Extended Abstract

Motivation. Large-order optimal execution remains dominated by classical stochastic-control methods, such as TWAP, VWAP, and Almgren–Chriss (AC). These methods are widely used because they are simple, stable, and interpretable, but their execution schedules are fixed before trading begins and therefore cannot fully adapt to short-horizon predictive signals or intraday regime changes. Reinforcement learning offers a natural framework for adaptive execution, since a policy can condition on many observables such as inventory, urgency, price momentum, volume, volatility, and other market-state variables. In practice, however, two obstacles have limited the use of RL in this setting: the scarcity of high-quality interaction data and low signal-to-noise ratios.

Method. First, we construct a regime-randomized execution simulator designed based off of the liquidity of US large-cap equities. Each episode represents a single trading day with stochastic volume, stochastic volatility, temporary impact, transient impact, permanent impact, over-participation penalties, flow-detection penalties, and a market signal (alpha). The goal is for the agent to minimize the cost of the execution of a large parent order by the end of the day. Rather than learning an execution schedule from scratch, we parameterize the policy as a bounded deviation from one of the above heuristic schedules,

$$a_t = \mathrm{clip}\big(v_t(1+\delta_t), 0, 1\big),$$

where v_t is the action prescribed by the baseline, and δ_t is sampled from a Gaussian policy.

Training Pipeline. This deviation policy is refined using either PPO or REINFORCE. The main variance-reduction tool is a paired-baseline reward, as for every sampled regime and random seed, the learned policy and its anchor baseline are rolled out on the same approximate market path. The training reward is the policy's implementation shortfall relative to the baseline's on that identical episode. This paired counterfactual design cancels price-path noise and turns the reward into a direct estimate of incremental execution improvement.

Results. The strongest configurations produce statistically significant and economically meaningful improvements over classical execution baselines. The best overall model, PPO initialized around VWAP without the alpha signal, improves over TWAP by $+1.27$ bps per episode with $p_t < 10^{-4}$ and a win rate of 58.5%. Across the 1000-episode evaluation horizon, this corresponds to approximately $+1274$ cumulative basis points of retained execution value. The best alpha-informed configuration, PPO initialized around TWAP with the alpha signal active, improves over TWAP by $+1.13$ bps per episode with $p_t = 0.003$ and a 95% confidence interval of approximately $[+0.38, +1.88]$ bps. Removing the alpha feature for TWAP, as an example, reduces the improvement from roughly $+1.13$ bps to $+0.12$ bps and makes the result statistically insignificant. This suggests that in this framework, we are able to embed informative actions better into the execution pipeline.

Discussion. PPO consistently outperforms REINFORCE in this environment. REINFORCE policies generally learn to reproduce their anchors and produce only limited improvements, typically below 0.5 bps per episode. PPO produces all of the largest gains, which is consistent with the structure of the execution problem as the policy must identify small beneficial deviations from well informed baselines under a very noisy raw reward signal.

Conclusion. This work shows that reinforcement learning can become usefull and outperform classical optimal-execution heuristics when domain knowledge is embedded directly into the policy class and reward design. Future work should test the same framework in richer order-book simulations, mid- and small-cap regimes, offline execution datasets, and live paper-trading environments.

Alpha-Informed Optimal Trading Execution

Reinforcement Learning with Domain Informed Priors

Ryan Padnis
Department of Statistics
Stanford University
rnpadnis@stanford.edu

Abstract

Reinforcement learning (RL) has not yet found its footing in quantitative finance, particularly in the domain of optimal trading execution where classical stochastic control approaches still dominate. In this work, we address two central challenges that limit the practical adoption of RL in execution: demonstrating its tangible advantages over established baselines and overcoming the scarcity of interaction data. By coupling simulated market and paired-baseline reward, we show RL is indeed a viable approach to the this environment. This approach yields stable training, improved sample efficiency, and robustness to distributional shifts. Empirically, we show that the learned policies outperform standard execution benchmarks by $+1$ to $+2$ basis points per episode.

1 Introduction and Related Works

The optimal execution literature has been developed extensively within the framework of stochastic control, where the objective is to minimize trading costs while managing execution risk. Early work by Bertsimas and Lo [1998] formulates execution as a dynamic programming problem under relatively simple price dynamics. Through simplifying impact assumptions and solving the Bellman equations via backward induction, this work establishes one of the foundational connections between optimal execution and control theory.

The seminal contribution in this space, that of Almgren and Chriss [2000] (AC), introduces a Markowitz-equivalent mean–variance framework for execution, modeling the trade-off between expected market impact and price risk. In this model, temporary and permanent price impacts are assumed to be linear functions of trade size, volatility is treated as constant over the execution horizon, and the resulting problem reduces to closed-form convex optimization. The solutions — the efficient frontier of optimal trading strategies — are deterministic and time-scheduled, built before the trading day, or episode, even begins. Contrary to later empirical work, AC argues that the introduction of new information through drift or spikes has minimal effects on the theoretical optimal trading schedule. While this framework remains highly influential due to its tractability, its core assumptions are empirically restrictive in transient, high-frequency environments. In later years, it has been shown that market impact behavior is closer to a concave power law in traded volume [Bouchaud et al., 2004].

Both these papers laid the groundwork for formulating the most simple heuristic execution strategies: Time-Weighted Average Price (TWAP) and Volume-Weighted Average Price (VWAP). TWAP executes trades uniformly over a fixed time horizon, implicitly this assuming constant liquidity and impact, while VWAP schedules trades proportionally to expected market volume, aiming to minimize slippage, or deviations, from the market's average traded price [Konishi, 2002, Kissell, 2013]. Both are widely used in practice and are often regarded as difficult-to-beat benchmark strategies due to their simplicity, robustness, and minimal reliance on model assumptions.

To try to relax some of the trading environment assumptions, Almgren [2003] generalizes the impact model to nonlinear forms, allowing for more realistic cost models. Similarly, Obizhaeva and Wang [2013a] introduce a structural model of the limit order book, which captures the notion of liquidity replenishment after trades. This model provides a more mechanical understanding of market impact but relies on strong parametric

assumptions about the dynamics of the order book, which are difficult to calibrate and hardly generalize across market regimes day-to-day and asset-to-asset.

Beyond static strategies, Almgren [2012] extends the models to account for stochastic liquidity and volatility, allowing trading rates to adjust in response to evolving market conditions. Similarly, Cartea and Jaimungal [2016] introduce signal-based execution frameworks in which trading decisions depend explicitly on predictive signals and observable state variables. These approaches represent a step in the direction of adaptive execution, as they allow schedules to respond in real time to market information.

Parallel work in market microstructure highlights the limitations of simplistic impact models. Gatheral [2010] establish no-dynamic-arbitrage conditions that constrain admissible forms of market impact. These limitations motivate novel approaches, particularly in reinforcement learning (RL). Early work by Nevmyvaka et al. [2006] shows that RL-based policies can outperform simple execution heuristics, but relies on low-dimensional state representations and tabular methods. Subsequent deep RL approaches, such as Ning et al. [2021], improve scalability but still simplify the action space, limiting execution flexibility.

Altogether these works present key tradeoffs: classical models provide structure but lack adaptability, while RL-based approaches offer flexibility but require substantial data collection and careful training. To the best of our knowledge, prior work in optimal execution has not jointly considered the integration of heuristic strategies with RL for policy refinement.

Thus, to enable effective learning in a simulated environment without access to calibrated market interaction data, we combine the above insights to generate a new framework. Rather than relying on exact execution trajectories or proprietary order book data, the environment incorporates domain-informed priors on price dynamics, volume evolution, and market impact, allowing the agent to learn robust execution policies under plausible market conditions.

2 Environment

We construct a (buy) execution simulator informed by the trading volume of US large-cap equities (NVIDIA top of book statistics over last year) at 1-minute intervals. An episode covers a single NYSE trading day: $t = 0, 1, \ldots, T-1$ with $T = 390$ and $\Delta t = 1$ min. At $t = 0$ the agent is given a large block of Q shares I.I.D proportional to the average daily volume (ADV). To emphasize the more difficult aspects of execution, the order is assumed to constitute a meaningful share of total daily volume, so the market impact is quite large and thus important. The agent must liquidate the order by $t = T - 1$; whatever remains is closed by a forced terminal liquidation block (Section 2.5). Each component below was selected to expose the agent to structural features that the static benchmarks (TWAP, VWAP, AC) cannot exploit.

2.1 Regime Randomization

To prevent overfitting to local shifts that don't generalize across volume/volatility regimes, we sample a fresh set of hyperparamterws at the start of each episode. The regime is parameterized by (i) the intraday seasonality (U-shape) of volume and volatility, (ii) activity bursts expressed as Hawkes parameters, (iii) the expected level of base volume per minute $\bar{Y}$ and base volatility $\bar{\sigma}$, (iv) the principal order size Q as a fraction of expected daily volume, and (v) the impact coefficients $(b_{\text{temp}}, b_{\text{trans}}, b_{\text{perm}})$ together with the square-root exponent γ. Each parameter is drawn independently from a uniform distribution centered on values that match the empirical ranges in Almgren et al. [2005].

2.2 Stochastic Market Dynamics

To preserve seasonality and clustering, we use a Hawkes' process for the volume model:

$$Y_t = m_Y(t) \cdot \exp(x_t), \quad x_{t+1} = x_t - \kappa_Y x_t \Delta t + c_Y \tilde{z}_{Y,t} \Delta t + \eta_Y \sqrt{\Delta t}\, \varepsilon_t^Y, \tag{1}$$

where $m_Y(t) = \bar{Y}\, s_Y(t)$ is the seasonal mean and $\tilde{z}_{Y,t} = z_{Y,t}/(1 + z_{Y,t})$ is the saturated Hawkes shot-noise. Volatility itself takes the form of an Ohrenstein-Ulhenbeck (OU) process in log-variance with seasonal mean $m_H(t) = 2\log(\bar{\sigma} s_\sigma(t))$, and the instantaneous volatility is $\sigma_t = \exp(H_t/2)$. Two independent Hawkes drivers $\{z_{Y,t}, z_{H,t}\}$ allow volume and volatility to cluster on different timescales, what is noted by Cont [2001] as what is called a stylized fact in financial markets. This also allows for the cross-correlation between the two to decay slowly over time.

The base midprice M_t, i.e. the listed price, follows Geometric Brownian Motion (GBM) equipt with this stochastic volatility model. In some of the trials, we add an *alpha-drift* component α_t described in Section 2.4

below:

$$M_{t+1} = M_t \exp\Big(-\tfrac{1}{2}\sigma_t^2 \Delta t + \sigma_t \sqrt{\Delta t}\, \varepsilon_t^S + \alpha_t \Delta t\Big). \tag{2}$$

The transient impact state I_t decays exponentially: $I_{t+1} = e^{-\delta \Delta t} I_t$, and the observable midprice is $P_t^{\text{mid}} = M_t + I_t$. The decoupling between M_t and I_t follows the market impact model of Obizhaeva and Wang [2013b].

2.3 Market Impact

Each fill applies impact in multiple pieces, each triggering at different time scales.

$$\rho_t = q_t / Y_t, \qquad \Delta P_t^{(\cdot)} = P_t^{\text{mid}} \cdot \frac{b_{(\cdot)}\, \rho_t^{\gamma}}{10^4}. \tag{3}$$

Temporary impact is paid only by the current fill, transient impact is added to I_t and decays, and permanent impact perturbs the latent mid-price M_t multiplied by a log-normal shock $\xi_t \sim \text{LogNormal}(0, \sigma_{\text{perm}})$.

Now, we add the following corrective terms to make the policy learning more nontrivial and less-exploitative:

Over-participation penalty. When the agent trades too large a fraction of market volume, execution costs increase sharply and nonlinearly. We model this with

$$\Delta_{\text{kick}}(\rho_t) = c_{\text{kick}} \big(\max\{0, \rho_t - \rho^\star\}\big)^2 \quad [\text{bps}], \tag{4}$$

where $\rho^\star$ is the participation level above which trading becomes especially costly. This discourages large bursts of trading and reflects the fact that liquidity becomes thinner after several price levels are consumed, even though we are not exposed to that information.

Flow detection. If the agent's recent average participation is too high, other market participants may infer the presence of a large informed order. When the trailing-window mean participation exceeds ρ^{det}, we increase permanent impact by a multiplier $m_{\text{flow}}(t) \in [1, m_{\text{flow}}^{\max}]$. This penalizes execution strategies whose trading pattern is easy to distinguish from normal market volume, giving us the implicit goal of a stealthy policy.

2.4 Alpha Drift (Exploitable Signal)

To measure how well our learned algorithms can evaluate exploitative market information, in half of the policy trains we add the drift component α_t in Eq. (2). It evolves as an AR(1) process per episode:

$$\alpha_t = \rho\, \alpha_{t-1} + \epsilon_t^\alpha, \qquad \epsilon_t^\alpha \sim \mathcal{N}(0, \sigma_\alpha^2), \qquad \alpha_0 = 0. \tag{5}$$

Because α_t enters the price additively in log-space, the log-return contains a memory component that is correlated with the lagged momentum features the agent observes.

2.5 Forced Terminal Liquidation

If at $t = T - 1$ the residual fraction $r_T = R_T / Q$ is nonzero, we charge the agent a 100%-participation impact cost multiplied by r_T and the prevailing m_{cum}:

$$r_T^{\text{liq}} = -10^4 \cdot \Big(\tfrac{s_T/2 + b_{\text{temp}}\, m_{\text{cum}}(T)}{10^4}\Big)\, r_T. \tag{6}$$

Thus, an under-execution policy strictly loses to a fully-executing one.

2.6 State, Action, and Reward

State. The observation $s_t \in \mathbb{R}^{28}$ consists of temporal embeddings, inventory features (remaining fraction, urgency $r_t/[1 - t/T]$, log shares left), price features (price drift, multi-window momentum), microstructure features, etc. All features are standardized on a sampled training set. These are all typical market microstrucutral features that traders would consider in a realistic environment.

Action. The action $a_t \in [0, 1]$ is the fraction of remaining inventory to submit (buy initiated volume). The requested quantity $q_t^{\text{req}} = \lceil a_t R_t \rceil$ is then capped by available volume Y_t and inventory R_t, giving $q_t = \min\{q_t^{\text{req}}, Y_t, R_t\}$.

Reward. The per-step reward is the arrival price (at $t = 0$) implementation shortfall (IS) in basis points [Perold, 1988]:

$$r_t = -10^4 \cdot \frac{(P_t^{\text{fill}} - P_0)\, q_t}{P_0\, Q}, \tag{7}$$

plus the terminal r_T^{liq} defined in Section 2.5.

2.7 Baselines: VWAP, TWAP, AC

We benchmark every learned policy against three deterministic schedulers run on the same seed and same regime as the policy being evaluated, producing a paired sample for each episode. Pairing dramatically reduces variance because both the policy and the baseline see the identical price path within each episode [Perold, 1988]. To build some of these heuristcs, this requires prior "historical" information, combined with some type of reweighting with new information.

Prior. We build a prior knowledge of information once through simulating N independent days under the regime distribution and recording (i) the per-minute volume profile $v_d \in \mathbb{R}^T$, (ii) the realized volatility σ_d, and (iii) the temporary-impact coefficient η_d. The prior consists of

$$\begin{aligned} \bar{p} &= \tfrac{1}{N} \sum_{d=1}^{N} v_d / \|v_d\|_1 && \text{(normalised mean profile)}, \\ \mu_\sigma,\ s_\sigma &= \text{mean}(\sigma_d),\ \text{std}(\sigma_d) && \text{(volatility prior)}, \\ \mu_\eta &= \text{mean}(\eta_d) && \text{(impact prior)}. \end{aligned}$$

2.7.1 TWAP

TWAP partitions the order X_0 into T equal slices and trades a constant fraction of remaining inventory at each bar:

$$s_t^{\text{TWAP}} = \frac{1}{T - t}, \qquad x_t^{\text{TWAP}} = s_t^{\text{TWAP}} \cdot X_t,$$

where X_t is the inventory left at the start of bar t. TWAP requires no prior or intraday update: by construction it depends only on the horizon T. It is optimal under the AC model when volatility is constant and risk aversion $\lambda \to 0$ [Almgren and Chriss, 2000].

2.7.2 VWAP

VWAP seeks to execute in proportion to intraday market volume so that the realized execution price tracks the market's volume-weighted average price. Let v_t denote market volume at bar t, and let $\bar{p}$ be the historical average intraday volume profile, i.e. the prior over relative volume shares above that is based off of [Konishi, 2002]. At each step, the scheduler maintains an effective profile p_t^{eff}, initialized from $\bar{p}$ and updated using realized volume from earlier bars. The trading fraction is then

$$s_t^{\text{VWAP}} = \frac{p_t^{\text{eff}}}{\sum_{i \geq t} p_i^{\text{eff}}},$$

which ensures execution remains aligned with both observed and expected market liquidity throughout the day.

2.7.3 AC

AC solves for the optimal execution path under linear temporary impact and a mean–variance trade-off between market impact and timing risk. Let σ denote per-bar volatility, η the temporary impact coefficient, and $\lambda > 0$ the risk-aversion parameter. The optimal inventory trajectory is governed by

$$\kappa = \sqrt{\frac{\lambda \sigma^2}{\eta}},$$

where larger κ implies faster execution. During the trading day, the scheduler updates volatility using realized log-returns and a Bayesian posterior that blends the historical prior with the observed sample standard deviation [Almgren, 2012, Cartea and Jaimungal, 2016], while temporary impact is held fixed since

it is difficult to infer from passive price data [Almgren et al., 2005, Gatheral, 2010]. Using the updated posterior volatility, the fraction of remaining inventory traded at time t is

$$s_t^{\text{AC}} = \frac{\kappa_t}{\tanh\big(\kappa_t(T-t)\big)}.$$

As $\lambda \to 0$, the strategy approaches TWAP; as λ increases, the model front-loads execution to reduce exposure to price risk.

So in this framework, we have constructed realistic and dynamic VWAP and AC algorithms, along with TWAP as a static baseline for comparison.

3 Reinforcement Learning Pipeline

We consolidate the optimal execution environment into finite-horizon episodic MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, T)$ and learn a gaussian policy π_θ in two stages: (i) a warm-start policy embedded into a neural network that imitates one of the three baseline schedulers of Sec. 2.7, then (ii) policy-gradient refinement under PPO or REINFORCE.

3.1 Parameterizing the Baseline

To avoid poor initialization, we parameterize the policy as a multiplicative factor around the baseline schedule v_t:

$$\delta_t \sim \pi_\theta(\cdot \mid s_t), \qquad \tilde{\delta}_t = \text{clip}(\delta_t, -\delta_{\max}, +\delta_{\max}), \qquad a_t = \text{clip}\big(v_t \cdot (1+\tilde{\delta}_t),\ 0,\ 1\big), \tag{8}$$

with $\delta_{\max} = 0.20$. So in essence this is learning when it is optimal to deviate from the "experts" in favor of finding a more optimal path.

The policy is a 3-layer MLP with a scalar μ-head and a learnable $\log\sigma$ head. We initialize $\log\sigma = -3.5$ ($\sigma \approx 0.030$) so the warm-start policy almost never clips against $\pm\delta_{\max}$, and we freeze $\log\sigma$ during RL to prevent both variance collapse (which kills exploration) and variance explosion (which produces constant clipping). This was noticed in preliminary findings that made learning too unstable. In the evaluation stage, we use μ for the actions we take which also gives more stability.

3.2 Policy-Gradient Methods

REINFORCE. The vanilla policy-gradient estimator uses the estimate:

$$\hat{g}_\theta^{\text{RF}} = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(\delta_t^{(i)} \mid s_t^{(i)})\big(G_t^{(i)} - V_\phi(s_t^{(i)})\big), \tag{9}$$

with rewards-to-go $G_t = \sum_{t'\geq t} \gamma^{t'-t} r_{t'}$ and V_ϕ the value function which is trained alongside.

PPO. From **?**:

$$L^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t\Big[\min\big(\rho_t(\theta)\hat{A}_t,\ \text{clip}(\rho_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\big)\Big], \tag{10}$$

with $\rho_t(\theta) = \pi_\theta(\delta_t \mid s_t)/\pi_{\theta_{\text{old}}}(\delta_t \mid s_t)$ and $\epsilon = 0.2$. Each rollout batch is consumed by $K = 10$ epochs of minibatch SGD with batch size $B = 256$, with inner-loop early stopping when $\mathbb{E}[\text{KL}(\pi_{\text{old}} \| \pi_\theta)] > \kappa_{\text{KL}} = 0.05$, to prevent deviating too far within each update.

Generalized advantage estimation (GAE) For PPO, we use the bootstrapping method

$$\hat{A}_t^{\text{GAE}} = \sum_{l\geq 0}(\gamma\lambda)^l \delta_{t+l}, \qquad \delta_t = r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t), \tag{11}$$

with $\gamma = 0.99$, $\lambda = 0.95$.

Value function. A 3-layer MLP V_ϕ (width 128) is trained alongside the policy by MSE on empirical returns over $E_V = 20$ inner epochs per outer iteration with gradient norm clipped at 0.5.

3.3 Variance Reduction: Paired Baseline Reward

The dominant obstacle is reward variance in the alpha informed environment. The single-episode IS has standard deviation $\sigma_{\text{IS}} \approx 1800$ bps while the realistic alpha edge is $\Delta \approx 10$ bps, giving signal-to-noise ratio < 0.01. We attack this using control variates: at each rollout episode, we re-run the matching baseline on the same seed and regime and record the per-step baseline reward r_t^{base}. The training reward is

$$\tilde{r}_t = r_t - r_t^{\text{base}}, \tag{12}$$

the per-step relative cost the policy incurs versus its anchor. The price path noise from the GBM and the alpha process cancels, and the variance of $\sum_t \tilde{r}_t$ is empirically 5–20$\times$ smaller than the unpaired counterpart for the same policy. The estimator remains unbiased because r_t^{base} is independent of θ given s_t.

3.4 Entropy Controls

We enforce a trust-region constraint $\kappa_{\text{KL}} = 0.05$ between successive policy iterates and apply an entropy bonus $c_{\text{ent}}\mathcal{H}(\pi_\theta(\cdot \mid s))$ initialized at 0.01 and decayed by 0.99 per update (outer epoch).

3.5 Experimental Design

To compare all of these algorithms, we run a sweep of different setups crossing $\{\text{PPO}, \text{REINFORCE}\} \times \{\text{VWAP}, \text{TWAP}, \text{AC}\} \times \{\alpha\text{-on}, \alpha\text{-off}\}$:

All runs share: 120 outer epochs, 128 episodes per epoch, evaluation every 10 epochs on 128 held-out seeds, paired reward, target-KL 0.05, and decaying entropy bonus. The α-on runs use $(\rho, \sigma_\alpha) = (0.95, 8\,\text{bps})$; the α-off cells set $\sigma_\alpha = 0$. Learning rates are 3×10^{-4} for PPO and 1×10^{-4} for REINFORCE, reflecting REINFORCE's higher gradient variance.

4 Results

4.1 Training Dynamics

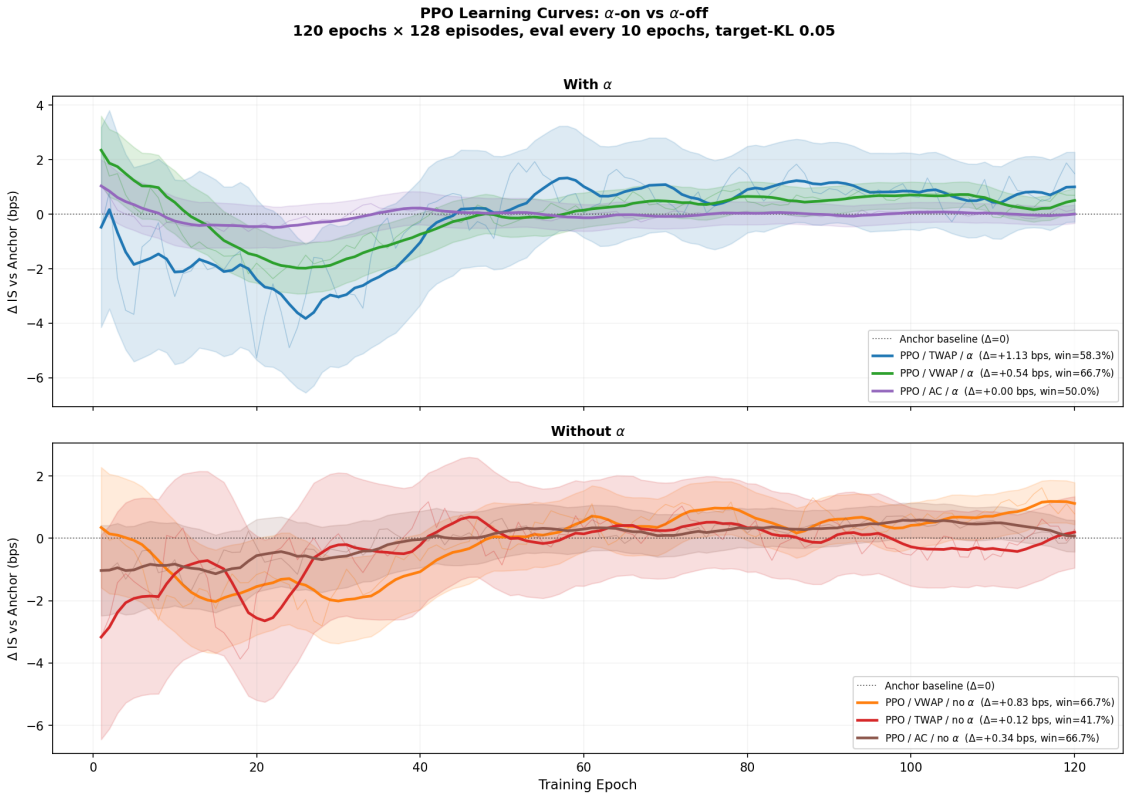

Figure 1: Moving average training reward across the main algorithmic configurations.

Figure 1 shows that most configurations begin with little exploitable signal, as expected in a noisy execution environment. Over training, however, the best PPO runs eventually learn well defined deviations from their anchors.

4.2 Evaluation Summary

We evaluate each trained policy over 1000 random episodes, using matched random seeds across policies and baselines.

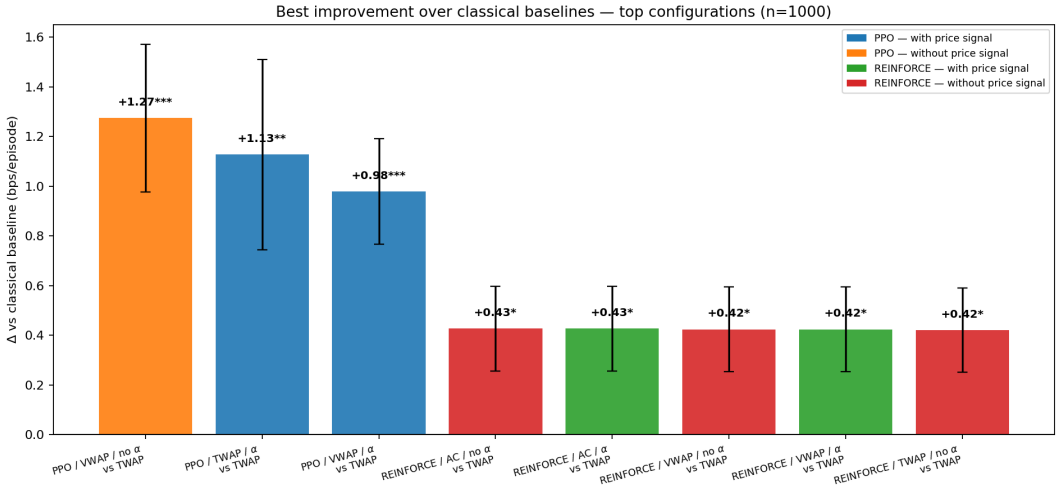

Figure 2: Policy improvement over baselines, strongest configurations. $^{*}p<0.05$, $^{**}p<0.01$, $^{***}p<0.001$.

Algo	Anchor	α	Δ_{best}	Best vs	Δ_{anchor}	SE	Win%	Cum. (bps)	Rel. gain %	p_t
PPO	VWAP	no	**+1.27**	TWAP	**+0.83**	0.30	58.5	**+1274**	**+2.38**	$<10^{-4}$
PPO	TWAP	yes	**+1.13**	TWAP	**+1.13**	0.38	53.5	**+1127**	**+2.10**	0.003
PPO	VWAP	yes	**+0.98**	TWAP	**+0.54**	0.21	54.0	**+978**	**+1.83**	$<10^{-4}$
REINFORCE	AC	no	**+0.43**	TWAP	**+0.43**	0.17	54.1	**+427**	**+0.80**	0.012
REINFORCE	AC	yes	**+0.43**	TWAP	**+0.43**	0.17	54.1	**+427**	**+0.80**	0.012
REINFORCE	VWAP	no	**+0.42**	TWAP	−0.02	0.17	47.6	**+424**	**+0.79**	0.013
REINFORCE	VWAP	yes	**+0.42**	TWAP	−0.02	0.17	47.6	**+424**	**+0.79**	0.013
REINFORCE	TWAP	no	**+0.42**	TWAP	**+0.42**	0.17	54.3	**+420**	**+0.78**	0.013
REINFORCE	TWAP	yes	**+0.42**	TWAP	**+0.42**	0.17	54.3	**+420**	**+0.78**	0.013
PPO	AC	no	**+0.34**	TWAP	**+0.34**	0.15	53.9	**+345**	**+0.64**	0.018
PPO	TWAP	no	+0.12	TWAP	+0.12	0.34	53.4	+119	+0.22	0.724
PPO	AC	yes	+0.00	TWAP	+0.00	0.10	48.3	+3	+0.01	0.975

Table 1: Consolidated 1000-episode paired evaluation. Δ_{best} reports each policy's strongest paired improvement against a classical baseline, while Δ_{anchor} reports improvement against the heuristic used during training. Bold entries denote positive statistically significant improvements.

Table 1 gives the summary of the evaluation results. The strongest overall configuration is PPO trained against VWAP without the price signal, which improves over TWAP by $+1.27$ bps per episode and over its own VWAP anchor by $+0.83$ bps. The next two strongest configurations are PPO/TWAP with α, and PPO/VWAP with α. This concentration of the best results among PPO policies suggests that stable policy optimization is more essential than unbiased gradients when the reward signal is quite small.

4.3 Effect of the Price Signal

The clearest controlled test of the price signal is the PPO/TWAP comparison. Without α, PPO/TWAP improves by only $+0.12$ bps per episode and is not statistically significant. With α, the improvement rises to $+1.13$ bps per episode with $p_t = 0.003$. This is consistent with the role of TWAP as a flat, non-adaptive

schedule because TWAP ignores intraday price dynamics, so a policy with access to α can improve by accelerating when short-term prices are favorable and slowing down when they are unfavorable.

The PPO/VWAP results tell a slightly different story. PPO trained against VWAP performs well both with and without α, improving significantly in both cases. This suggests that volume-profile information already provides a useful structure for learning around VWAP, whereas α is most valuable when the anchor itself is less responsive to intraday price movement. In other words, α matters most when the baseline leaves more timing and market information unused.

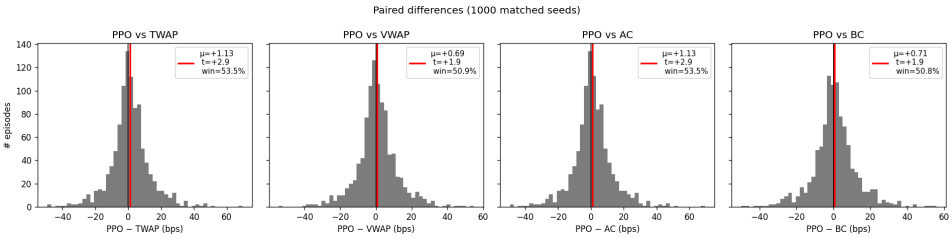

Figure 3: PPO / TWAP / α episode-by-episode paired differences against TWAP.

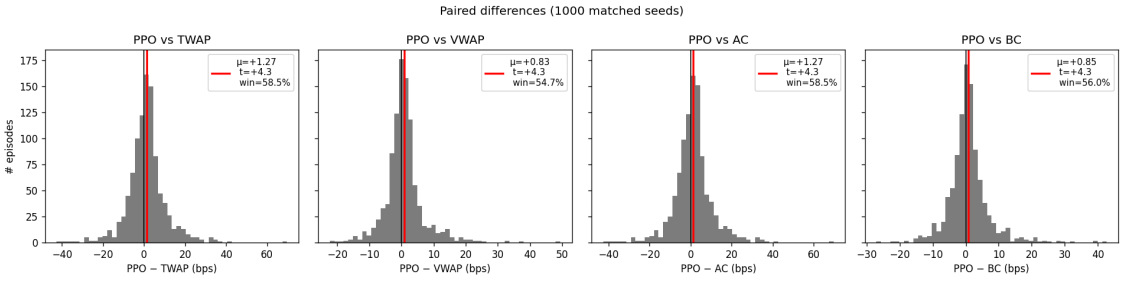

Figure 4: PPO / VWAP / no α episode-by-episode paired differences against TWAP.

Figures 3 and 4 show that the improvements are driven by repeatable simulations of data. The learned policies produce small but persistent paired gains across many simulated executions. The average improvement is modest, but the law of large numbers tells us this repeated advantage makes the result statistically and thus economically meaningful.

4.4 PPO vs. REINFORCE

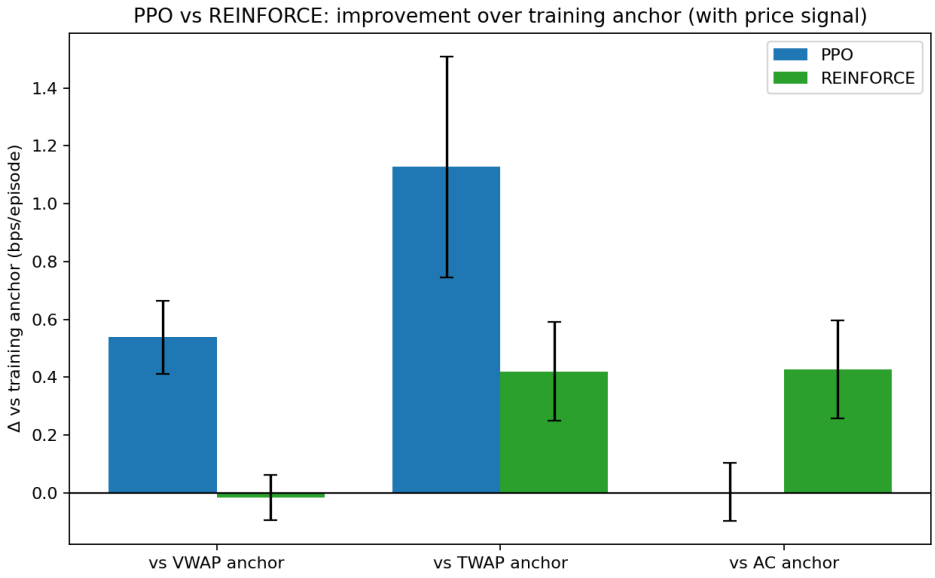

Figure 5: Improvement over the training anchor for PPO and REINFORCE with the α price signal active.

Figure 5 and Table 1 show a the separation between PPO and REINFORCE. Every improvement larger than roughly half a basis point comes from a PPO configuration, while REINFORCE remains clustered around smaller gains.The reward is computed from terminal implementation shortfall, so individual actions receive only indirect credit, of which PPO can more clearly assign correctly through the clipped surrogate objective. Several REINFORCE variants still produce small statistically significant improvements over TWAP.

4.5 Economic Significance

The raw gains are measured in basis points, so the absolute numbers can appear small. At institutional scale, however, a one-basis-point improvement is meaningful. For example, on a $10M execution, one basis point is approximately $1,000. Repeated over 1000 comparable executions, a persistent one-basis-point improvement corresponds to roughly $1M in execution-cost savings. Thus, the practical importance of the learned policy is not that it radically changes the execution schedule since we know it already satisfies some aspect of optimality (albeit under stricter conditions), but rather we could benefit from timely deviations.

5 Discussion

The broader lesson from these experiments is that optimal execution is a difficult setting for RL because the performance edge is small relative to the noise. Unlike many benchmark RL environments, where a good policy visibly changes the reward distribution, execution policies often differ by only a few basis points while being evaluated under large stochastic price variation. The central methodological challenge is not simply finding a more expressive policy class, or a better reward function, but designing a reliable learning procedure.

A second implication is that, as hypothesized, the choice of anchor changes the residual learning problem. Training around TWAP, VWAP, or AC resulted in quite different outputs. If the anchor captures much of the useful structure in the environment, then the learned policy has less room to improve and may mostly learn small timing corrections, hence why the results on AC are the least informative. This, in our opinion, reaffirms that the RL execution is best implemented as a residual-control problem as reward assignment can be incredibly complex without the paired baseline reward. This framing also changes how weak results should be interpreted, as a policy that fails to beat a strong anchor is not necessarily evidence that RL is ineffective. Instead, it may indicate that the simulated environment does not contain enough exploitable state information beyond what the anchor already uses.

There is also an important distinction between exploiting a synthetic signal and learning an actionable trading rule. The alpha process used here is useful because it creates a controlled experiment. When the signal is present, an adaptive policy should be able to condition its execution rate on favorable or unfavorable short-term price movement. However, this does not imply that a comparable signal would be available in real trading. In practice, the predictive signal would need to be estimated from order flow, intraday returns, imbalance, volatility, or other proprietary features, and its stability would have to be tested out of sample. Thus, this may indeed be the most unrealistic aspect of this experiment as it doesn't necessarily translate well to real environment.

Finally, the results must be interpreted in terms of the simulation strength. The environment is intentionally structured enough to compare algorithms cleanly, but real execution involves queue priority, partial fills, hidden liquidity, cancellations, latency, and venue selection. These frictions could reduce the edge, but they could also create additional opportunities for adaptation and should be considered in future works in this space.

6 Conclusion

The main takeaway of this work demonstrates that policy-gradient methods can learn execution policies that achieve statistically significant and economically meaningful improvements over classical optimal-execution benchmarks. The best configuration, PPO / VWAP / no α, outperforms its strongest classical counterpart by $+1.27$ bps per episode ($p_t < 10^{-4}$, win rate 58.5%), equivalent to $+1{,}274$ bps of cumulative retained P&L over the 1000-episode evaluation horizon. At institutional scale, these small per-execution improvements can compound into substantial retained pnl in the large trade frequency limit.

The main conclusion is not that RL should replace TWAP, VWAP, or AC, but that it can learn useful deviations around them. Classical schedules provide robust priors, while RL supplies a way to adapt around those anchors using additional state information.

Future work should test whether these simulated improvements persist in richer and more realistic market environments. One direction is to use agent-based simulators such as ABIDES, which provides a high-fidelity interactive market simulation framework [Byrd et al., 2019]. Another direction is to incorporate historical execution data through behavior cloning or offline reinforcement learning, so that policies can be warm-started from data rather than the control heuristics which contain their own issues. Further extensions should also consider model-based RL, where the transition dynamics of the order book are learned directly in addition to the execution policy. This would help address simulator inaccuracy and move the framework closer to the real institutional setting, where proprietary data, stronger in-house benchmarks, and richer market-state features are available.

Changes from Proposal. The original proposal envisioned a single-baseline (VWAP-only) PPO pipeline trained against fixed-regime episodes. In addition, the intention was use behavior cloning, or filtered behavioral cloning, as a warm start mechanism for the policy refinement. This early on in training, as shown in the milestone, became an issue as prefitting can be too rigid of a baseline under the beta or truncated normal policies. This is when I decided it would be better to reframe actions as deviations from the "expert", which reduced variance and allowed for more exploration without the penalty of massive distributional shift. I also added an explicit alpha-on/alpha-off environment modes as a structural control to demonstrate that the framework's gains are signal-driven rather than artifacts. REINFORCE was also added as a baseline to test against to see how much the clipped surrogate objective really matters in such noisy environments.

AI Disclosure Copilot was used for the development of the OOP trading simulation environment, as well as general debugging and github maneuvers.

Code: github.com/ryanpadnis/RLExecution

References

Dimitris Bertsimas and Andrew W. Lo. Optimal control of execution costs. *Journal of Financial Markets*, 1 (1):1–50, 1998.

Robert Almgren and Neil Chriss. Optimal execution of portfolio transactions. *Journal of Risk*, 3(2):5–39, 2000.

Jean-Philippe Bouchaud, Yuval Gefen, Marc Potters, and Matthieu Wyart. Fluctuations and response in financial markets: The subtle nature of random price changes. *Quantitative Finance*, 4(2):176–190, 2004.

Hizuru Konishi. Optimal slice of a vwap trade. *Journal of Financial Markets*, 5(2):197–221, 2002.

Robert Kissell. *The Science of Algorithmic Trading and Portfolio Management*. Academic Press, 2013.

Robert Almgren. Optimal execution with nonlinear impact functions and trading-enhanced risk. *Applied Mathematical Finance*, 10(1):1–18, 2003.

Anna A. Obizhaeva and Jiang Wang. Optimal trading strategy and supply/demand dynamics. *Journal of Financial Markets*, 16(1):1–32, 2013a.

Robert Almgren. Optimal trading with stochastic liquidity and volatility. *SIAM Journal on Financial Mathematics*, 3(1):163–181, 2012.

Álvaro Cartea and Sebastian Jaimungal. A closed-form execution strategy to target volume weighted average price. *SIAM Journal on Financial Mathematics*, 7(1):760–785, 2016.

Jim Gatheral. No-dynamic-arbitrage and market impact. *Quantitative Finance*, 10(7):749–759, 2010.

Yuriy Nevmyvaka, Yi Feng, and Michael Kearns. Reinforcement learning for optimized trade execution. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 673–680, 2006.

Brian Ning, Franco Ho Ting Lin, and Sebastian Jaimungal. Double deep q-learning for optimal execution. *Applied Mathematical Finance*, 28(4):361–380, 2021.

Robert Almgren, Chee Thum, Emmanuel Hauptmann, and Hong Li. Direct estimation of equity market impact. *Risk*, 18(7):58–62, 2005.

Rama Cont. Empirical properties of asset returns: Stylized facts and statistical issues. *Quantitative Finance*, 1(2):223–236, 2001.

Anna A. Obizhaeva and Jiang Wang. Optimal trading strategy and supply/demand dynamics. *Journal of Financial Markets*, 16(1):1–32, 2013b.

André F. Perold. The implementation shortfall: Paper versus reality. *Journal of Portfolio Management*, 14 (3):4–9, 1988.

David Byrd, Maria Hybinette, and Tucker Hybinette Balch. Abides: Towards high-fidelity market simulation for ai research, 2019.