Enhancing Financial Algorithms for Pairs Trading using Reinforcement Learning

Constrained Portfolio Optimization

Cristian Petre-Luca[1]

Supervisor(s): Fenghui Yu[1]**, Frans 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: Cristian Petre-Luca
Final project course: CSE3000 Research Project
Thesis committee: Fenghui Yu, Frans Oliehoek, Neil-Yorke Smith

Abstract

Pairs trading exploits the mean reversion of a cointegrated spread of two stocks, classically traded with fixed z-score rules. We recast it as a continuous *portfolio-optimisation* problem and train reinforcement-learning (PPO) agents to size the two legs and a risk-free asset, comparing a *constrained* agent forced into the market-neutral hedge against a *free* agent that weights the legs independently. Agents are trained on a generative market model calibrated to each real pair, and evaluated out of sample, with and without transaction costs, against the classical z-rule. The constrained agent learns the spread's direction but does not beat the z-rule: it over-trades when no arbitrage is available instead of stepping aside. The free agent earns higher but far more variable returns, mixing directional market exposure with some genuine arbitrage. Transaction costs push both toward smoother, more conservative policies. We outline a potential improvement to the constrained agent to leverage its sizing capabilities in a future work.

1 Introduction

Over the past decade, finance has shifted markedly from discretionary to systematic trading: institutions increasingly automate strategies with statistical and machine-learning models. A prominent example is **pairs trading**, which exploits the relationship between two related stocks. It is attractive to financial institutions because it is **market-neutral**: by going long one stock and short the other, it reduces risk from broad market moves by profiting (solely) from asset relative mispricings. Equities in the same sector (energy, tech, finance) often share fundamental and statistical co-movement, which pairs trading turns into a low-risk, diversifiable source of return.

Of the different studied pair relationships in the literature, We focus on **cointegration** [5]: when a linear combination of the two asset prices, their **spread**, is mean-reverting, oscillating around a stable value even though each individual price wanders. **Statistical arbitrage** exploits this by trading the spread itself: short-selling it when it is unusually high and long-buying it when unusually low, betting that it reverts to its mean.

Classically these trades are triggered by fixed rules on the spread's z-score, but such rules have lost part of their edge as markets have grown more efficient [3]. They are also brittle to **regime changes** [1]: episodic shifts in market dynamics during which a pair's cointegration breaks down and the strategy stops paying. Modern approaches therefore seek models that adapt to changing conditions and size positions dynamically rather than relying on static thresholds. Reinforcement learning techniques, which treat markets as environments to which agents adapt and optimize returns, are a natural approach to the problem.

1.1 Relevant Literature

The first applications of RL specifically to pairs trading are only quite recent. Fallahpour et al. [6] were among the first, using Q-learning to *choose* entry/exit and stop-loss parameters rather than fixing them by hand. Kim and Kim [10] extended this with deep Q-networks and an early-exit action and accounted for transaction costs when computing returns. Both, however, restrict the agent to a *discrete* set of thresholds yielding rigid policies and neither places transaction costs *inside* the reward that shapes the policy. A further recurring limitation is the neglect of regime breakdowns: Lu, Lai, Shih et al. [11] tackle this directly, training sophisticated models to detect when cointegration holds and gating a threshold strategy on their predictions, but detection accuracy stays below 50%, underscoring the difficulty of the problem.

A parallel line of work frames trading as **continuous portfolio optimisation**: learning real-valued asset weights rather than timing entries and exits. Jiang et al. [8] learn long-only continuous portfolio weights in crypto markets via policy gradient methods, with transaction costs built into the objective. Yang et al. [16] specifically show that actor–critic methods such as PPO (proximal policy optimization) are effective for position sizing , including sensitivity to transaction costs. These methods size positions (almost) continuously but they do not approach the problem from pairs trading or market neutrality, instead profiting mainly from market trends. Closest to our setting, Wu et al. [15] train separate RL models to long and short stocks and combine them into a dollar-neutral book. This is an **equity-market-neutral** approach, but it selects relative-strength baskets rather than exploiting the cointegration relationship of a specific pair.

1.2 Contribution

We bridge these two lines of work: the pairs-trading literature that treats trading as an entry/exit timing problem, and the portfolio-optimisation literature that learns continuous position sizes based on signal strength. In Section 2 we define a reinforcement-learning setting that explicitly enforces the pairs-trading structure through **constrained actions**, which distinguish themselves from regular pairs trading by enabling **position sizing**, and we compare it against an ablation that removes the market-neutrality constraint (a free allocation). Our goal is to examine (i) what performance and behavioural differences the market-neutral constraint *produces*, for both cointegrated and non-cointegrated pairs; (ii) whether the constrained agent can outperform a classical z-score entry/exit rule for pairs trading thanks to its sizing features; and (iii) how transaction-cost-informed rewards affect the learned policies, the returns and risk taken to trade by the agents.

We leverage PPO as the policy-optimisation method for the portfolio-sizing problem. We also build a market simulator rather than relying on historical data alone, as opposed to the works discussed [10][6][16]: we deliberately expose the agents to synthetic episodes of mean reversion punctuated by volatility clustering and cointegration breakdowns, similar to those discussed by [1][11]. These conditions are qualitative and deliberately adversarial, but they give the agent a more

honest exposure to the kind of unseen, shifting market conditions it would face in deployment.

In Section 3, we find that the constrained agents learn the direction of the spread, but struggle to outperform the z-rule due to poor portfolio management when there are no arbitrage opportunities. The free agents have higher variance in trading outcomes, though it seems like they manage to learn both how to exploit market trends and some level of statistical arbitrage. Transaction costs lead to smoother policies and more conservative position adjustments. We discuss a hypothetical improvement to the constrained setting in Section 4.

2 A portfolio optimization framework for RL agents

In the following section, we present the exact formulations of the portfolio allocation settings of the reinforcement learning agents. We then proceed to describe the markov decision process (MDP) which describes our RL agents. We finish with a qualitative description of data generating process (DGP) we used to simulate the market.

2.1 Basic pairs trading and constraint defintion

We model the *spread* as a discrete-time, mean-reverting stochastic process (ignoring noise terms):

$$s_t = x_t - \beta\, y_t, \tag{1}$$

where x_t and y_t are the prices of the two cointegrated stocks at time t, and β is the cointegration (hedge) coefficient. We assume that we have knowledge of β, and, if it changes with time, its change is sufficiently slow so that the changes induced in the spread are locally marginal.

2.2 Capital allocation

In a pairs trading position, we always short one stock and long the other: we make a profit via s_t's variations Δs_t directly by holding stocks proportionally to their coefficients in the formula (1). Thus, if h_t gives the portion of the portfolio going into the first stock, we can define the weight vector

$$\mathbf{w}_t^c \equiv h_t\,[\,1,\,-\beta\,] = [\,h_t,\,-\beta h_t\,], \tag{2}$$

This means that we allocate a proportion of h_t of our capital to the first stock, and the second carries $-\beta h_t$. Negative weight values represent short-sells, and positive values represent long-buys. Note that β must be positive for mean reversion.

This allows to define two broad classes of portfolio optimization strategies which form a basis of comparison for this work: **a constrained allocation strategy** ($\mathbf{w}_t^c$), which always follows the pairs trading formula above, and whose choice of h_t determines portfolio sizing, and *an ablation*, a **free strategy**, which independently chooses the weights of the assets such as to maximize profits, i.e. a free weight selection strategy:

$$\mathbf{w}_t^f \equiv [\,w_{x,t},\,w_{y,t}\,] \tag{3}$$

Both legs block capital. For long buys, capital is directly converted into the asset, which we aim to sell at a higher price later. For short-sells, we borrow an asset with a contract to return it later at a lower price, pocketing the difference. Typical exchanges block the value inside the short as collateral (acting as "frozen" cash in the account), so we will apply the same consideration. The capital deployed in the pair is the *gross exposure* i.e., the sum of the absolute leg weights, or the ℓ_1 norm

$$\|\mathbf{w}_t\|_1^c = |h_t| + |\beta h_t| = |h_t|\,(1 + |\beta|). \tag{4}$$

$$\|\mathbf{w}_t\|_1^f = |w_{x,t}| + |w_{y,t}| \tag{5}$$

We do not enforce deploying the entirety of the capital in a trade (i.e., $\|w_t\|_1 \leq 1$). The remaining proportion can be placed in a **risk-free asset**, i.e., an asset whose return has no volatility (such as a savings account). The proportion is given by the residual $1 - \|w_t\|_1$ which translates to $1 - |h_t|(1 + |\beta|)$ or $1 - |w_{x,t}| + |w_{y,t}|$. We also force a **no borrowing requirement**, which reads as $\|w_t\|_1 \leq 1$, or

$$|h_t|\,(1 + |\beta|) \leq 1 \qquad \Longleftrightarrow \qquad |h_t| \leq \frac{1}{1 + |\beta|}. \tag{6}$$

and

$$|w_{x,t}| + |w_{y,t}| \leq 1 \tag{7}$$

respectively.

We assume every allocation w_t is feasible: there is sufficient liquidity (available orders to buy and sell stocks) to execute it, and our own trades do not move the market. These assumptions are mild here, given that the pairs we choose are large, widely traded equities, and the portfolios are small.

2.3 Reinforcement learning background

We assume that the task of the trading agent is to continuously adjust the weights of a portfolio according to the two allocation schemes we discussed to optimize returns. We represent this as a **discounted Markov decision process (MDP)**: a tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$: $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, $P(s' \mid s, a)$ the transition kernel (the probability of moving to s' after taking action a in state s), $r(s, a)$ the reward, and $\gamma \in (0, 1]$ a discount factor. The agent acts through a learnt *policy* $\pi_\theta(a \mid s)$, a distribution over actions given the state, parameterised by θ. Running the policy produces a trajectory of states/actions/rewards $\tau = (s_0, a_0, r_1, s_1, a_1, r_2, \dots)$, and the goal is to choose θ to maximise the expected **discounted** return

$$J(\theta) \;=\; {}_{\tau\sim\pi_\theta}\Big[\textstyle\sum_{t\geq 0}\gamma^t\, r_{t+1}\Big]. \tag{8}$$

The farther in time the returns, the more they are discounted, as later returns are increasingly influenced by market noise and should contribute less to the goal.

Learning is organised around two quantities. The *state-value* $V^\pi(s) =_\pi [\sum_{k\geq 0}\gamma^k r_{t+1+k} \mid s_t = s]$ is the expected return from s under π, averaged over all actions admissible by the policy, and the *action-value* $Q^\pi(s, a)$ is the same quantity when action a is taken first. Their difference, the *advantage* $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$, measures how much better taking a in state s is better than the policy's average behaviour there [13]. The advantage is used to determine which policy updates improve returns.

Policy optimization. Policy-optimization methods search directly over policies by ascending the return (8) in the parameters θ, rather than first learning action values and acting greedily with respect to them, as would certain variants of Q-Learning. The policy gradient theorem [14] gives an unbiased gradient that can be estimated from sampled trajectories,

$$\nabla_\theta J(\theta) \;=\; {}_{\pi_\theta}\big[\,\nabla_\theta \log \pi_\theta(a_t \mid s_t)\; A^\pi(s_t, a_t)\,\big], \qquad (9)$$

so each update simply makes advantageous actions more likely.

We use a gradient-based method since our control variable is continuous at each step, as the agent chooses real-valued portfolio weights and their sizing. Value-based methods such as Q-learning/DQN require a discrete action set or an explicit maximisation over actions at every step, which is unnatural and potentially suboptimal for the continuous problem.

PPO uses an on-policy actor–critic method in which a policy network (the actor) selects actions and a value network (the critic) estimates expected returns, the two trained jointly. Its defining principle is that each update is kept close to the policy that collected the data: by limiting how far the policy may move in a single step, one unlucky batch of noisy returns cannot collapse it, which also makes PPO notably robust to hyperparameter choice. These factors are valuable, given the low signal-to-noise and non-stationarity of market data. We used the standard library implementation Stable-Baselines3 (SB3).

2.4 The portfolio-sizing MDP

We now instantiate the state, action, and rewards of the agent. We don't explicitly model the transition function P, as PPO is model free, and we assume that the agents' trades (see 2.2) have no effect on the market, but only on the portfolio. Thus, no explicit transition representation is required.

State S. At step t the state splits into a *market* block and a *position* block,

$$\mathbf{s}_t \;=\; \big(\underbrace{z_t,\; \Delta z_t,\; \sigma_t}_{\text{market }\mathbf{m}_t},\;\; \underbrace{g_{t-1},\; \mathbf{w}_{t-1}}_{\text{position }\mathbf{p}_t}\big). \qquad (10)$$

The market block informs the reversion opportunity: z_t is the z-score of the spread (1) over a causal rolling window of length W (how far the spread sits from its mean), $\Delta z_t = z_t - z_{t-1}$ its one-step change (which way, and how fast, it is moving), and σ_t the rolling spread volatility (how much risk a position carries). When training our agent, we select W such that it encompasses a few (average) mean-reversion episodes of the spread.

The position block records what the agent is already holding as it enters step t: the weight vector $\mathbf{w}_{t-1}$ and its gross exposure $g_{t-1} = \|\mathbf{w}_{t-1}\|_1$ (5). We feed the current holdings back to the agent for two reasons: they tell it which way it is currently leaning (long or short the spread), and they are the reference point against which portfolio adjustments are measured, which allows the agent to reason about transaction costs.

Action and translator. At each step the agent selects portfolio weights $\mathbf{w}_t$ according to either the **constrained** or **free** allocation formula of Section 2.2. PPO's actor emits raw real numbers in $[-1, 1]$, and a translator maps them to an admissible weight vector. The uninvested residual $1 - \|\mathbf{w}_t\|_1$ is placed in the risk-free asset.

- **Constrained.** The actor emits a *single* scalar $a \in [-1, 1]$, mapped to the hedged position $\mathbf{w}_t = h_t\,[1, -\beta]$ with $h_t = a/(1+|\beta|)$, so the gross exposure is $\|\mathbf{w}_t\|_1 = |a| \in [0, 1]$. The sign of a sets the trade direction (long or short the spread) and $|a|$ the fraction of capital deployed. The translator builds the hedge from β, fixed at its real-world value.
- **Free.** The actor emits three values $[d_1, d_2, m] \in [-1, 1]^3$. The first two set the (sign-free) direction $\mathbf{d} = (d_1, d_2)/\|(d_1, d_2)\|_1$ (so $\|\mathbf{d}\|_1 = 1$), and m sets the magnitude $\rho = (m + 1)/2 \in [0, 1]$; the weights are $\mathbf{w}_t = \rho\,\mathbf{d}$, giving $\|\mathbf{w}_t\|_1 = \rho \in [0, 1]$. The two legs are chosen independently, with a separate dimension controlling how much to deploy.

Reward. Given that v_t is the portfolio value, x_t and y_t the values of the assets (in currency/unit), and $\mathbf{w}_t = [w_{x,t}, w_{y,t}]$ the weights chosen at instant t, and r_f the risk-free return per unit time, per unit currency invested, the net portfolio change at the next time step will be:

$$\Delta v_{t+1} \;=\; v_t \left(w_x \,\frac{\Delta x_{t+1}}{x_t} + w_y \,\frac{\Delta y_{t+1}}{y_t} + \big(1 - |w_x| - |w_y|\big)\, r_f \right), \qquad (11)$$

Let $R_{t+1} = \Delta v_{t+1}/v_t$ be the realised one-step portfolio return, represented in percentile change: what the portfolio was multiplied by from one day to the next. Trading on exchanges can incur percentile transaction costs, so we charge a proportional turnover cost and define the *net* one-step return

$$R^{\text{net}}_{t+1} \;=\; R_{t+1} \;-\; \frac{c}{10^4}\,\|\mathbf{w}_t - \mathbf{w}_{t-1}\|_1\,, \qquad (12)$$

where c is the cost in basis points per unit of **turnover** or change in stock investments, equivalent to $||\Delta \mathbf{w}||$. ($c = 0$ in without transaction costs). We selected the **log return** (also used by [10]) function as a reward

$$r_t \;=\; \log\big(R^{\text{net}}_{t+1} + 1\big), \qquad (13)$$

This function learns risk management as portfolio losses are more heavily punished (wipeouts, where $R^{net}_{t+1} = -100\%$, lead to $r_t = -\infty$) than gains are rewarded. Since we want to avoid wipeouts, this is a natural choice specifically when optimizing portfolios.

2.5 Training data generation

Training an RL agent policy typically requires more data than is available using an asset's daily history of closing prices. A real pair gives us a single realisation of the market, which is impossible to repeat under controlled conditions. We therefore train on *synthetic* data produced by a simulator (a data-generating process, or DGP) that we can sample from for a number of steps, drawing fresh, independent episodes for every training run. By assuming cointegrated pairs, we encode the synthetic spread as mean-reverting, so we can study how the agent behaves when a genuine statistical-arbitrage opportunity exists.

We calibrate the simulator to *each real pair* we study, so the synthetic data inherits that pair's characteristics. The parameters describe how the spread reverts and how risky it is. We model the spread as a first-order autoregression (AR(1) process) $s_t = \theta(1-\phi_t) + \phi_t s_{t-1} + e_t$, fit by ordinary least squares. θ is the mean s reverts to, ϕ dictates the speed of reversion, and yields the *half-life* of mean reversion ($\ln 2/(-\ln \phi)$), and the *spread volatility* (the residual standard deviation, how much noise the agent must trade through), while the *hedge ratio* β that defines the spread is estimated separately. We also match the heaviness of the return tails. The innovations ε_t that drive the spread and the price process below are drawn from a variance-normalised Student-t distribution, whose degrees of freedom ν control how often large, abrupt moves occur. We measure the empirical *excess kurtosis* of the real returns and set ν so the simulated shocks are as fat-tailed as the real ones.

The pair is then built around the spread. One leg is modelled as a random walk with drift, which gives the common stochastic trend the two stocks share, and the other is derived based on the spread formula we defined and the other leg's value.

We include occasional artificial *regime switches*: stretches where the spread becomes more volatile and reverts weakly or not at all implemented as a two-state (calm/turbulent) Markov chain: a day is labelled turbulent when its rolling volatility exceeds a high percentile of the window, and the transition probabilities (how often a turbulent regime begins and how long it typically lasts) are read directly from the lengths of these labelled stretches. The full calibration procedure and parameter values are given in Appendix C.

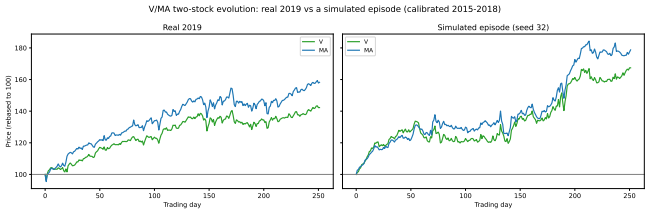

Figure 1: Real versus simulated price evolution for the V/MA pair. *Left:* the two stocks over 2019, each rebased to 100 at the start of the window. *Right:* a representative episode drawn from the DGP calibrated on 2015–2018, rebased identically. The simulated legs (qualitatively) reproduce the co-movement (cointegration) of the real pair without sharing its specific realisation.

3 Results and discussion

The goal of this protocol is to assess the performance of agents with free and constrained weight selections discussed in Section 2, against traditional pairs trading benchmarks: a simple market position where we long both stocks equally, a constant risk-free investment giving us 5% compound interest (similar to most banks) and a simple **z-rule** that computes the rolling z-score of the spread, which it then shorts/longs at $z = \pm 2$. It exits the trade at $z = \pm 0.5$, which is the profitable move as the spread has reverted towards the mean at $z = 0$, and uses a stop loss at $z = \pm 3$. Outside of trades, the z-rule defaults to the risk-free investment.

Note that the z-rule is similar to the **constrained allocation strategy** while in-position: it chooses the same kind of weight vector $\mathbf{w}_t^c$, but $|h_t| = \frac{1}{1+|\beta|}$ such that the gross exposure is equal to 1. This is the typical benchmark used in the literature on pairs trading without portfolio adjustments.

In this section, we describe the procedure used to select both cointegrated and non-cointegrated pairs. We then compare the performances of the models after training them on synthetic data generated based on the 2016–2019 trading window (in-sample data), and test their performance on the 2020–2023 period (out of sample). We train agents across the same parameters (see App. C). The agents trade on the **daily closing prices** of those periods. To evaluate risk-adjusted returns, we use the annualized returns, and the **annualised Sharpe ratio** [12]: the ratio of a strategy's average excess return to its volatility, scaled to a yearly horizon,

$$\mathrm{SR} = \frac{\overline{r} - r_f}{\hat{\sigma}_r}\sqrt{252}, \tag{14}$$

where $\overline{r}$ and $\hat{\sigma}_r$ are the sample mean and standard deviation of the daily returns r_t, r_f is the daily risk-free rate, and the factor $\sqrt{252}$ annualises the daily ratio (252 trading days per year). A higher value indicates more return earned per unit of risk taken.

3.1 Pairs, cointegration tests

The portfolio-optimisation framing requires a cointegrated pair, whose hedge ratio β we estimate with Johansen's test [9],a classical statistical technique based on a maximum likelihood estimator. We start from eleven economically linked U.S. equity pairs and apply a *filtering screen* on the in-sample window 2016–2019 (keeping pairs if $0.50 < \beta < 3.00$, and their mean-reversion half-life corresponds to 10–150 trading days), retaining eight. The hedge ratio is fixed at its 2016–2019 value across all evaluation windows.

We note that our experiments show that β drifts in time even in strongly cointegrated pairs, so recalibrating the training window and hedge ratio would provide a more robust training regimen.

We characterise tradeability with an *ADF test [2] on the formation-β spread* $s_t = \log a_t - \beta \log b_t$, the standard autoregressive test in the literature. Note that this is a different formulation of the spread from (1), but it leads to equivalent results. We run ADF on *long* windows the in-sample period (2016–2019), the full out-of-sample period (2020–2023), and an extended span to 2026, so that ADF captures long-term patterns in spread mean reversion. Table 1 reports the result.

We note the *regime-switching problem*: in-sample cointegration does not imply out-of-sample cointegration. Of the four pairs cointegrated in 2016–2019, one (HD/LOW) is no longer cointegrated in 2020–2023, while a pair cointegrated out-of-sample (KO/PEP) was not in-sample; over the longer span to 2026 even fewer persist. This corroborates the findings of Lu, Lai et al [11] previously discussed. The tradeable set therefore drifts across eras, and an agent calibrated

pair	β	HL (d)	ADF_{IS}	ADF_{OOS}	ADF_{long}
V/MA	0.77	16	**0.001**	**0.010**	0.496
MCD/YUM	1.01	39	**0.035**	**0.041**	**0.003**
HD/LOW	1.23	37	**0.039**	0.412	0.195
KO/PEP	0.94	52	0.088	**0.010**	0.933
PSX/MPC	0.68	112	0.330	0.212	**0.028**
GS/MS	0.61	72	0.141	0.377	0.964
LMT/NOC	0.99	42	0.059	0.302	0.121
JPM/BAC	0.87	75	0.235	0.930	0.944

Table 1: ADF p-values on the formation-β spread (bold = cointegrated at 5%), with the Johansen hedge ratio β and OU half-life. Cointegration is *non-stationary*: HD/LOW is cointegrated in-sample but not out-of-sample, KO/PEP the reverse, and most pairs are cointegrated in neither. Only V/MA and MCD/YUM hold in both.

on one regime is not guaranteed a tradeable spread in the next. We accordingly focus the head-to-head comparison on the two pairs that are cointegrated in both periods, **V/MA** and **MCD/YUM**.

3.2 Performance validation

Albeit exhibiting similar distribution statistics, the generative model inherently projects qualitative features that may not generalize, and thus, training on it will eventually lead to overfitting. The training budget in days generated is thus a hyperparameter, so we select it on data a validation set outside the data generator never sees. We calibrate the simulator on 2015–2018 and validate on the immediately following year, 2019, leaving 2020–2023 untouched. As training grows, we track performance both on 50 fresh synthetic episodes (the training distribution) with a different seed, and on the real 2019 validation data. As the agent optimizes log-return, we read the evolution of the performance of the *annualised return* to gain an understanding on the effect of training.

Figure 2 shows V/MA validation performance with and without transaction costs. On synthetic (training) data, performance gains for constrained agents are marginal (somewhat significant when dealing with transaction costs), and for free agents, there are significant gains in the first 250k training steps, which then plateau, with higher benefits when transaction costs are involved. These trends are roughly confirmed on the validation set, with higher associated variance yet also somewhat higher returns, likely due to the specific market conditions in 2019. It seems like constrained agents need little training for their policy to converge, whereas free agents extract more improvements with more exposure to data, yet their perfromance is also noisier, an artefact of a much larger action space (see 2.4).

It seems like both agents subtle improvements, likely related to turnover, with more training when transaction costs shape the reward, yet the effect on extracted edge (profit) differs: free agents seem to adjust better with more training, whereas the constrained agent plateaus very quickly. This comes with the caveat that variance is high out of sample, and that a proper analysis would look at the performance of each stock and window, as optimal training amount differs based on our experiments. For consistency, we used 250k training points across all tests due to computational limitations, but we note that an in-depth analysis would require per-pair evaluation.

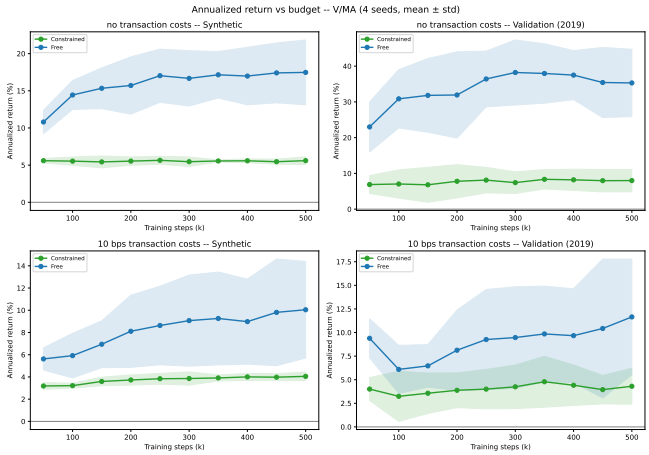

Figure 2: Annualised return versus training budget for V/MA, constrained (green) and free (blue), mean ± seed band across seeds (0,1,2,3) for training / seeds (70000, 70001, 70002, 70003) for synthetic testing. Left: fresh synthetic episodes; right: the 2019 validation data; top: no costs; bottom: net of 10 bps.

3.3 Out-of-sample performance without transaction costs

Figure 3 shows out-of-sample equity (2020–2023); the learned policies and realised weights are in Figures 4 and 5; Table 2 summarises.

pair	strategy	Sharpe	ann. ret	turn.	gross
V/MA	constrained	+0.32±0.20	+7%±1%	0.21±0.06	0.80±0.03
	free	+0.23±0.21	+11%±5%	0.15±0.04	0.76±0.12
	z-rule	+0.65	+8%	0.09	0.43
	market (hold)	+0.19	+11%	0.00	1.00
	risk-free	0.00	+5%	0.00	0.00
MCD/YUM	constrained	−0.15±0.07	+4%±1%	0.14±0.03	0.80±0.02
	free	+0.27±0.39	+9%±5%	0.37±0.19	0.61±0.07
	z-rule	+0.43	+8%	0.05	0.29
	market (hold)	+0.24	+11%	0.00	1.00
	risk-free	0.00	+5%	0.00	0.00

Table 2: Out-of-sample results (2020–2023), without transaction costs, on cointegrated pairs (V/MA, MCD/YUM). Continuous full-period annualised Sharpe and return, mean per-step turnover (ℓ_1 weight change), and mean gross exposure (± over seeds). The z-rule banks the risk-free rate while flat.

The constrained agent earns a small but genuinely market-neutral return on the cointegrated pairs (Sharpe +0.22 on V/MA, +0.11 on MCD/YUM) with a tight seed band. It succeeds at the central aim of pairs trading, suppressing risk, if only modestly. The free agent has higher overall returns, though also with a higher seed-induced variance across all metrics. Its allocations (Fig. 5) are directional bets, typically long both stocks, so it rides the market line rather than the spread.

This can be an artefact of a mild positive-weight exploration bias compounded by the general uptrend of the stocks themselves. On the stationary noisy pair, such as JPM/BAC,

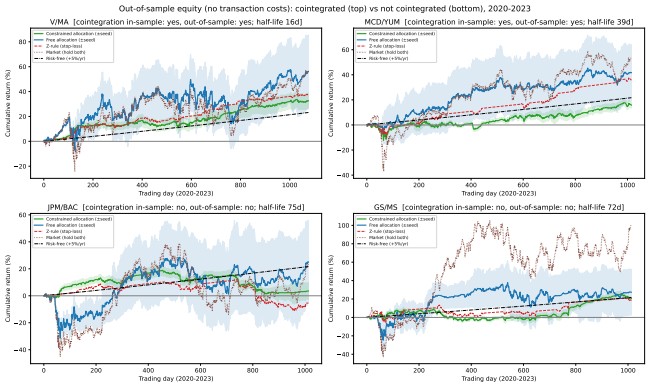

Figure 3: Out-of-sample equity curves (2020–2023), no transaction costs, for two cointegrated pairs (V/MA, MCD/YUM) and two non-cointegrated controls (JPM/BAC, GS/MS) for both agents settings and benchmarks.

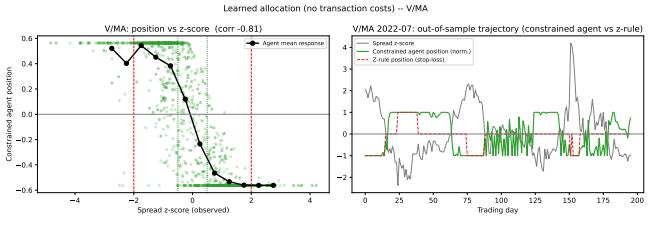

Figure 4: Learned policy of the constraind agent versus z-score without transaction costs.The constrained agent leans against the spread. It adjusts faster than the z-rule, but its policy is also jittery when the spread sits close to the mean, around $z = 0$.

its net returns fluctuate and remain more modest on average (albeit still very noisy). Interestingly, on V/MA, the weight allocation of the free agent appears to mix in real arbitrage (Fig. 5), as it moves the legs oppositely while staying net-long, showing that it may have begun learning a mixed strategy.

We observe that these two agent configurations exploit two different market factors: alpha, which represents the returns obtainable from spread mean reversion, and beta, which is given by the overall market direction. Constrained agents cannot exploit beta by design, and the available arbitrage is their performance ceiling. Free agents are, in principle, free to exploit both, yet the performance shows that they lean towards beta exploitation (as it likely results in higher returns), with some capability to absorb alpha as well.

Even without costs the banked z-rule *dominates* on a risk-adjusted basis ($+0.65$, $+0.42$). The gross/turnover values tell us that it stays in cash, well over half the time (gross 0.95–0.42), banking the risk-free rate, and carries spread risk only at strong spread dislocations, whereas both agents are almost always deployed (gross $\sim 0.7 - 0.8$) and trade far more (turnover 0.15–0.37 versus 0.05–0.09). The policy plot (Fig. 4) illustrates this: the constrained agent reacts to the spread slightly faster than the z-rule and has the right reversion direction, but at roughly constant size and seldom flat. It never learned **when not to trade**, and its hyper-

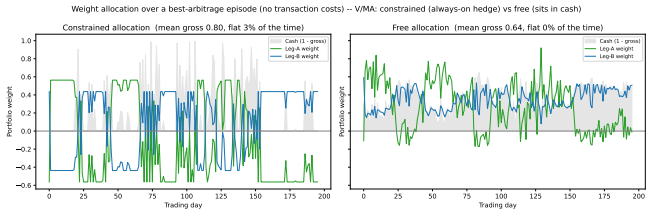

Figure 5: Per-leg weights and cash residual over one out-of-sample episode, constrained (left) versus free (right) without transaction costs. The constrained agent holds an always-on market-neutral hedge ($w_A \approx -\beta\, w_B$, gross ~ 0.8); the free agent tends to long both stocks.

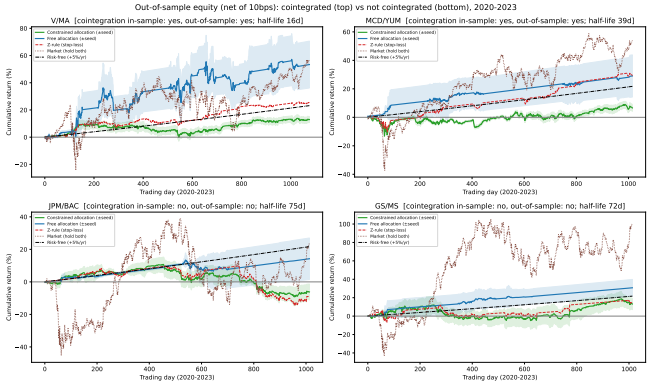

Figure 6: Out-of-sample equity curves (2020–2023), with transaction costs, for two cointegrated pairs (V/MA, MCD/YUM) and two non-cointegrated controls (JPM/BAC, GS/MS).

responsiveness to small spread moves manufactures turnover that, as the next section shows, is fatal under costs.

3.4 Out-of-sample performance with transaction costs

We repeat the evaluation net of 10 bps on per-step turnover, using agents trained with the matching penalty (Figs. 6, 7, 8; Table 3).

pair	strategy	Sharpe	ann. ret	turn.	gross
V/MA	constrained	-0.36 ± 0.08	$+3\%\pm0\%$	0.09 ± 0.01	0.83 ± 0.05
	free	$+0.36\pm0.13$	$+10\%\pm3\%$	0.05 ± 0.03	0.29 ± 0.16
	z-rule	$+0.11$	$+5\%$	0.09	0.43
	market (hold)	$+0.19$	$+11\%$	0.00	1.00
	risk-free	0.00	$+5\%$	0.00	0.00
MCD/YUM	constrained	-0.39 ± 0.06	$+2\%\pm1\%$	0.07 ± 0.00	0.84 ± 0.01
	free	$+0.19\pm0.47$	$+6\%\pm3\%$	0.02 ± 0.02	0.13 ± 0.10
	z-rule	$+0.24$	$+7\%$	0.05	0.29
	market (hold)	$+0.24$	$+11\%$	0.00	1.00
	risk-free	0.00	$+5\%$	0.00	0.00

Table 3: Out-of-sample results (2020–2023), with transaction costs, on cointegrated pairs (V/MA, MCD/YUM). Continuous full-period annualised Sharpe and return, mean per-step turnover, and mean gross exposure ($\pm$ over seeds).

Under costs the constrained agent's per-trade edge (or risk-adjusted returns) is further diminished by its turnover (Sharpe -0.36, -0.39), while the rarely-trading z-rule maintains an edge ($+0.11$, $+0.24$). Interestingly, the free agent manages to

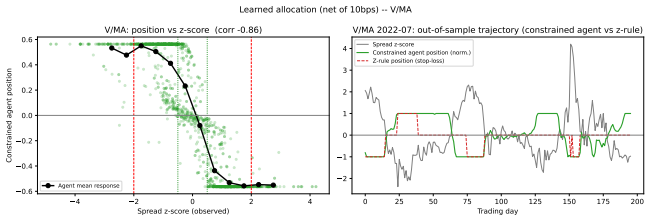

Figure 7: Learned policy of the constraind agent versus z-score with transaction costs. The positions the agent learns tend to be either fully deployed, or a closer to 0 deployment. The policy trades like a more active z-rule, with noisy adjustments when $z \sim 0$.

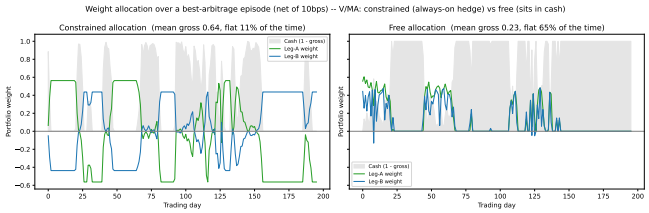

Figure 8: Per-leg weights and cash residual over one out-of-sample episode, constrained (left) versus free (right) with transaction costs. The free agent tends to sit out of position (gross ~ 0.23), whereas the constrained deploys a lot more capital (gross ~ 0.64)

outperform all benchmarks on V/MA with transaction costs when it comes to risk-adjusted returns (Sharpe 0.36 ± 0.13, always above all others), including *itself* when transactions costs aren't involved. The penalty reshapes both free and constrained policies: turnover roughly halves, and the allocations become smooth, but in opposite ways: the free agent learns to sit out, collapsing into cash (gross $0.14 \sim 0.29$), whereas the constrained agent does the reverse, becoming *more* like the z-rule and favouring full exposure over small bets (gross ~ 0.85, Fig. 7). The free agent's selective entries and exits are likely the strength behind its V/MA performance.

The intuition is that changing one's mind is costly, the constrained agent stops making small confidence adjustments to exposure and instead holds a large position through the reversion, losing reactiveness for a larger expected payoff.

The magnitude of portfolio adjustments matters: at the ~ 0.2 daily turnover of the frictionless policies, roughly 5% of capital per year would be lost to 10 bps fees. This is larger than the differences in annualized returns between the two settings, so transaction costs in the reward did their work: cost-aware training is essential for economic trades.

The constraint mechanism of arbitrage nonetheless survives costs. After training across all 8 pairs, the constrained agent's returns scales with realized arbitrage: regressing against the z-rule's returns, agent's return scales tightly (slope $+0.76$, $r = +0.65$, $p < 10^{-8}$). The market-neutral constraint keeps exploiting available arbitrage by predicting the correct spread direction, but it simply cannot time entries and exits well enough to beat a threshold rule.

3.5 Why the z-rule wins, and how to beat it

The z-rule does not optimise the portfolio at all. It enters a fixed-size directional spread trade, and yet it consistently beats the constrained agent, which could learn the z-policy in principle if it fixes the gross exposure to 1. The agent gets the direction of the spread right but makes *timing* mistakes. Cointegration breaks down periodically, and reversion opportunities are confined to specific regimes. The z-rule limits exposure to these events, as entries are limited to when $|z| > 2$, but it is vulnerable to the spread hovering around the entry signal and not hitting one of the exits. This allows us to hypothesize an improvement: a signal like the z-score decides when to enter a pairs trade, and an agent can focus solely on *how much to size* once in a trade.

4 Conclusion

4.1 Summary

In this work, we wished to examine the differences between free and constrained portfolio allocation settings based on pairs trading principles using RL agents. We noticed that constrained portfolio allocation agents learn the direction of the spread, but without proper exposure adjustment (i.e., knowing when to stay out of trades), they do not manage to outperform a simple z-rule which enters/exits trades with full exposure, thus failing to show that controlling position sizes brings benefits when coupled with statistical arbitrage.

Agents which freely allocate weights can learn both how to do statistical arbitrage and exploit market trends, but their performance is noisier and more sensitive to the training. Constrained agents that have a small action space have policies that are converge faster and are less sensitive to training differences. Free agents have a potentially higher skill ceiling and adaptability to various trading conditions, but the quality and quantity of data they are exposed to is crucial for extracting the a good policy from a bad one.

In all scenarios, introducing transaction costs in reward functions leads to lower turnover and smoother policies. Nonetheless, these policies are more robust for real-world applications, making it an important consideration when developing trading agents for real markets.

4.2 Further work

A first point of investigation would be to devise a simpler, more robust training protocol: either by adapting the market simulator, or by looking at intraday historical closes and taking a real-data-only approach. The goal of these experiments would be to reduce the variance of the training outcomes and bring the in-sample and out-of-sample performances closer. We note, however, that this problem is strenuous, and an effective market simulator could be the main focus of a research problem in its own right.

We additionally point out a further hypothesis: that the optimal returns of constrained portfolio optimization for pairs trading are best observed with a proper model for entries and exits. We suggest that refined pairs trading signals, such as those developed by [10], coupled with models that do not determine direction but only adjust position size, potentially including leverage, could lead to noticeable performance benefits in the case of cointegrated pairs trading.

5 Responsible Research

5.1 Experimental Limitations

To make a general claim of the performance of our models on real markets, further experimentation would be needed. We've tested our models only on two proper cointegrated pairs (along with some non-cointegrated ones), and only across four seeds. Based on the validation curves for V/MA alone, we see that seed-related variance is high even on training data and that performance did not yet begin declining at 500k steps for the free agent. This indicates that further agent tuning and validation would be needed for a clear cut result.

Nonetheless, the time and resource constraints of this project limited the amount of experimentation we could do. On the laptop we used to run the training pipelines (see Appendix), the average training run for 250k steps would take about 10 minutes: for each seed, transaction cost setting, asset pair, agent configuration.

Further, we make assumptions about the market as well as experiment design choices, which may fail to generalize to real data. We consider our data generation protocol for training as a kind of limitation rather than a contribution, as it inherently projects assumption such as regime types and market shocks which are, at best, an approximation.

Finally, an inherent limitation to any time-bound study on the market is that it changes with time. As pairs trading becomes more widely used, the opportunities for statistical arbitrage will become wane. These phenomena need to be regularly studied, as any pattern we see will ultimately depend on the trading period.

5.2 Differences with real-world deployment

In Section 2.4, we presented the assumptions that we have access to sufficient liquidity to fulfill all of our trades, and that the trades themselves do not influence the market. We also assigned fixed transaction costs (0 or 10bps) in our experiments, and we only considered the daily closing prices of stocks and daily trading.

In practice, available liquidity and transaction costs are heavily determined by exchanges and assets. The market has active order books, and these buy and sell orders determine the price and quantity at which an asset can be traded — closing daily prices are a snapshot of the full picture. In practice, fulfilling trades is heavily dependent on market interactions, and for highly volatile assets with low liquidity or intraday trading conditions, it is entirely possible that markets may move without an order being (completely) filled, in what is known as slippage. This failure mode introduces another kind of risk which our agents do not model.

We consider the goal of our study what approaches teach RL agents how to take profitable trading directions, reduce risk related to pricing uncertainties and examine the effects of transaction cost awareness. The insights gained are fundamental for general trading applications, but we acknowledge that there exists an implementation gap between our model usage to the deployment on an actual exchange.

5.3 Ethical implications

A significant limitation of our experiments was computational power itself. The more access to computational resources someone has, the more models they can train and the better they can exploit the market.

This means that the utility of studies in finance favor financial institutions who can invest in optimizing their strategies, even if the research is available for anyone to read. However, as our study focuses on daily trading, and we don't consider large liquidities, most of the computational horsepower and raw capital that trading firms use in their trading do not pose as much of an unfair advantage.

In all scenarios, trading contributes to a more fairly priced market, and ensures better access to asset liquidities for regular individuals as well. This makes financial research relevant for society as a whole.

A AI Usage Disclosure

Claude Opus 4.8 was used to support the writing and coding process of this research paper.

For the implementation part: the model was used to build the boilerplate (Gym environment, translators...) for the Reinforcement Learning pipeline. Additionally, AI was used to engineer the data generation process iteratively. Finally, AI was used to regenerate notebooks for the different experiments I designed. I personally checked all results that I presented in my work, as well as the code snippets used. All code can be found on the associated GitHub.

For the writing part: AI was used to proofread, rephrase and synthesize the text, as well as LaTeX syntax generation for tables. AI was also used to suggest relevant literature on various topics for further reading. All arguments and analysis were my own.

B Laptop specifications

Table 4: Hardware used for training and evaluation.

Component	Specification
Processor	Intel Core i7-1270P (12th Gen), 12 cores (4P + 8E), 16 threads, 2.20 GHz
RAM	16.0 GB (15.7 GB usable)
Graphics	Intel Iris Xe Graphics (128 MB)
System type	64-bit OS, x64-based processor

C PPO parameters

Table 5 lists the PPO configuration used for all learned agents. Network architecture and most hyperparameters were not tuned. They are standard PPO/Stable-Baselines3 defaults, appropriate for the low-dimensional observation of the state (5 features), but a few choices follow directly from the mean-reversion structure of the problem:

- **Episode length**, $\mathrm{clip}(12 \times \text{half-life}, 252, 756)$ steps: each episode spans roughly twelve mean-reversion half-lives, so the agent experiences many complete reversion cycles per episode. It is floored at one trading year (252) for the shortest-half-life pairs and capped at three years (756) for the longest, to keep episodes tractable. The selected pairs' half-lives span 16–112 days.

- **Discount factor** $\gamma = 0.99$: the effective horizon $1/(1-\gamma) \approx 100$ steps comfortably exceed a reversion cycle for almost all pairs, so the agent values holding a position through a full mean-reversion, while remaining short enough not to over-weight distant rewards where market noise dominates.
- **Total training steps** 2.5×10^5: a training-budget sweep located ~ 2–3×10^5 in the no-overfit region (a flat out-of-sample curve, see Section **??**); our claims are relative and mechanistic at a fixed budget, and per-pair budget calibration is left to future work.
- **Entropy coefficient** 0: the continuous action space and the stochasticity of the simulated paths provide sufficient exploration, so no entropy bonus was used, Based on our experimentation, the performance benefits from tuning this parameter were limited.
- **Observation/reward normalisation** (running `VecNormalize`): standardizes the small-magnitude per-step returns and state features , which acts as a stabilizer for PPO updates and ensures each feature has equal importance in training.

Table 5: PPO configuration used for all learned agents.

(Hyper)parameter	Value
Policy/value network	2-layer MLP, 32 units/layer
Activation function	Tanh
Optimizer	Adam
Learning rate	3×10^{-4}
Discount factor γ	0.99
GAE parameter λ	0.95
Clip range ϵ	0.2
Entropy coefficient	0
Minibatch size	64
Rollout per update	2048 env-steps (256×8)
Parallel environments	8
Episode length	clip($12 \times$ half-life, 252, 756) steps
Total training steps	2.5×10^5
Seeds	$0, 1, 2, 3$
Observation / reward scaling	normalised (`VecNormalize`)

D Data generation considerations

This appendix gives a formal description of the data-generating process (DGP) summarised in Section 2.5. The guiding principle is *calibrate-once, sample-many*: we fit the DGP a single time on a real training window for each pair, and thereafter draw a fresh, independent market on every episode reset. We justify this as follows. First, every component of the model corresponds to a qualitative feature of financial returns (mean reversion of the spread, volatility clustering, heavy tails, regime changes) and is included only to reproduce it. Second, no component is a free knob: each parameter is *estimated* from the real pair (Appendix D.4), and we verified that the resulting synthetic moments match the real ones (the model was validated against the realised PSX/MPC return moments.

D.1 The cointegrated price system

The pair is built from a stationary tradeable signal (the spread) and a common stochastic trend (one leg's price), so that the two log-prices are cointegrated by construction.

Spread (Ornstein–Uhlenbeck / AR(1)). As the basis of our spread construction, we use the theoretical framework of the Ornestein-Uhlenbeck process. The latent spread s_t follows a discrete-time mean-reverting process,

$$s_t \;=\; \theta\,(1-\phi_t) \;+\; \phi_t\, s_{t-1} \;+\; e_t, \qquad e_t = \sigma_s\, m_t\, \xi_t, \tag{15}$$

with long-run mean θ towards which it reverts, a persistence factor of the cointegration $\phi_t \in (0,1)$, and innovations e_t of base scale σ_s (the regime multiplier m_t and the standardised shock ξ_t are defined in (22)–(19)). The persistence fixes the mean-reversion *half-life*

$$H \;=\; \frac{\ln 2}{-\ln \phi}, \qquad \Longleftrightarrow \qquad \phi \;=\; 2^{-1/H}, \tag{16}$$

the number of steps over which a deviation from θ decays by half. This is the standard continuous-time OU model of a pairs spread discretised to daily steps [4].

Common trend (leg B). The second leg is a random walk with drift — the non-stationary factor shared by the pair,

$$r_t^b \;=\; \mu_b \;+\; \sigma_b\, m_t\, \xi_t^b, \qquad \log B_t \;=\; \log B_0 \;+\; \sum_{k \le t} r_k^b. \tag{17}$$

First leg and cointegration identity. Leg A is defined so that the two logs are cointegrated with hedge ratio β and stationary residual s_t,

$$\log A_t \;=\; \beta\, \log B_t \;+\; s_t. \tag{18}$$

Equation (18) is the Engle–Granger cointegration relation [5]. $\log A_t$ and $\log B_t$ are each $I(1)$, but the combination $\log A_t - \beta \log B_t = s_t$ is stationary, which is exactly the condition a pairs strategy exploits.

D.2 Heavy-tailed innovations

The shocks ξ_t, ξ_t^b in (15)–(17) are standardised Student-t variates,

$$\xi_t \;=\; Z_t \sqrt{\tfrac{\nu-2}{\nu}}, \qquad Z_t \sim t_\nu, \tag{19}$$

rescaled to unit variance so that σ_s, σ_b retain their meaning as standard deviations. A finite degrees-of-freedom ν produces the excess kurtosis (fat tails) that is a universal stylised fact of asset returns. ν is calibrated (Appendix D.4) so the heavy tails are inherited from the real pair.

D.3 Volatility-clustering / reversion-breakdown regime

Market turbulence arrives in clusters, so we modulate both legs with a simplified two-state Markov regime $g_t \in \{\mathsf{calm}, \mathsf{turbulent}\}$ with transition probabilities

$$\Pr(\mathsf{calm} \to \mathsf{turb}) = p_{\text{enter}}, \qquad \Pr(\mathsf{turb} \to \mathsf{calm}) = p_{\text{exit}}. \tag{20}$$

Its stationary turbulent fraction and mean turbulent spell length are

$$\pi_{\text{turb}} \;=\; \frac{p_{\text{enter}}}{p_{\text{enter}} + p_{\text{exit}}}, \qquad \mathbb{E}[\text{spell}] \;=\; \frac{1}{p_{\text{exit}}}. \tag{21}$$

A turbulent step does two things at once: it scales both innovation streams up by $\kappa_{\text{vol}} > 1$ to increase volatility, and it damps mean reversion toward a random walk,

$$m_t = \begin{cases} \kappa_{\text{vol}} & g_t = \textsf{turb} \\ 1 & g_t = \textsf{calm} \end{cases}, \qquad \phi_t = \begin{cases} 1 - (1-\phi)\,\kappa_{\text{rev}} & g_t = \textsf{turb} \\ \phi & g_t = \textsf{calm}, \end{cases} \tag{22}$$

where $\kappa_{\text{rev}} \in [0, 1]$ is the reversion-damping factor: $\kappa_{\text{rev}} = 0$ leaves reversion intact, while $\kappa_{\text{rev}} = 1$ sets $\phi_t = 1$, i.e. the spread becomes a pure random walk with no mean reversion. This assumes a co-occurrence of high volatility and weakened mean reversion during stress, and the regime mechanism is the standard Markov-switching device for financial time series [7].

D.4 Calibration

All parameters are estimated on a real training window; the fit returns the dict of floats in Table 6, which is the single source of truth for the generator.

Table 6: DGP parameters and their estimators (fit per pair on the train window).

Symbol	Meaning	Estimator
$\phi,\ \theta,\ \sigma_s$	spread persistence, mean, shock scale	OLS of s_t on s_{t-1}: ϕ, $\theta = c/(1-\phi)$, residual s.d.
H	spread half-life	$\ln 2/(-\ln \phi)$
β	hedge ratio	Johansen cointegrating vector
$\mu_b,\ \sigma_b$	leg-B drift, calm volatility	mean / calm-subset s.d. of leg-B log returns
$p_{\text{enter}},\ p_{\text{exit}}$	regime transition rates	rolling-vol threshold labelling (see below)
κ_{vol}	turbulent vol multiplier	$\text{sd}(\text{turb returns})/\text{sd}(\text{calm returns})$
κ_{rev}	reversion damping	$(1-\phi_{\text{turb}})/(1-\phi_{\text{calm}})$, clipped to $[0, 1]$
ν	Student-t dof	kurtosis inversion (23), clipped to $[4.5, 30]$

Regime labelling. A day is labelled turbulent when the trailing 21-day volatility of leg-B returns exceeds its 80th percentile. The transition rates follow from the labelled sequence as $p_{\text{exit}} = 1/\overline{\text{spell}}$ (one over the mean turbulent spell) and $p_{\text{enter}} = (\#\text{entries})/(\#\text{calm days})$, and $\phi_{\text{calm}}, \phi_{\text{turb}}$ are AR(1) slopes fit on the day-pairs lying wholly inside each regime.

Tail calibration. For a two-state scale mixture of standardised t_ν innovations (calm scale 1, turbulent scale κ_{vol}, turbulent weight $w_t = \pi_{\text{turb}}$), the marginal excess kurtosis is

$$K \;=\; \left(3 + \tfrac{6}{\nu-4}\right) R - 3, \qquad R \;=\; \frac{w_c + w_t\,\kappa_{\text{vol}}^4}{\left(w_c + w_t\,\kappa_{\text{vol}}^2\right)^2}, \quad w_c = 1 - w_t. \tag{23}$$

We invert (23) on the *empirical* leg-B excess kurtosis to recover ν. Fitting the total (rather than calm-only) kurtosis is essential: the regime mixing already contributes kurtosis through $R \geq 1$, and ν absorbs only the remainder.

D.5 Domain randomization

To prevent the agent from memorising one calibrated path, each episode perturbs the per-realization parameters around their calibrated values (jitter $j = 0.30$):

$$\begin{aligned} H &\leftarrow H \cdot \mathcal{U}(0.5, 2), & \sigma_s &\leftarrow \sigma_s\, e^{\mathcal{N}(0,j)}, \\ \theta &\leftarrow \theta + \mathcal{N}(0, \sigma_s), & \sigma_b &\leftarrow \sigma_b\, e^{\mathcal{N}(0,j)}, \\ \beta &\leftarrow \beta \cdot \mathcal{U}(1-0.2j,\ 1+0.2j), & \kappa_{\text{vol}} &\leftarrow \kappa_{\text{vol}} \cdot \mathcal{U}(0.8, 1.3), \end{aligned} \tag{24}$$

with the leg-B drift jittered at the horizon-scaled rate $\mu_b \leftarrow \mu_b + \mathcal{N}(0, \sigma_\mu)$. The agent is thus trained on a *distribution* of markets sharing the real pair's character but differing in their exact dynamics — the standard domain-randomization route to a policy that transfers to the single, fixed realisation that is the real out-of-sample data.

References

[1] Andrew Ang and Allan Timmermann. Regime changes and financial markets. *Annu. Rev. Financ. Econ.*, 4(1):313–337, 2012.

[2] David A Dickey and Wayne A Fuller. Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American statistical association*, 74(366a):427–431, 1979.

[3] Binh Do and Robert Faff. Does simple pairs trading still work? *Financial analysts journal*, 66(4):83–95, 2010.

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

[5] Robert F Engle and Clive WJ Granger. Co-integration and error correction: representation, estimation, and testing. *Econometrica: journal of the Econometric Society*, pages 251–276, 1987.

[6] Saeid Fallahpour, Hasan Hakimian, Khalil Taheri, and Ehsan Ramezanifar. Pairs trading strategy optimization using the reinforcement learning method: a cointegration approach. *Soft Computing*, 20(12):5051–5066, 2016.

[7] James D Hamilton. A new approach to the economic analysis of nonstationary time series and the business cycle. *Econometrica: Journal of the econometric society*, pages 357–384, 1989.

[8] Zhengyao Jiang, Dixing Xu, and Jinjun Liang. A deep reinforcement learning framework for the financial portfolio management problem. *arXiv preprint arXiv:1706.10059*, 2017.

[9] Søren Johansen. Statistical analysis of cointegration vectors. *Journal of economic dynamics and control*, 12(2-3):231–254, 1988.

[10] 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.

[11] 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.

[12] William F Sharpe et al. The sharpe ratio. *Streetwise–the Best of the Journal of Portfolio Management*, 3(3):169–85, 1998.

[13] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 2 edition, 2018.

[14] Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In Sara A. Solla, Todd K. Leen, and Klaus-Robert Müller, editors, *Advances in Neural Information Processing Systems 12 (NIPS 1999)*, pages 1057–1063. MIT Press, 2000.

[15] Mu-En Wu, Jia-Hao Syu, Jerry Chun-Wei Lin, and Jan-Ming Ho. Portfolio management system in equity market neutral using reinforcement learning. *Applied Intelligence*, 51(11):8119–8131, 2021.

[16] 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.