Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? *Journal of Econometrics*, 54(1):159–178, 1992. ISSN 0304-4076. doi: https://doi.org/10.1016/0304-4076(92)90104-Y. URL `https://www.sciencedirect.com/science/article/pii/030440769290104Y`.

[18] Jing-You Lu, Hsu-Chao Lai, Wen-Yueh Shih, Yi-Feng Chen, Shen-Hang Huang, Hao-Han Chang, Jun-Zhe Wang, Jiun-Long Huang, and Tian-Shyr Dai. Structural break-aware pairs trading strategy using deep reinforcement learning. *The Journal of Supercomputing*, 78(3):3843–3882, 2022. doi: 10.1007/s11227-021-04013-x.

[19] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518 (7540):529–533, 2015.

[20] Boming Ning and Kiseop Lee. Advanced statistical arbitrage with reinforcement learning. *International Journal of Financial Engineering*, 12(04):2550019, 2025. doi: 10.1142/S2424786325500197. URL `https://doi.org/10.1142/S2424786325500197`.

[21] Wee Ling Tan, Stephen Roberts, and Stefan Zohren. Deep learning for options trading: An end-to-end approach. *arXiv preprint arXiv:2407.21791*, 2024. URL `https://arxiv.org/abs/2407.21791`.

[22] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1), 2016. doi: 10.1609/aaai.v30i1.10295. URL `https://doi.org/10.1609/aaai.v30i1.10295`.

[23] Ganapathy Vidyamurthy. *Pairs Trading: Quantitative Methods and Analysis*. John Wiley & Sons, Hoboken, NJ, 2004.

[24] Yahoo Finance. Historical market data. `https://finance.yahoo.com`, 2025. Retrieved via the `yfinance` Python library. Accessed: 2025-06-02.

[25] Hongyang Yang, Xiao-Yang Liu, and Qingwei Wu. A practical machine learning approach for dynamic stock recommendation. In *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications / 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 1693–1697. IEEE, 2018. doi: 10.1109/TrustCom/BigDataSE.2018.00253.

[26] Hongyang Yang, Xiao-Yang Liu, Shan Zhong, and Anwar Walid. Deep reinforcement learning for automated stock trading: An ensemble strategy. In *Proceedings of the First ACM International Conference on AI in Finance*, pages 1–8, 2020.

A Real vs Synthetic Data Training

This section justifies our use of synthetic training data over real training to prevent overfitting, as presented in Section 4.2. Table 2 shows that the agent trained on synthetic data was consistently better on the "test" region, as it did not overfit on the specific patterns of the limited "train" data. We test this on a separate set of pairs to confirm the effect is not specific to our main evaluation set.

Table 2: Training the agent on real historical spreads versus on synthetic spreads fitted to the oracle's cointegration windows. Real training attains high in-sample (train) returns but overfits and loses money out of sample; synthetic training generalises to the test period. Mean ± std over three seeds. We evaluated using the DDQN agent ran on the oracle determined cointegration windows. (As explained in Section 4.3)

Pair	Train data	Train ann. (%)	Test ann. (%)	Test Sharpe
PSX/MPC	real	26.23	-2.75 ± 0.76	-0.23 ± 0.08
	synth	11.81	15.68 ± 1.73	1.84 ± 0.20
HD/LOW	real	16.07	2.47 ± 2.77	0.41 ± 0.42
	synth	7.28	6.65 ± 0.75	1.68 ± 1.03
JPM/BAC	real	36.37	1.90 ± 1.06	0.45 ± 0.24
	synth	20.61	5.33 ± 0.50	1.53 ± 0.14
ZC/ZW	real	15.26	4.49 ± 1.38	0.37 ± 0.07
	synth	7.37	11.28 ± 0.79	0.91 ± 0.01

B Impact of fixing Kalman Beta on cointegration window start

This section justifies our decision in Section 4.5 to fix the kalman beta upon entering a cointegrated window and use the frozen value. As a clarification, note that the β is only fixed within any given cointegration window (it is frozen as the value of the Kalman filter β when the window first begins), but still varies between cointegration windows. Table 3 shows that fixing β in this way indeed reduces the number of trades made by the strategy and increases the profitability of the strategy.

Table 3: Effect of fixing the hedge ratio β once per cointegration window (snapshot) versus updating it daily with the Kalman filter, during the oracle cointegration periods. Across all three pairs, holding β fixed once we enter a cointegrated window gives higher risk-adjusted returns with far fewer trades; a daily-updating β forces a rebalance on every change, churning the position and eroding returns through transaction costs (on ZC/ZW it turns the excess Sharpe negative). Mean ± std over three seeds.

Pair	Hedge ratio β	Test ann. (%)	Test Sharpe	Trades
V/MA	Snapshot (fixed)	8.45 ± 1.00	0.90 ± 0.25	111
	Daily Kalman	7.96 ± 0.43	0.68 ± 0.08	911
ZC/ZW	Snapshot (fixed)	7.77 ± 1.31	0.38 ± 0.17	58
	Daily Kalman	2.69 ± 0.48	-0.31 ± 0.07	444
EWA/EWC	Snapshot (fixed)	6.23 ± 0.41	0.45 ± 0.16	103
	Daily Kalman	5.43 ± 0.13	0.16 ± 0.05	530