Cookie Clicker Stock-Market Autotrader: A POMDP Formulation with Cost-Aware Mean-Reversion Control

Linbo Gong

September 25, 2025

Contents

Abstract

The Cookie Clicker "Stock Market" minigame offers a controlled environment for studying trading with state uncertainty, regime changes, and transaction costs. In this paper, I formalize the game as a discounted stochastic control problem under partial observability, derive cost-aware trading band policies from an Ornstein–Uhlenbeck approximation, maintain beliefs with a particle filter, and benchmark implementable performance against a clairvoyant dynamic-programming upper bound using a code-faithful simulator. The approach demonstrates a reproducible research workflow that mirrors quantitative trading practice.

1 Introduction

The "Stock Market" minigame in *Cookie Clicker* provides a compact laboratory for trading under uncertainty with a fully specified microstructure. Prices update in one–minute ticks according to a stochastic mechanism that combines mean reversion toward building–dependent resting values, regime switches that alter drift and volatility, heavy–tailed idiosyncratic shocks, and occasional global disturbances. Trading faces realistic frictions: a proportional overhead on buys (reduced by hiring brokers), finite per–asset inventory capacities tied to progress in the game, cash constraints, and a one–tick prohibition against buying and selling the same asset. Despite the playful setting, this environment captures the core ingredients that preoccupy quantitative finance: state uncertainty, nonstationarity, inventory/risk control, and execution costs—with the added advantage that the ground–truth transition kernel can be recovered from source code.

In this work, I use the minigame as a controlled testbed for cost–aware statistical arbitrage and sequential decision making. I formulate the trader's objective as a discounted stochastic control problem over multiple assets: maximize cumulative profit (measured in the game's normalized currency) subject to inventory, budget, and timing constraints. Because the latent drivers of price (drift, regime, and regime duration) are not directly observable to a legitimate agent, the problem is naturally a partially observed Markov decision process (POMDP). Decisions must be based on filtered beliefs about the hidden state rather than on the state itself.

For analysis and benchmarking, I construct a code–faithful simulator that reproduces the tick dynamics exactly, enabling a "clairvoyant" dynamic programming upper bound under full information. For tractable control, regime–switching mean–reversion models are adopted: a single–regime AR(1)/Ornstein–Uhlenbeck approximation to derive explicit, transaction–cost–aware trading bands and a hidden Markov extension to accommodate regime changes. I maintain beliefs over latent state with a particle filter, producing a practical POMDP controller whose actions depend on posterior means and uncertainties rather than on unobservable internals.

The multiasset nature of the minigame raises an allocation question familiar to portfolio trading: when simultaneous buy signals arise under a shared cash budget, how should capital be assigned across independent opportunities with heterogeneous expected edges and costs? I address this with a budgeted index policy that ranks assets by posterior edge per unit notional and allocates greedily subject to capacity and cash. I also analyze the economics of broker hiring—a discrete control that lowers buy overhead at the cost of a fixed outlay—by deriving a simple break–even condition in terms of projected future buy notional.

Contribution: I make three contributions

1. A precise formalization of the minigame's mechanics as a stochastic control problem, together with an audited simulator that matches the price–update and friction code paths and yields a dynamic programming upper bound under full information.

2. Derivation and calibration of cost–aware trading bands from an Ornstein–Uhlenbeck approximation, refined by incorporating discounting and first–passage (hitting–time) behavior to set entry/exit thresholds quantitatively.
3. A deployable POMDP trading system that couples particle–filtered state estimation with band policies, extends to a budgeted multiasset allocator and broker–hiring rule, and is benchmarked against the upper bound to quantify the gap to clairvoyant optimality.

Beyond the immediate application, the study illustrates a reproducible path from mechanics to deployed strategy: extract dynamics, pose the control problem, build a faithful simulator, derive transparent baselines, introduce state–space inference, and evaluate against clear upper bounds with costs and inventory fully accounted for. These steps mirror quantitative research practice and demonstrate techniques —-state–space modeling, hidden–regime inference, dynamic programming, and cost–aware execution —- that transfer beyond the game.

2 Stock-Market Mechanics

This section formalizes every mechanic of the game that affects the Cookie Clicker stock market. I adopt a notation that mirrors the source code so that each line below maps to a concrete implementation detail.

2.1 State, Timing, and Units

Per-asset state. For each good $g \in \{1, \dots, 17\}$ and tick $t \in \mathbb{N}$ (one tick = 60 seconds in real time), I track:

$$\begin{aligned} v_t^{(g)} &\in [1, \infty) && \text{price in } \textit{\$econds}, \\ d_t^{(g)} &\in \mathbb{R} && \text{drift}, \\ m_t^{(g)} &\in \{0,1,2,3,4,5\} && \text{mode}, \\ \tau_t^{(g)} &\in \mathbb{N} && \text{mode time-to-live.} \end{aligned}$$

Inventory is $q_t^{(g)} \in \{0, \dots, q_{\max}^{(g)}\}$ (no shorting). I also retain a last-trade flag $\ell_t^{(g)} \in \{0, 1, 2\}$ (none/bought/sold) to enforce the no buy-and-sell-in-same-tick rule.

Currency and conversion. A *\$econd* is one second of the player's highest raw cookies-per-second (CpS). Converting between *\$econds* and cookies uses

$$\text{cookies} = v \times \text{CpS}_{\max}.$$

Resting value. Each good has a deterministic resting value

$$r^{(g)} \;=\; 10 + 10 \cdot (g - 1) + (\text{BankLevel} - 1),$$

so goods later in the list have higher baselines; bank level shifts all baselines equally.

Activation. Good g becomes *active* (tradable and visible) once the corresponding building has ever been owned (`building.highest` > 0). Before activation, g is hidden.

2.2 Mode Process and Durations

Each good occupies a mode $m_t^{(g)} \in \{0, \ldots, 5\}$ with semantics

m	name (for reference)
0	stable
1	slow rise
2	slow fall
3	fast rise
4	fast fall
5	chaotic

A countdown $\tau_t^{(g)}$ decrements by one each tick; when $\tau_t^{(g)} \le 0$, a new $\tau_{t+1}^{(g)}$ is drawn and the mode may switch:

$$\tau_{t+1}^{(g)} \sim \lfloor 10 + U \cdot (690 - 200\,\zeta) \rfloor, \qquad U \sim \mathrm{Unif}(0,1),$$

where $\zeta = \text{auraMult}(\text{“Supreme Intellect”}) \ge 0$ (if the aura is inactive, $\zeta = 0$). Conditional on reset, the next mode is

$$\mathcal{E}_t \;\coloneqq\; \{U_1 < \zeta,\ U_2 < 0.5\} \;\text{ or }\; \{U_3 < 0.7,\ m_t^{(g)} \in \{3,4\}\}.$$

$$m_{t+1}^{(g)} \sim \begin{cases} 5, & \text{if } \mathcal{E}_t, \\ \mathrm{Cat}\,[0{:}1,\ 1{:}2,\ 2{:}2,\ 3{:}1,\ 4{:}1,\ 5{:}1], & \text{otherwise.} \end{cases}$$

i.e., base weights $(1, 2, 2, 1, 1, 1)/8$ with extra propensity toward chaotic mode when the aura is strong or when leaving a fast-rise/fall regime.

2.3 Drift Update (intra-mode)

Let $U(\cdot) \sim \mathrm{Unif}(0,1)$, and write $\tilde{U} \equiv U - 1/2$ for centered uniforms. Per tick, the drift is first shrunk and nudged in a mode-specific way:

$$d_{t+\frac{1}{2}}^{(g)} \;=\; \alpha(\zeta, m_t)\, d_t^{(g)} \;+\; \Delta_d(\zeta, m_t),$$

with

$$\alpha(\zeta, m) = \begin{cases} 0.97 + 0.01\zeta & \text{all } m \end{cases} \quad \text{then} \quad \Delta_d(\zeta, m) = \begin{cases} 0.05\,(U - 0.5) & m = 0 \\ 0.05\,(U - 0.1) & m = 1 \quad \text{(upward bias)} \\ -0.05\,(U - 0.1) & m = 2 \quad \text{(downward bias)} \\ 0.15\,(U - 0.1) & m = 3 \\ -0.15\,(U - 0.1) & m = 4 \\ 0 & m = 5 \end{cases}$$

Additional stochastic kicks to drift:

$$d_{t+\frac{2}{3}}^{(g)} \;=\; d_{t+\frac{1}{2}}^{(g)} \;+\; 0.1\,\tilde{U}_1 \;+\; \mathbb{I}\{U_2 < 0.1\} \cdot \tilde{U}_3 \cdot (0.3 + 0.2\zeta).$$

In chaotic mode ($m = 5$), with probability 0.2 the drift is *reset*:

$$d_{t+\frac{3}{4}}^{(g)} \;=\; \begin{cases} \tilde{U}_4 \cdot (2 + 6\zeta), & \text{with prob. } 0.2 \\ d_{t+\frac{2}{3}}^{(g)}, & \text{otherwise.} \end{cases}$$

When in fast regimes, there are extra small perturbations:

$$\text{if } m = 3 \text{ and } U < 0.3: \quad d \leftarrow d + 0.1\,\tilde{U}, \qquad \text{if } m = 4 \text{ and } U < 0.3: \quad d \leftarrow d + 0.1\,\tilde{U}.$$

Finally, a high-price damping limits runaway drift:

$$\text{if } v_t^{(g)} > 100 + 3 \cdot (\text{BankLevel} - 1) \text{ and } d > 0, \quad d \leftarrow 0.9\,d.$$

2.4 Price Update (per tick)

The price evolves by a mean-reverting step toward the resting value, plus drift, plus several noise components and occasional global shocks. Define a heavy-tailed micro-kick

$$\eta \;\equiv\; 3 \cdot \operatorname{sign}(\tilde{U}) \cdot |\tilde{U}|^{11},$$

and independent uniforms U_i. Then, before clamps,

$$\begin{aligned}
\underbrace{v_{t+\frac{1}{3}}^{(g)}}_{\text{reversion}} &= v_t^{(g)} + 0.01\big(r^{(g)} - v_t^{(g)}\big),\\
\underbrace{v_{t+\frac{2}{3}}^{(g)}}_{\text{local noise}} &= v_{t+\frac{1}{3}}^{(g)} + \eta + \mathbb{I}\{U_1 < 0.15\} \cdot 3\,\tilde{U}_2 + \mathbb{I}\{U_3 < 0.03\} \cdot (10 + 10\zeta)\,\tilde{U}_4\\
&\quad + \mathbb{I}\{m_t^{(g)} = 3 \wedge U_5 < 0.3\} \cdot 10\,(U_6 - 0.7) + \mathbb{I}\{m_t^{(g)} = 4 \wedge U_7 < 0.3\} \cdot 10\,(U_8 - 0.3)\\
&\quad + \mathbb{I}\{m_t^{(g)} = 5 \wedge U_9 < 0.5\} \cdot 10\,\tilde{U}_{10},\\
\underbrace{v_{t+1^-}^{(g)}}_{\text{drift addition}} &= v_{t+\frac{2}{3}}^{(g)} + d_{t+\frac{3}{4}}^{(g)}.
\end{aligned}$$

Global shock (cross-asset). Per tick, with probability $0.1 + 0.1\zeta$ a global factor $\Delta \sim \text{Unif}(-1, 1)$ is drawn; with independent probability U_g per asset (i.e., a shared threshold $U^* \sim \text{Unif}(0, 1)$ and the condition $U_g < U^*$), each affected asset receives an additive hit:

$$v_{t+1^-}^{(g)} \leftarrow v_{t+1^-}^{(g)} - \Delta \cdot \Big\{\big[1 + d \cdot U_a^3 \cdot 7\big] + \big[1 + U_b^3 \cdot 7\big]\Big\}, \qquad d_{t+1^-}^{(g)} \leftarrow d_{t+1^-}^{(g)} + \Delta \cdot (1 + 4U_c),$$

and the current mode duration is forced to expire by setting $\tau_{t+1}^{(g)} = 0$.

Floors and clamps. After all additive terms,

$$\text{if } v < 5: \quad v \leftarrow v + 0.5\,(5 - v), \qquad \text{if } v < 5 \;\&\; d < 0: \quad d \leftarrow 0.95\,d, \qquad v \leftarrow \max\{v,\, 1\}.$$

These enforce a soft floor near 5 and a hard floor at 1.

Last-step bookkeeping. The displayed percent change uses only the last two stored values:

$$\Delta^{\%} v_t^{(g)} \;=\; 100 \times \left(\frac{v_t^{(g)}}{v_{t-1}^{(g)}} - 1\right),$$

and the price history buffer keeps the 65 most recent values for charting.

2.5 Trading Mechanics and Costs

Feasible actions and one-tick lock. At tick t, for asset g, the agent may choose buy-n, sell-n, or hold, subject to:

$$0 \le n \le q_{\max}^{(g)} - q_t^{(g)} \text{ (buy)}, \qquad 0 \le n \le q_t^{(g)} \text{ (sell)}, \qquad \ell_t^{(g)} \ne 1 \text{ for sell, } \ell_t^{(g)} \ne 2 \text{ for buy.}$$

The last condition enforces "no buy and sell of the same stock within the same tick."

Overhead on buys and brokers. Let b be the number of hired brokers. The *proportional buy overhead* is

$$\kappa(b) \;=\; 0.2 \cdot 0.95^{\,b}, \qquad \text{overhead factor } F(b) = 1 + \kappa(b).$$

Hiring broker $b{+}1$ costs 1200 *\$econds* (i.e., 20 minutes of $\mathrm{CpS}_{\max}$) and is allowed up to

$$b_{\max} \;=\; \left\lceil \frac{\text{Grandma.highest}}{10} + \text{Grandma.level} \right\rceil .$$

Cash, cost, and proceeds. Given available cookies C_t and $\mathrm{CpS}_{\max}$,

$$\text{buy cost in cookies} \;=\; n\, v_t^{(g)}\, F(b)\, \mathrm{CpS}_{\max}, \qquad \text{sell proceeds in cookies} \;=\; n\, v_t^{(g)}\, \mathrm{CpS}_{\max}.$$

A budget constraint $C_t \ge$ cost must hold to execute a buy. Displayed "profit" maintains a book in *\$econds*:

$$\Pi_{t+1} \leftarrow \begin{cases} \Pi_t - n\, v_t^{(g)}\, F(b) & \text{on buy} \\ \Pi_t + n\, v_t^{(g)} & \text{on sell} \\ \Pi_t & \text{on hold.} \end{cases}$$

Inventory capacity. Maximum inventory per asset depends on building progress and office upgrades:

$$M(L) := \begin{cases} 1.5, & L > 4, \\ 1, & L \le 4, \end{cases}$$

$$B(L) := 25\nVdash\{L > 0\} + 50\nVdash\{L > 1\} + 75\nVdash\{L > 2\} + 100\nVdash\{L > 3\},$$

$$q_{\max}^{(g)} = \left\lceil \text{highest}^{(g)}\, M(L) \;+\; B(L) \;+\; 10\, \text{BuildingLevel}^{(g)} \right\rceil .$$

where the office-level additive bonus is cumulative

$$B(L) \;=\; 25 \cdot \mathbb{I}\{L > 0\} + 50 \cdot \mathbb{I}\{L > 1\} + 75 \cdot \mathbb{I}\{L > 2\} + 100 \cdot \mathbb{I}\{L > 3\}.$$

2.6 Loans (CpS Buff/Debuff; Budget Side Only)

Three loan types change CpS (not prices) and require an immediate downpayment of a fraction of the current cookie bank. Let loan j have parameters $(\mu_j,\ T_j^+,\ \mu_j^{\text{pay}},\ T_j^-,\ \delta_j)$ meaning:

on take: $\mathrm{CpS}{\times}\mu_j$ for T_j^+, then: $\mathrm{CpS}{\times}\mu_j^{\text{pay}}$ for T_j^-, and pay δ_j of current cookies immediately.

In game defaults:

name	μ	T^+	μ^{pay}	T^-	δ
modest	1.5	short	0.25	long	0.20
pawnshop	2.0	very short	0.10	medium	0.40
retirement	1.2	very long	0.80	very very long	0.50

Durations are measured in ticks internally; they alter cookie budget trajectories and thus trading capacity but do not enter the price kernel above.

2.7 Tick Cadence and Cheats

The market advances every 60 seconds by default. A developer-only toggle can set the tick to 0.1 seconds for testing; the dynamics remain identical aside from speed.

2.8 Summary: One-Line Kernel

Collecting the pieces (suppressing asset index g and clamps for brevity), the per-tick evolution is

$$\boxed{\begin{aligned} d_{t+1} &= \underbrace{(0.97+0.01\zeta)\,d_t + \Delta_d(m_t)}_{\text{mode shrink + bias}} + \underbrace{0.1\,\tilde{U} + \mathbb{I}\{U<0.1\}\tilde{U}\,(0.3+0.2\zeta)}_{\text{drift noise}} + \underbrace{\mathbb{I}\{m_t=5\}\,\text{reset}_{0.2}}_{\text{chaotic reset}} \\ v_{t+1} &= v_t + \underbrace{0.01\,(r-v_t)}_{\text{mean reversion}} + \underbrace{d_{t+1}}_{\text{drift}} + \underbrace{\eta + \mathbb{I}\{U<0.15\}3\tilde{U} + \mathbb{I}\{U<0.03\}(10+10\zeta)\tilde{U}}_{\text{local shocks}} + \underbrace{\text{global-shock}_{0.1+0.1\zeta}}_{\text{cross-asset}} \end{aligned}}$$

followed by the soft floor toward 5 and the hard floor at 1, mode-duration decrement and possible mode resampling, and enforcement of the trade lock ℓ_t. Every trading and capacity rule in the previous subsections applies to this kernel.

3 Stochastic Models

This section introduces tractable stochastic models that approximate the code-faithful kernel in Section 2. The aim is twofold: (i) obtain calibrated dynamics that reproduce empirical behavior under realistic time and cost scales; (ii) expose conditional structure that later admits control policies with transparent performance bounds.

3.1 Markov State and Factorization

For each good g, define the deviation from resting value $x_t^{(g)} := v_t^{(g)} - r^{(g)}$. A compact latent state is

$$S_t^{(g)} \;=\; \big(x_t^{(g)},\, d_t^{(g)},\, m_t^{(g)}\big),$$

with the duration counter absorbed into the transition. Ignoring clamps for notation, the one–step kernel factorizes as

$$\mathbb{P}\Big(S_{t+1}^{(g)} \mid S_t^{(g)}\Big) = \mathbb{P}\Big(m_{t+1}^{(g)} \mid m_t^{(g)}\Big)\; \mathbb{P}\Big(d_{t+1}^{(g)} \mid d_t^{(g)}, m_{t+1}^{(g)}\Big)\; \mathbb{P}\Big(x_{t+1}^{(g)} \mid x_t^{(g)}, d_{t+1}^{(g)}, m_{t+1}^{(g)}\Big)\,.$$

A cross–asset global shock acts as a sparse jump factor G_{t+1} shared across goods; conditional on G_{t+1}, assets are independent.

3.2 Single–Regime OU (SR–OU) Baseline

The SR–OU baseline treats deviations as a discrete–time OU/AR(1) with rare jumps:

$$x_{t+1}^{(g)} \;=\; \phi^{(g)}\, x_t^{(g)} \;+\; \mu^{(g)} \;+\; \sigma^{(g)}\varepsilon_{t+1}^{(g)} \;+\; J_{t+1}\, A_{t+1}^{(g)},$$

where

$$\varepsilon_{t+1}^{(g)} \sim \text{Student-}t_\nu, \qquad J_{t+1} \sim \text{Bern}(p_J), \qquad A_{t+1}^{(g)} \sim \mathcal{N}(0, \sigma_J^2).$$

Here $\phi^{(g)} \in (0,1)$ captures mean reversion (the code's 1% pull per tick implies $\phi \approx 0.99$), $\mu^{(g)}$ is a small bias induced by asymmetric kicks, the Student-t innovation models heavy tails, and the jump mixture (p_J, σ_J) approximates occasional large moves (including the global disturbance). Stationary variance is

$$\mathrm{Var}(x^{(g)}) \;=\; \frac{\sigma^2 \, \frac{\nu}{\nu-2} \;+\; p_J \, \sigma_J^2}{1-(\phi^{(g)})^2} \qquad (\nu > 2).$$

Calibration. Estimate $\phi^{(g)}$ and $\mu^{(g)}$ by robust AR(1) regression on x_{t+1} versus x_t using Huber loss; identify jump times by median–absolute–deviation (MAD) residual screening; refit σ on nonjump residuals; set (p_J, σ_J) by frequency and variance of jump residuals; choose ν by maximizing the Student–t likelihood (or by a grid over $\nu \in [3, 10]$). A continuous–time mapping is $\kappa^{(g)} := -\log \phi^{(g)}$ and $\theta^{(g)} := -\mu^{(g)}/(1-\phi^{(g)})$, giving $dx_t = -\kappa(x_t - \theta)\,dt + \tilde{\sigma}\,dW_t$ when jumps are off.

3.3 Regime–Switching OU HMM (RSO–HMM)

To reflect coded regimes, let $m_t^{(g)} \in \{1, \ldots, K\}$ be hidden modes (e.g., $K = 5$ or 6) with transition matrix Π. Conditional on $m_{t+1}^{(g)} = k$,

$$x_{t+1}^{(g)} \;=\; \phi_k \, x_t^{(g)} \;+\; \mu_k \;+\; \sigma_k \varepsilon_{t+1}^{(g)} \;+\; J_{t+1} \, A_{t+1}^{(g)}, \qquad \varepsilon_{t+1}^{(g)} \sim \text{Student-}t_{\nu_k}.$$

The $(\phi_k, \mu_k, \sigma_k, \nu_k)$ capture stable/slow/fast/chaotic behavior; Π encodes persistence and the extra propensity into a "chaotic" mode.

Filtering (forward pass). With $\alpha_t(j) = \Pr(m_t = j \mid \mathcal{F}_t)$ and emission density $f_k(x_{t+1} \mid x_t) = \sum_{j \in \{0,1\}} \Pr(J_{t+1}{=}j)\, f_{k,j}(x_{t+1} \mid x_t)$, the update is

$$\tilde{\alpha}_{t+1}(k) \;=\; f_k(x_{t+1} \mid x_t) \sum_{j=1}^{K} \alpha_t(j)\, \Pi_{jk}, \qquad \alpha_{t+1}(k) \;=\; \frac{\tilde{\alpha}_{t+1}(k)}{\sum_{\ell=1}^{K} \tilde{\alpha}_{t+1}(\ell)}.$$

The one–step predictive mean and variance used later by controllers are

$$\hat{x}_{t+1|t} \;=\; \sum_k \alpha_t(k)\,(\phi_k x_t + \mu_k), \qquad \widehat{\mathrm{Var}}_t \;=\; \sum_k \alpha_t(k)\big(\sigma_k^2 + (\phi_k x_t + \mu_k - \hat{x}_{t+1|t})^2\big).$$

Estimation (EM). Baum–Welch with t–emissions: E–step uses the forward–backward algorithm to compute expected state occupancies and transitions; M–step updates Π by normalization and $(\phi_k, \mu_k, \sigma_k, \nu_k)$ by t–regression closed forms (or ECM). Initialize by clustering residuals from the SR–OU fit and mapping clusters to modes.

3.4 Drift–Explicit Switching State Space (IMM/Kalman)

An alternative keeps a continuous drift d_t in the latent state and conditions its evolution on the mode:

$$\begin{aligned} d_{t+1} \mid (m_{t+1} = k, d_t) \;&=\; a_k \, d_t + b_k + \eta_{t+1}^{(k)}, \quad \eta_{t+1}^{(k)} \sim \mathcal{N}(0, \sigma_{d,k}^2), \\ x_{t+1} \mid (m_{t+1} = k, d_{t+1}, x_t) \;&=\; \phi \, x_t + d_{t+1} + \xi_{t+1}^{(k)}, \quad \xi_{t+1}^{(k)} \sim \mathcal{N}(0, \sigma_{x,k}^2), \end{aligned}$$

optionally with an additive jump term as in the SR–OU. Because the model is conditionally linear–Gaussian given the mode, an interacting multiple model (IMM) filter applies:

1. **Mixing:** form mixed initial $(\bar{\mu}_t^{(k)}, \bar{P}_t^{(k)})$ from prior mode weights α_t and covariances.
2. **Mode–conditioned Kalman step:** run a Kalman predict/update for each k.
3. **Mode probability update:** update $\alpha_{t+1}(k)$ by the mode–specific likelihoods.

This captures the code's "drift memory" while preserving fast inference.

3.5 Global Shock Factor (Cross–Asset)

Introduce a latent jump indicator $G_{t+1} \sim \mathrm{Bern}(p_G)$ and a common amplitude $Z_{t+1} \sim \mathcal{N}(0, \sigma_G^2)$. Conditional on $G_{t+1} = 1$, each asset is hit with probability U^* and magnitude proportional to $|Z_{t+1}|$. A likelihood for the cross–section at $t{+}1$ is then a two–component mixture (no–shock vs. shock) whose E–step infers p_G and assigns responsibilities; M–step updates σ_G and the per–asset hit probability U^*. This factor explains occasional synchronized moves and improves regime estimates.

3.6 Model Selection and Pooling Across Assets

Calibrate each good separately for (ϕ, μ, σ, ν) and pool Π (and optionally chaotic–mode parameters) across goods to stabilize estimates. Information criteria (AIC/BIC) on held–out segments choose between SR–OU and RSO–HMM; a likelihood–ratio test with parametric bootstrap compares K vs. $K{-}1$ modes. In practice, SR–OU often suffices for control design, while RSO–HMM yields better filters in volatile phases.

3.7 Outputs Needed by Controllers

Subsequent sections only require the following summaries per tick and asset:

$$\hat{x}_{t+1|t}, \quad \widehat{\mathrm{Var}}_t, \quad \alpha_t(k) \text{ for } k = 1, \dots, K, \quad \text{and (optionally) } \hat{d}_{t+1|t}.$$

These feed cost–aware entry/exit band calculations and multi–asset budgeted allocation under inventory and cash constraints.

4 Optimization and Control

This section develops implementable controllers and a benchmark under full information. The objective is discounted profit in *\$econds*, subject to budget and inventory constraints. Let $\gamma \in (0, 1)$ denote a per–tick discount.

4.1 Objective and State for Control

For one good g, let state $s_t^{(g)} = (x_t^{(g)}, d_t^{(g)}, m_t^{(g)}, \tau_t^{(g)}, q_t^{(g)}, C_t)$. Actions are $a_t^{(g)} \in \{\text{buy } n, \text{ sell } n, \text{ hold}\}$ with feasibility as in Section 2. Instantaneous reward (in *\$econds*) is

$$R_t^{(g)} = \begin{cases} -\, n\, v_t^{(g)}\, F(b), & \text{buy } n, \\ \;\; n\, v_t^{(g)}, & \text{sell } n, \\ 0, & \text{hold}. \end{cases}$$

The multi–good objective is $\mathbb{E}\big[\sum_{t \ge 0} \gamma^t \sum_g R_t^{(g)}\big]$ under the market kernel of Section 2.

4.2 Fully Observed MDP Benchmark and Band Structure

Assume full knowledge of (x_t, d_t, m_t, τ_t) and ignore global shocks for the moment (they can be added as an exogenous jump). For a single unit $n = 1$, finite–horizon dynamic programming yields value

$$V_t(x, d, m, \tau, q, C) \;=\; \max_{a \in \mathcal{A}(q,C)} \Big\{ R_t(a) + \gamma\, \mathbb{E}\big[V_{t+1}(\cdot) \mid x, d, m, \tau, a\big] \Big\}.$$

Because R_t is linear in $v_t = r + x_t$ and the next–state distribution under the kernel is monotone in x_t, the optimal policy is of *band* type: there exist thresholds $\ell_t(d, m, \tau, q, C)$ and $u_t(d, m, \tau, q, C)$ with

$$\text{buy if } x_t \le \ell_t, \qquad \text{sell if } x_t \ge u_t, \qquad \text{hold otherwise.}$$

With integer inventory and budget, the optimal n fills up to capacity or liquidates, so the bands lift to multi–unit *(s,S)*–type controls. This benchmark is used to quantify the gap between implementable controllers and a clairvoyant policy.

4.3 OU Closed–Form Bands (Single Good, Myopic Cycle)

Consider the SR–OU approximation $x_{t+1} = \phi x_t + \mu + \sigma \varepsilon_{t+1}$ with $\phi \in (0, 1)$, buy overhead factor $F(b) = 1 + \kappa$, and no sell cost. Adopt a buy–low/sell–high cycle with thresholds $\ell < 0 < u$: buy one unit at $x_t = \ell$; sell the unit at the first time T_u when $x_{t+T_u} \ge u$. Approximate $\mathbb{E}[T_u \mid x_t = \ell]$ by the deterministic mean path,

$$\hat{T}(\ell \to u) \;\approx\; \left\lceil \frac{\log\big((u - \bar{x})/(\ell - \bar{x})\big)}{-\log \phi} \right\rceil, \qquad \bar{x} := \frac{\mu}{1 - \phi}.$$

The expected discounted profit per cycle is then

$$\Pi(\ell, u) \;\approx\; (r + u)\, \gamma^{\hat{T}(\ell \to u)} \;-\; F(b)\, (r + \ell),$$

and the per–tick discounted *rate* is $\Pi(\ell, u)/(\sum_{h=0}^{\hat{T}-1} \gamma^h)$. Maximizing $\Pi(\ell, u)$ over $\ell < 0 < u$ gives a closed–form *pair*:

$$u^\star \;=\; \bar{x} + \alpha\, \sigma, \qquad \ell^\star \;=\; \bar{x} - \beta\, \sigma,$$

with $\alpha, \beta > 0$ depending on $(\phi, \gamma, \kappa, r)$. A convenient calibration is to choose α, β to solve the two first–order conditions

$$\partial_u \Pi(\ell, u) = 0, \qquad \partial_\ell \Pi(\ell, u) = 0,$$

where $\hat{T}$ is treated as a smooth function of (ℓ, u) via log-interpolation. This yields explicit numerics per asset and updates online as (ϕ, μ, σ) are re–estimated. Two practical adjustments: (i) cap $|\ell|, |u|$ to respect the soft floor at $v \simeq 5$; (ii) inflate κ mildly to reflect jump risk (conservative bands).

Interpretation. Only buys pay overhead, so $|\ell^\star|$ is typically larger than $u^\star$, producing an asymmetric band that waits for deep dips but takes profit sooner. The term $(1 - F)r$ inside $\Pi(\ell, u)$ penalizes high–baseline goods, favoring low–baseline goods when capacity is scarce.

4.4 POMDP Controller via Particle–Filtered Bands

Under partial observability, maintain N particles for (d, m) (and optionally the global–shock flag). At tick t, compute posterior summaries:

$$\hat{x}_{t+1|t}, \quad \widehat{\mathrm{Var}}_t, \quad \alpha_t(k).$$

Map these to $(\hat{\phi}_t, \hat{\mu}_t, \hat{\sigma}_t)$ using local linear regression on recent (x_{t-h}, x_{t-h+1}), weighted by $\alpha_t(k)$. Form bands (ℓ_t, u_t) by plugging $(\hat{\phi}_t, \hat{\mu}_t, \hat{\sigma}_t)$ and $F(b)$ into the OU formulas above. Execute:

$$\text{buy if } x_t \le \ell_t, \quad \text{sell if } x_t \ge u_t, \quad \text{hold otherwise,}$$

subject to inventory, budget, and the one–tick lock. This "particle–filtered bands" controller inherits the band structure while adapting to regime shifts through the filter.

4.5 Multi–Good Budget: Greedy Index Allocation

When several goods signal "buy" at the same tick, allocate under a cookie budget C_t. For each good g, define the expected *edge per unit cost*

$$\mathrm{Idx}_t^{(g)} \;=\; \frac{\mathbb{E}[\text{cycle profit per unit} \mid \text{posterior}]}{\text{buy cost per unit in cookies}} \;=\; \frac{(r^{(g)} + u_t^{(g)})\,\gamma^{\hat{T}^{(g)}} - F(b)\,(r^{(g)} + \ell_t^{(g)})}{F(b)\,(r^{(g)} + \ell_t^{(g)})\ \mathrm{CpS}_{\max}}.$$

Rank goods by $\mathrm{Idx}_t^{(g)}$ and buy greedily, filling each up to $\min\{q_{\max}^{(g)} - q_t^{(g)},\ \lfloor C_t/\text{unit cost} \rfloor\}$ until C_t is exhausted. This knapsack–like rule is admissible and near–optimal when goods are independent and integer lots are small.

4.6 Broker Hiring Economics

The buy overhead is $\kappa(b) = 0.2 \cdot 0.95^{\,b}$. Hiring the $(b{+}1)$th broker reduces overhead by

$$\Delta\kappa(b) \;=\; \kappa(b) - \kappa(b{+}1) \;=\; 0.01 \cdot 0.95^{\,b}.$$

Let W^+ be the expected *discounted future buy notional* across all goods (in *\$econds*). The present value of savings is $\Delta\kappa(b)\,W^+$. The broker costs 1200 *\$econds*. A break–even rule is therefore

$$\Delta\kappa(b)\,W^+ \;\ge\; 1200 \quad\Longleftrightarrow\quad W^+ \;\ge\; \frac{1200}{0.01\,0.95^{\,b}}.$$

Estimate W^+ online by summing planned buys from the index policy, discounted by γ, over a rolling horizon. Cap by the broker limit $b_{\max}$ from Section 2.

4.7 Algorithm Summary (Per Tick)

1. **Filter update:** ingest latest prices; update particle weights and mode posteriors $\alpha_t(k)$; compute $\hat{x}_{t+1|t}$ and $\widehat{\mathrm{Var}}_t$ for each good.
2. **Local model fit:** map posteriors to $(\hat{\phi}_t, \hat{\mu}_t, \hat{\sigma}_t)$ per good.
3. **Band computation:** form (ℓ_t, u_t) from OU band formulas using $F(b)$.
4. **Signal set:** collect goods with $x_t \le \ell_t$ (buy) and $x_t \ge u_t$ (sell), enforcing the one–tick lock.

5. **Sell first:** execute sells (freeing cash).
6. **Budgeted buys:** compute $\mathrm{Idx}_t^{(g)}$; allocate greedily under C_t and capacities.
7. **Broker check (daily or weekly):** estimate W^+; hire if the break–even rule holds and $b < b_{\max}$.

4.8 Notes on the Benchmark Gap

The fully observed MDP with the true kernel gives an upper bound on implementable performance. The particle–filtered band controller is suboptimal due to partial information and band approximations, but is computation–light and robust to costs and locks. Empirically, the gap shrinks when regimes are persistent (RSO–HMM filtering is accurate) and when jumps are rare (OU bands align with the kernel's local behavior).

5 Implementation Details: Tick Timing, Constraints, and Numerics

5.1 Tick Cadence and Update Order

One market tick equals 60 real seconds. A developer toggle can accelerate to 0.1 s; the kernel is unchanged. Each tick applies the following order per the source:

1. **Per–good prelude:** set last–trade flag to "none"; ensure activation if the building has ever been owned.
2. **Drift shrink and mode bias:** multiply d_t by $0.97 + 0.01\zeta$, add the mode–specific bias/noise.
3. **Price reversion:** $v \leftarrow v + 0.01\,(r - v)$.
4. **Local shocks:** add heavy-tailed micro-kick, occasional small kicks, and mode-conditional kicks.
5. **Chaotic adjustments:** in mode 5, with prob. 0.5 add a large value kick; with prob. 0.2 reset d.
6. **Add drift:** $v \leftarrow v + d$.
7. **Global shock (cross-asset):** with prob. $0.1 + 0.1\zeta$, draw $\Delta \sim \mathrm{Unif}(-1, 1)$, draw a shared threshold $U^* \sim \mathrm{Unif}(0, 1)$, and apply the shock to goods with $U_g < U^*$. For affected goods, set mode duration to expire ($\tau \leftarrow 0$).
8. **Floors and clamps:** soft pull toward $v = 5$ when $v < 5$; hard floor $v \geq 1$; damp d if $v > 100 + 3 \cdot (\mathrm{BankLevel} - 1)$ and $d > 0$.
9. **History:** push new v to the front of the length–65 buffer used for charts and percent change.
10. **Mode duration and resample:** decrement τ; if $\tau \leq 0$ draw a new τ and sample the next mode using the base weights with chaotic preference rules.

5.2 Action Window and Constraints

At the end of the tick, evaluate actions subject to:

- **One–tick lock:** cannot both buy and sell the same good within the same tick; enforce via last–trade flag.
- **Capacity:** $0 \leq q_t^{(g)} \leq q_{\max}^{(g)}$ with $q_{\max}^{(g)}$ from Section 2.
- **Budget:** buy cost per unit (in cookies) equals $v_t^{(g)} F(b)\,\mathrm{CpS}_{\max}$; sells earn $v_t^{(g)}\mathrm{CpS}_{\max}$.
- **No shorting:** inventory cannot be negative.
- **Loans:** loans modify CpS and require an immediate downpayment; price dynamics are unaffected.

Execute *sells first* (to free cash), then *budgeted buys* via the index allocator.

5.3 RNG and Reproducibility

Use a single 64-bit seed to initialize independent streams:

1. One stream for per–good idiosyncratic draws (advance deterministically by a fixed amount per good per tick).
2. One stream for the global-shock draw and the shared threshold U^*.
3. One stream for mode resampling and durations.

This yields path determinism across platforms. Store the seed with each simulation artifact.

5.4 Numerical Settings (Recommended Defaults)

Double precision for all stochastic updates and value aggregates. All internal computations in *\$econds*; convert to cookies only at trade execution. Defaults:

Discount	γ	0.999
OU fit window	H	120 ticks (2 hours)
OU ridge	λ	10^{-4} (stabilize ϕ)
Vol estimator		MAD$\times$1.4826 on residuals
Particles	N	512 (systematic resampling at ESS/N < 0.5)
Band solver steps	K	12 iterations; step $\eta_1 = \eta_2 = 0.3$
Band guardrails		$\lvert u - \ell \rvert \geq 0.25\,\sigma,\ \ell \geq 1 - r$
Index refresh		every tick after sells; greedy allocation
Broker check		every 60 ticks (1 hour, real time)

5.5 Stable OU Estimation

Form $(\hat{\phi}, \hat{\mu})$ by OLS on (x_{t-1}, x_t) over the last H points, with ridge λ. Compute residuals $\widehat{\varepsilon}_t = x_t - \hat{\phi} x_{t-1} - \hat{\mu}$. Estimate $\hat{\sigma} = \mathrm{MAD}(\widehat{\varepsilon}) \times 1.4826$. Optionally weight by mode posteriors to down–weight suspected chaotic ticks.

5.6 Particle Filter Details

State per particle: (d, m) (and optionally a latent global-shock flag). Prediction: propagate d using the mode-specific drift kernel; sample m and duration when needed. Update: likelihood from the price increment $x_t - x_{t-1}$ under the SR–OU approximation with parameters mapped from the particle (e.g. mode-dependent variance inflation). Resample when ESS falls below half the particles; add small jitter to d to avoid degeneracy.

5.7 Band Computation and Fallbacks

Compute $\widehat{T}$ via $\widehat{T} = \lceil \log\left((u - \bar{x})/(\ell - \bar{x})\right)/(-\log\phi) \rceil$. Use $\gamma^{\widehat{T}} = \exp(\widehat{T} \cdot \ln\gamma)$ for stability. If the fixed-point solver fails to contract in K steps:

1. Inflate κ by 10% (conservative buy) and retry.
2. Fall back to symmetric bands $\bar{x} \pm 1.5\,\hat{\sigma}$ capped by floors and capacity economics.

5.8 Budgeted Multi–Good Allocation

Compute $\mathrm{Idx}^{(g)}$ as in Section 4, rank descending, and allocate greedily subject to (i) remaining budget, (ii) per–good capacity margin, (iii) one–tick lock. Sorting cost is $O(G \log G)$. Execute integer lots; remainders are ignored to avoid fractional noise.

5.9 Broker Hiring Rule (Implementation)

Maintain an exponentially weighted estimate of discounted future buy notional W^{+}:

$$W_t^{+} = \rho\, W_{t-1}^{+} + (1-\rho)\,(\text{planned buy notional at } t), \quad \rho = 0.99.$$

If $\Delta\kappa(b)\, W_t^{+} \geq 1200$ and $b < b_{\max}$, hire one broker. Throttle the decision to at most once per 60 ticks.

5.10 Testing and Validation

- **Kernel parity:** with a fixed seed, match time–series moments of v_t and the empirical distribution of $\Delta^{\%} v_t$ against a reference run of the original game logic.
- **Shock coverage:** verify global-shock frequency $\approx 0.1{+}0.1\zeta$ and the cross-asset hit distribution.
- **Constraints:** unit tests for capacity, cost, and lock enforcement, including edge cases at the soft floor.
- **Numerics:** invariance to float order (Kahan summation for long profit series), and stability of γ^{T}.

6 Experimental Design (Pre-Registered)

This section fixes the hypotheses, environments, baselines, metrics, analysis protocol, and reporting templates *prior* to implementation. The goals are (i) to make the evaluation replicable and decision-relevant, and (ii) to reduce analytic flexibility once results are observed.

6.1 Objectives and Hypotheses

I will evaluate a family of mean-reversion controllers under transaction costs and capacity constraints.

Primary objectives.

O1. Quantify the performance of cost-aware Ornstein–Uhlenbeck (OU) band policies relative to simple baselines.

O2. Measure the gap between implementable *belief-based* control and a full-information dynamic-programming (DP) upper bound.

O3. Assess capital-allocation efficiency across multiple goods under a shared budget.

Hypotheses.

H1. **H1** (OU gains): OU bands with proportional buy overhead achieve higher discounted profit than No-Trade and naive thresholds, with strictly positive paired improvement.

H2. **H2** (Belief closes gap): Particle-filtered bands recover at least 30% of the full-information DP bound (fraction-of-bound metric; see §6.4).

H3. **H3** (Allocation helps): The budgeted index allocator dominates independent per-good controllers under the same cash budget.

H4. **H4** (Broker ROI): The broker-hiring rule yields nonnegative net present value (NPV) under projected buy notional; ROI increases in overhead and decreases in mean reversion.

6.2 Environments and Settings

I will use a code-faithful simulator matching the tick order of operations (drift update, mean reversion, local shocks, global shock, floors/clamps, bookkeeping). Unless stated, Aura *off* and fixed Bank Level.

Time and horizons. One tick corresponds to the game's minute update. I will simulate in discrete ticks.

- **Burn-in:** 2,000 ticks from a reset state to reach a stationary regime per good.
- **Evaluation horizons:** Short T_S = 1,440 ticks (1 day), Medium T_M = 4,320 ticks (3 days), Long T_L = 10,080 ticks (7 days).

Global configuration (default).

Bank level	= 1 (affects resting values uniformly)
Aura "Supreme Intellect"	off (on only in ablations)
Initial cookies (bank)	B_0 = 5,000 *$econds* (converted to cookies via $\text{CpS}_{\max}$)
Brokers at $t = 0$	$b_0 = 0$; hiring enabled per break-even rule
Broker cap	game-legal cap; if unavailable, set $b_{\max} = 10$ (ablation sweeps)
Capacity	per-good $q_{\max}$ as in §2 (fixed buildings snapshot)
Tick RNG	common random numbers across strategies (see below)

Randomness and common seeds. I will use disjoint, named RNG streams to enable paired comparisons:

- `rng-price-local`, `rng-price-global`, `rng-mode`, `rng-trade-ties`.
- Seed list for paired experiments (10 seeds): $\{101, 103, 107, 109, 113, 127, 131, 137, 139, 149\}$.

For each seed, all strategies experience identical price paths (common random numbers), enabling low-variance paired estimates.

Initialization per good. At the end of burn-in, set $q_0^{(g)} = 0$, last-trade flag "none", and record initial price $v_0^{(g)}$.

6.3 Baselines and Planned Methods

I pre-register the following methods.

Baselines.

B1. **No-Trade**: Hold zero inventory; serves as a floor benchmark.

B2. **Buy&Hold**: At $t = 0$, buy each active good to capacity (subject to budget), then hold.

B3. **Naive Threshold**: Buy when $v_t < r - k\sigma$, sell when $v_t > r + k\sigma$ with $k = 1$, σ estimated from a rolling 256-tick window of $|\Delta v|$.

B4. **FO-DP Upper Bound**: Full-information per-good dynamic program (no budget coupling), with costs and capacity; summed across goods. This is an *upper bound* for implementable controllers.

Planned methods (to be evaluated).

P1. **OU Bands (cost-aware)**: Closed-form entry/exit bands derived in §4 with proportional buy overhead, discount factor, and no shorting.

P2. **PF Bands (POMDP)**: Particle-filtered belief over latent drift/mode; apply belief-dependent bands; resample with systematic resampling.

P3. **Budgeted Index Allocation**: When multiple buys fire, allocate cash greedily by posterior edge-per-$econd index under capacity.

P4. **Broker Hiring Rule**: Myopic NPV-positive hiring based on projected discounted future buy notional.

6.4 Metrics and Estimands

Primary and secondary metrics are defined per seed and horizon; reported as paired differences vs. baselines and as absolute values.

Primary.

- **Discounted Profit** (in *\$econds*): $J_\gamma = \sum_{t=0}^{T-1} \gamma^t \, \Delta\Pi_t$, with $\gamma = \exp(-\ln 2/1440)$ (1-day half-life).
- **Undiscounted Profit**: $J_1 = \sum_{t=0}^{T-1} \Delta\Pi_t$.
- **Fraction of DP Bound**: $\phi = \dfrac{J_\gamma(\text{method}) - J_\gamma(\text{No-Trade})}{J_\gamma(\text{FO-DP}) - J_\gamma(\text{No-Trade})}$, clipped to $[0, 1]$ for reporting.

Secondary.

- **Turnover**: total buy notional (in *\$econds*) over horizon.
- **Average Inventory Utilization**: per-good mean of $q_t/q_{\max}$.
- **Average Holding Time**: per fill, mean ticks held.
- **Max Drawdown**: running-peak drawdown of cumulative Π_t.
- **Broker ROI**: discounted proceeds attributable to overhead reduction minus hiring costs.
- **Hit Rate**: fraction of round-trips with positive P&L.

Computational metrics. Wall-clock per tick, memory footprint, and effective particles (ESS) for PF methods.

6.5 Protocol and Statistical Analysis

Protocol. For each seed and horizon:

1. Simulate burn-in, then freeze initial conditions.
2. Run all strategies with common random numbers.
3. Compute metrics, serialize logs, and write a per-seed report.

Pairing and aggregation. For each metric M, compute paired differences $\Delta M = M_{\text{method}} - M_{\text{baseline}}$ per seed. Aggregate across seeds with the mean and 95% percentile bootstrap confidence intervals (10,000 bootstrap draws over seeds).

Decision rules (pre-registered).

- **Support H1**: If the lower 95% CI of ΔJ_γ vs. No-Trade is > 0 and the point estimate exceeds +5% of the DP gap.
- **Support H2**: If the median ϕ across seeds is ≥ 0.30 with lower 95% CI ≥ 0.20.
- **Support H3**: If the index allocator beats independent allocation by $\geq 3\%$ of the DP gap with lower 95% CI ≥ 0.
- **Support H4**: If broker ROI median ≥ 0 with lower 95% CI ≥ -0.02 (tolerating small estimation noise).

Multiple comparisons. When simultaneously testing several ablations, I will control the false discovery rate at 10% using Benjamini–Hochberg on the set of paired improvements for the *primary* metric only; secondary metrics are descriptive.

6.6 Ablations and Stress Tests

I will vary one factor at a time from the default environment.

Model/estimation.

- OU window length: $\{128, 256, 512, 1024\}$ ticks.
- Particle count: $\{100, 200, 500, 1000\}$; resampling threshold ESS $\in \{0.3, 0.5, 0.8\}$.
- Discount factor: half-life $\in \{12\text{h}, 1\text{d}, 3\text{d}\}$.

Market frictions.

- Initial overhead (no brokers): $\kappa_0 \in \{0.10, 0.20, 0.30\}$.
- Broker cap: $b_{\max} \in \{0, 3, 10\}$.
- Budget pressure: $B_0 \in \{2{,}000,\ 5{,}000,\ 20{,}000\}$ *\$econds*.

Game settings.

- Aura: off vs. on.
- Bank level: $\{1, 5, 10\}$.

Stress.

- Global-shock frequency scaled by factors $\{0.5, 1.0, 2.0\}$ (for robustness; not a game-canonical setting).
- Chaotic-mode dwell-time shortened by 25% (robustness).

6.7 Implementation Plan for Evaluation Artifacts

Logging schema (per tick).

Field	Description
`t`	tick index
`good_id`	asset identifier
`v_t`, `r_g`	price, resting value
`q_t`, `q_max`	inventory and capacity
`action`, `n_units`	buy/sell/hold and size
`cost`, `proceeds`	in cookies and in \$econds (both)
`overhead_factor`	$F(b) = 1 + \kappa(b)$ on buys
`posterior_summaries`	PF means/variances (if applicable)
`pi_cum`	cumulative profit in \$econds
`rng_state_hash`	hash of RNG stream states

File layout (planned). Per experiment: a YAML config, per-seed logs (`.parquet` or `.csv`), and an aggregated `.json` report with metrics and CIs. Each run is reproducible via a single CLI command taking the config path.

6.8 Figures, Tables, and Reporting Templates

I pre-commit to the following presentation:

- **Table 1**: Environment configurations (default and ablations).
- **Table 2**: Primary and secondary metrics (mean ± 95% CI) for all methods.
- **Figure 1**: Cumulative discounted profit trajectories (median across seeds; bands = 25–75th percentiles).
- **Figure 2**: Fraction-of-bound distribution (ϕ) as violin/box plots per method.
- **Figure 3**: Broker ROI vs. time with hiring events marked.
- **Figure 4**: Sensitivity heatmaps (OU window, particles) for J_γ.
- **Appendix plots**: Per-good P&L and inventory traces; runtime and ESS diagnostics.

6.9 Stopping, Failures, and Deviations

Stopping. For each configuration, run all planned seeds and horizons unless a runtime budget is exceeded; in that case, prioritize Short and Medium horizons for all methods rather than completing Long for a subset.

Failure handling. If a strategy violates feasibility (e.g., capacity, budget, or lock), the run is flagged invalid and excluded *for all methods* for that seed (to preserve pairing), with a documented reason.

Deviations. Any change to seeds, horizons, or decision rules after seeing results will be documented explicitly in an addendum.

6.10 Threats to Validity

Construct validity risks include mismatch between OU approximation and heavy-tailed shocks; mitigation: report DP bound fractions and ablations under increased shock frequency. Internal validity risks include coding errors; mitigation: unit tests for kernel parity and conserved quantities. External validity is limited to this minigame; the purpose is methodological, not predictive for real markets.

6.11 Computational Budget

Target wall-clock per tick for PF bands ≤ 1 ms on a modern laptop CPU, with memory < 1 GB for $N \leq 1000$ particles across 17 goods. If exceeded, reduce particles and/or reporting frequency; document any adjustments.

7 Implementation Plan

This section translates the modeling and evaluation design into a concrete build plan with milestones, module boundaries, tests, and deliverables. The goal is to produce a reproducible, code-faithful simulator and a set of controllers that can be evaluated through the pre-registered protocol in Section 6.

7.1 Tech Stack and Repository Layout

Languages and libraries. Python 3.11 (NumPy, SciPy, pandas), Numba for CPU jit, `pydantic` for config validation, `pyyaml`/TOML for experiment configs, `pytest`+`hypothesis` for testing, `matplotlib` for plots. Optional: JAX for vectorized filtering if profiling reveals bottlenecks.

Repo skeleton.

```
cookieclicker-autotrader/
  README.md
  LICENSE
  pyproject.toml               # build + deps
  src/
    ccstk/                     # package root
      __init__.py
      kernel/                  # code-faithful market kernel
        state.py               # dataclasses: GameState, GoodState, RNGState
        tick.py                # per-tick update (Section 2)
        rng.py                 # stream manager: local/global/mode
      models/                  # stochastic models (Section 3)
        ou.py                  # SR-OU fit, rolling estimators
        rso_hmm.py             # EM + forward/backward
        imm.py                 # IMM-Kalman for drift-explicit model
      control/                 # controllers (Section 4)
        bands.py               # OU band formulas + solver
        pf.py                  # particle filter (systematic resampling)
        index_alloc.py         # multi-good budgeted index allocator
        brokers.py             # hiring rule
      sim/
        run.py                 # simulator harness (single seed)
        cli.py                 # CLI entrypoint
        logging.py             # Parquet/CSV writers; RNG hashes
        metrics.py             # profit, bound fraction, etc.
      utils/
        io.py                  # config I/O, path helpers
        math.py                # numerics: stable exp/log, Kahan sum
  tests/
    test_kernel_parity.py
    test_constraints.py
    test_models_ou.py
    test_pf.py
    test_bands.py
    test_allocator.py
  experiments/
    configs/                  # YAML/TOML configs (pre-registered)
    scripts/                  # run_all.py, aggregate.py, plot.py
    artifacts/                # seeded outputs (ignored by git via .gitignore)
  docs/
    figures/                  # generated plots
    tables/
```

7.2 Milestones and Deliverables

1. **M0 – Build Scaffolding** (1–2 days): repo, CI, linting, formatting, base `GameState`/`GoodState`, deterministic RNG streams.
2. **M1 – Kernel** (2–4 days): implement tick order exactly as in Section 2; unit/property tests; golden traces for fixed seeds.
3. **M2 – Estimators** (2–4 days): SR–OU rolling fit; RSO–HMM EM (small K); emission likelihoods; convergence guards.
4. **M3 – Controllers** (3–5 days): OU band formulas + stable fixed-point solver; particle-filtered bands; one-tick lock logic.
5. **M4 – Multi-Asset Layer** (2–3 days): greedy index allocator; broker hiring rule; cash/capacity coupling.
6. **M5 – Harness & Metrics** (2–3 days): CLI, config schema, logging (per-tick schema), metrics/CI plots.
7. **M6 – Pre-Registered Runs** (2–3 days): default and ablations, seeds, aggregation, paper tables/figures.
8. **M7 – Packaging & Docs** (1–2 days): README with quickstart, usage, and reproducibility instructions.

7.3 Module Contracts and APIs

Kernel.

- `GoodState`: price v, drift d, mode m, duration τ, inventory q, last-trade flag ℓ, history buffer (length 65).
- `GameState`: list of goods, bank level, office level, brokers, $\text{CpS}_{\max}$, cookies C, discount γ, RNG streams.
- `tick(state)`: pure function returning the next `GameState`; applies the order of operations in Section 2.

Estimators (models).

- `ou.fit(x_hist, lambda, H)` $\to (\hat{\phi}, \hat{\mu}, \hat{\sigma})$.
- `rso_hmm.fit(x_hist, K)` $\to \hat{\Pi}$ and per-mode $(\phi_k, \mu_k, \sigma_k, \nu_k)$.
- `rso_hmm.filter(x_t, x_t-1)` $\to \alpha_t(k), \hat{x}_{t+1|t}, \widehat{\text{Var}}_t$.

Controllers.

- `bands.ou_solve(phi, mu, sigma, r, gamma, F)` $\to (\ell, u)$ with guardrails.
- `pf.step(particles, obs)` $\to$ updated particles and summaries; ESS-based resampling.
- `index_alloc.allocate(signals, costs, capacity, C)` $\to$ list of buy orders.
- `brokers.should_hire(b, Wplus)` $\to$ boolean (break-even rule).

Simulator and CLI.

- `sim.run(config, seed)`: executes burn-in, then evaluation horizon; returns metrics and paths; writes logs.
- `ccstk.sim.cli`: `python -m ccstk.sim.cli -config experiments/configs/default.yaml -seed 101`.

7.4 Testing Strategy

Unit tests.

- **Kernel invariants:** soft floor near $v = 5$, hard floor $v \geq 1$, mode duration countdown and resample, one-tick lock, capacity and budget constraints.
- **RNG determinism:** identical traces across platforms for the same seed; independent streams for local vs. global draws.
- **OU fit:** recovers parameters on synthetic AR(1); robust to outliers; ridge stabilizes ϕ near 1.
- **PF:** weight normalization, ESS computation, unbiased resampling (systematic).
- **Bands:** monotonicity: increasing F widens $|\ell|$, narrows u; $\phi \uparrow$ (slower reversion) widens band.
- **Allocator:** greedy correctness under integer lots; budget never negative.

Property-based tests (Hypothesis).

- $\forall$ sequences: executing sells before buys never reduces feasible buy volume.
- $\forall \gamma \in (0, 1)$: discounted sum equals Kahan-summed undiscounted when $\gamma \to 1$ (within tolerance).

Golden parity checks.

- Freeze seed and config; serialize the first 10^4 ticks; compare hashes in CI against a stored "gold" artifact.

7.5 Numerical Stability and Performance

- **Stable powers:** compute γ^T as $\exp(T \ln \gamma)$; clamp $T \geq 0$.
- **Summation:** use Kahan or pairwise summation for cumulative profit.
- **Complexity:** per tick $\mathcal{O}(G) + \mathcal{O}(GN)$ for PF with N particles; target $N \leq 500$.
- **Acceleration:** Numba `@njit` for `tick`, PF propagation, and band solver loops.

7.6 Reproducibility and Artifacts

- **Config immutability:** all runs specified by a single config file; SHA of the config embedded in outputs.
- **Seeds:** store per-stream seeds and a combined RNG state hash each tick in logs.
- **Outputs:** per-tick Parquet logs, aggregated JSON metrics, and plots; organized by `experiment_id/seed/horizon`.

7.7 Risk Register and Fallbacks

- **Non-convergence (EM):** cap EM iterations; if not converged, fall back to SR–OU bands for that segment.
- **Band solver divergence:** inflate κ by 10% and retry; else use symmetric $\bar{x} \pm c\sigma$.
- **PF degeneracy:** resample at ESS/N < 0.5; add small Gaussian jitter to d.
- **Performance regressions:** profile; drop to lower N or longer estimator windows; document change.

7.8 Timeline (Indicative)

Week 1	M0 scaffolding; M1 kernel with tests; golden traces established.
Week 2	M2 estimators (SR–OU); M3 bands + solver; initial PF skeleton.
Week 3	M3 PF completed; M4 allocator + brokers; end-to-end single-seed runs.
Week 4	M5 harness/metrics; M6 default experiments; plots/tables for paper.

7.9 Definition of Done

- All unit and property tests pass in CI; golden parity hashes match.
- Simulator produces identical outputs across OS for a fixed seed.
- Controllers run under default config with wall-clock per tick $\leq 1\,\text{ms}$ (PF) on a laptop CPU or documented alternative.
- Pre-registered experiments complete for all seeds and horizons; metrics and figures generated reproducibly from configs.
- README documents install, run commands, and how to reproduce the paper's figures.

8 Anticipated Risks and Limitations

This section records the main technical, statistical, and interpretive risks that could affect the usefulness of the work, together with practical mitigations. The goal is to make the limitations explicit before implementation and to commit to guardrails that reduce the chance of drawing misleading conclusions.

8.1 Modeling Assumptions and Approximation Error

OU approximation vs. true kernel. The single–regime OU (AR(1)) and its regime–switching extensions smooth over several coded effects: heavy–tailed micro–kicks, discrete mode resets, and non-Gaussian global shocks. When tails dominate, OU-based bands may be overly tight. *Mitigation:* use Student-t innovations and an explicit jump mixture; inflate buy costs in band computation (conservative κ); include ablations with increased shock frequency in Section 6.

Stationarity and drift in baselines. Resting values shift with Bank Level; capacity grows with buildings. Treating $r^{(g)}$ and $q^{(g)}_{\max}$ as fixed within an evaluation window introduces non-stationarity when the game state changes. *Mitigation:* freeze the environment snapshot during an experiment (burn-in, then hold levels fixed), and re-run experiments at multiple snapshots.

Cross–asset dependence. The kernel couples goods only through the occasional global shock; otherwise the design assumes conditional independence across assets. Any hidden correlation induced by simultaneous capacity/budget constraints is not modeled by OU/RSO-HMM. *Mitigation:* keep the global-shock factor in the simulator; report sensitivity of multi–asset allocation to shock rate.

Floors and clamps. Soft/hard floors near $v \approx 5$ break linearity and bias OU fits at low prices. *Mitigation:* exclude clamped ticks from the estimation window or down–weight them via robust loss; cap bands to respect floors.

8.2 Identification and Statistical Stability

Near–unit–root estimation. With slow mean reversion, ϕ estimates are close to 1 and sensitive to window length, producing unstable band widths. *Mitigation:* ridge the AR(1) fit; report window–length ablations; propagate parameter uncertainty into band guardrails.

EM non-convexity (RSO–HMM). EM can converge to local optima; regime counts K are only weakly identified in short series. *Mitigation:* multi–start EM with pruning; initialize by clustering SR–OU residuals; compare K by held–out likelihood and AIC/BIC; fall back to SR–OU when EM fails to meet a likelihood improvement threshold.

Particle filter degeneracy. Weight collapse can yield spurious certainty and brittle actions. *Mitigation:* systematic resampling at ESS/N < 0.5; jitter the drift state post–resample; cap per–tick likelihood ratios; monitor ESS in logs.

8.3 Control/Optimization Risks

Band optimality gap. OU bands are myopic cycle approximations and ignore higher–order effects (future opportunities, budget coupling). *Mitigation:* benchmark against a full–information dynamic–programming upper bound; report fraction–of–bound; use conservative bands when the gap widens.

Hitting–time approximation. Replacing random first–passage times with mean–path $\widehat{T}$ biases cycle value, especially under jumps. *Mitigation:* smooth $\widehat{T}$ via log–interpolation; add a variance penalty proportional to $\widehat{\mathrm{Var}}$; test accuracy on synthetic OU+jump series.

Broker hiring myopia. The break–even rule uses a myopic projection of buy notional; misestimation causes over/under–hiring. *Mitigation:* throttle decisions (e.g., hourly), require margin above break–even, and log counterfactual ROI for audit.

8.4 Implementation and Engineering Risks

Kernel parity. Any deviation from the source tick order (Section 2) or RNG usage invalidates comparisons. *Mitigation:* golden–trace tests for fixed seeds; unit/property tests for each constraint and floor; independent RNG streams for local/global/mode draws.

Floating–point nondeterminism. Cross–platform differences in fused–multiply–add and math libraries can desynchronize traces. *Mitigation:* prefer NumPy reference implementations over vendor BLAS for scalar ops in the kernel; hash per–tick RNG states and serialized prices.

Numerical stability. Naive computation of γ^T and long sums can under/overflow or accumulate error. *Mitigation:* compute γ^T via $\exp(T \ln \gamma)$; use Kahan or pairwise summation; clamp $T \geq 0$; sanitize NaNs/Infs in logs.

Performance. Particle filtering and per–good estimation can exceed real–time budgets. *Mitigation:* Numba–jit critical loops; cap particles; batch vectorize across goods; adopt slower cadences for expensive steps (e.g., EM only every 30 ticks).

8.5 Evaluation and Reproducibility Risks

Simulator overfitting. Design choices tuned on the simulator may not generalize to minor kernel variants. *Mitigation:* lock the simulator before controller development; run ablations that stress tails and regime persistence.

Analytic flexibility. Changing metrics or seeds post hoc risks biased inference. *Mitigation:* adhere to the pre–registration in Section 6; store configs and seeds; use common random numbers and paired differences; bootstrap CIs with fixed seed sets.

Runtime budgets and incomplete grids. Partial ablations can favor some methods. *Mitigation:* prioritize breadth over depth (short/medium horizons for all methods) if resources bind; document deviations.

8.6 Scope and External Validity

Game vs. markets. The minigame omits microstructure realities (order books, latency, slippage, shorting, market impact) and has stylized costs (buy–only overhead). *Implication:* numerical gains here are not predictive of real P&L. *Value:* the project demonstrates methodology (state–space modeling, POMDP control, cost–aware execution, reproducible evaluation) relevant to quantitative research irrespective of domain.

Ethical/gameplay constraints. Developer toggles (e.g., ultra–fast ticks) and external automation may violate terms for some distributions of the game. *Mitigation:* restrict experiments to a stand–alone simulator; do not attach to a live game.

8.7 Residual Risks and What Will Be Reported

Despite mitigations, residual risks remain: (i) OU/RSO–HMM misspecification under extreme chaos; (ii) EM convergence variance; (iii) sensitivity of band placement to ϕ near 1; (iv) budget–coupled interactions not captured by per–good bands. I will report, for each experiment: golden–trace hash, seeds, window lengths, particle counts, and any fallbacks triggered (e.g., EM failover, band solver fallback), along with sensitivity plots that show how key metrics move under reasonable perturbations of estimation and control hyperparameters.

9 Conclusion

This paper turns the Cookie Clicker stock market into a precise laboratory for cost–aware trading under uncertainty. I reconstructed the minigame's mechanics (§2) into a stochastic kernel with mean reversion, regime switching, heavy–tailed shocks, cross–asset disturbances, transaction costs on buys, inventory limits, and a one–tick trade lock. On top of this kernel, I proposed a modeling stack that ranges from a single–regime OU baseline to a regime–switching HMM and a drift–explicit switching state space with IMM/Kalman inference (§3). These models feed implementable controllers: closed–form, cost–aware OU bands; belief–dependent (particle–filtered) bands for partial information; a greedy index allocator for multi–good budgets; and a broker–hiring rule grounded in discounted buy notional (§4). I then fixed a code–faithful implementation plan, numerical settings, and a pre–registered evaluation protocol with paired seeds and a full–information dynamic–programming upper bound (§5, §6). Finally, I documented risks and mitigations spanning model misspecification, EM non–convexity, particle filter degeneracy, and engineering parity with the source kernel (§8).

The contribution is twofold. Methodologically, the work demonstrates how to move from raw mechanics to a reproducible research pipeline: derive the state transition from code, choose tractable approximations that preserve the dominant economics, design controllers that make explicit trade–offs between edge and cost, and benchmark against a clear upper bound. Professionally, the project highlights skills relevant to quantitative trading: state–space modeling, hidden–regime inference, sequential decision making under constraints, cost modeling, experiment design with common random numbers, and disciplined reporting.

Three near–term outcomes are expected once the implementation begins. First, the auditor (golden–trace) simulator will enable exact parity tests and remove ambiguity about correctness. Second, the OU and particle–filtered band controllers should realize a measurable fraction of the dynamic–programming bound, making the optimality gap interpretable rather than anecdotal. Third, the budgeted index allocator and broker–hiring economics will translate modeling output into concrete execution decisions under realistic frictions.

Next steps. The following items translate directly into the initial coding milestones:

- Build the simulator with the tick order and RNG stream partitioning specified in §5; ship golden–trace tests.
- Implement stable OU estimation with ridge and robust volatility, then add the fixed–point band solver with guardrails (§4).
- Add the particle filter and IMM variants, exposing the posterior summaries required by the band logic.
- Wire the multi–good index allocator, capacity/budget checks, and the throttled broker–hiring rule.

- Stand up the evaluation harness using the pre–registered seeds, horizons, metrics, and reporting templates in §6.

Outlook. After the core system runs, I will extend the analysis in two directions: (i) stress tests that scale shock rates and alter regime persistence to map where OU–style bands remain reliable; and (ii) ablations that quantify how belief quality (particles, window lengths) trades off against runtime, a consideration central to real–time trading. The companion write–up that documents implementation choices, failures, and empirical results will close the loop and position the project as a complete research–to–production case study suitable for a quant internship portfolio.

References

Baum, L. E., T. Petrie, G. Soules, and N. Weiss (1970). “A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains”. In: *The Annals of Mathematical Statistics* 41.1, pp. 164–171.

Benjamini, Y. and Y. Hochberg (1995). “Controlling the False Discovery Rate”. In: *Journal of the Royal Statistical Society: Series B* 57.1, pp. 289–300.

Blom, H. A. P. and Y. Bar-Shalom (1988). “The Interacting Multiple Model Algorithm for Systems with Markovian Switching Coefficients”. In: *IEEE Transactions on Automatic Control* 33.8, pp. 780–783.

Cookie Clicker Wiki (Accessed YYYY-MM-DD). Community documentation of game mechanics.

DashNet (2013–). *Cookie Clicker*. Game by Orteil (Julien Thiennot); project unaffiliated.

Doucet, Arnaud, Nando de Freitas, and Neil Gordon, eds. (2001). *Sequential Monte Carlo Methods in Practice*. Springer.

Efron, B. (1979). “Bootstrap Methods: Another Look at the Jackknife”. In: *The Annals of Statistics* 7.1, pp. 1–26.

Gordon, N. J., D. J. Salmond, and A. F. M. Smith (1993). “Novel Approach to Nonlinear/Non-Gaussian Bayesian State Estimation”. In: *IEE Proceedings F* 140.2, pp. 107–113.

Huber, P. J. (1964). “Robust Estimation of a Location Parameter”. In: *The Annals of Mathematical Statistics* 35.1, pp. 73–101.

Kahan, W. (1965). “Further Remarks on Reducing Truncation Errors”. In: *Communications of the ACM* 8.1, p. 40.

Kalman, R. E. (1960). “A New Approach to Linear Filtering and Prediction Problems”. In: *Journal of Basic Engineering* 82.1, pp. 35–45.

Lange, K. L., R. J. A. Little, and J. M. G. Taylor (1989). “Robust Statistical Modeling Using the t Distribution”. In: *Journal of the American Statistical Association* 84.408, pp. 881–896.

Uhlenbeck, G. E. and L. S. Ornstein (1930). “On the Theory of the Brownian Motion”. In: *Physical Review* 36, pp. 823–841.

10 Appendix

A OU Band Derivation and a Tiny Solver

Consider the SR–OU approximation

$$x_{t+1} \;=\; \phi\, x_t + \mu + \sigma\, \varepsilon_{t+1}, \qquad \varepsilon_{t+1} \sim \mathcal{N}(0,1), \;\; \phi \in (0,1),$$

with stationary mean $\bar{x} := \mu/(1-\phi)$. Let the buy overhead factor be $F(b) = 1+\kappa$ (no sell cost) and the per–tick discount be $\gamma \in (0,1)$. A myopic cycle buys one unit at $x_t = \ell < \bar{x}$ and sells at the first time T such that $x_{t+T} \geq u > \bar{x}$.

Mean–path hitting–time approximation. Ignoring noise, the mean path satisfies $x_{t+h} \approx \bar{x} + \phi^h(x_t - \bar{x})$. Solving $x_{t+T} \approx u$ from $x_t = \ell$ gives

$$\widehat{T}(\ell \to u) \;=\; \left\lceil \frac{\log((u-\bar{x})/(\ell-\bar{x}))}{-\log\phi} \right\rceil.$$

Discounted cycle value. Measured in *\$econds* with price $v = r + x$,

$$\Pi(\ell, u) \;\approx\; (r+u)\,\gamma^{\widehat{T}(\ell \to u)} \;-\; F(b)\,(r+\ell).$$

Treating $\widehat{T}$ as the smooth surrogate $T(\ell, u) \equiv \big[\log((u-\bar{x})/(\ell-\bar{x}))\big]/(-\log\phi)$, the first–order conditions are

$$\partial_u \Pi(\ell, u) \;=\; \gamma^T \left[1 \;+\; (r+u)\,\frac{\ln\gamma}{-\log\phi}\,\frac{1}{u-\bar{x}}\right] = 0, \tag{1}$$

$$\partial_\ell \Pi(\ell, u) \;=\; -F(b) \;-\; (r+u)\,\gamma^T\,\frac{\ln\gamma}{-\log\phi}\,\frac{1}{\ell-\bar{x}} = 0. \tag{2}$$

Because $\ln\gamma < 0$ and $-\log\phi > 0$, the feasible solution has $\ell < \bar{x} < u$ and is asymmetric when $\kappa > 0$ (buy cost only): $|\ell - \bar{x}| > |u - \bar{x}|$.

Fixed–point solver (10 lines). Let $c > 0$ be an initial band width in σ–units (e.g. $c = 1.5$).

```
Input: r, phi, barx, gamma, F, sigma, c, steps K=12, stepsizes eta1=eta2=0.3
Initialize: u = barx + c*sigma;  ell = barx - c*sigma
repeat K times:
  T  = log((u-barx)/(ell-barx))/(-log(phi))
  g1 = 1 + (r+u)*log(gamma)/( (-log(phi))*(u-barx) )
  g2 = -F - (r+u)*exp(T*log(gamma))*log(gamma)/( (-log(phi))*(ell-barx) )
  u   = u   - eta1 * g1  * max(1, |u-barx|)
  ell = ell - eta2 * g2  * max(1, |ell-barx|)
  enforce: ell < barx < u,  and  u - ell >= 0.25*sigma
return (ell, u)
```

If the loop fails to contract, (i) inflate κ by 10% and retry; else (ii) fall back to symmetric bands $\bar{x} \pm c\sigma$ with a larger c.

Monotonicity (informal). From (1)–(2): increasing κ (larger F) pushes ℓ downward (requires deeper buys) while leaving u nearly unchanged; decreasing ϕ (faster reversion) shrinks T and allows tighter (ℓ, u).

B RSO–HMM: EM Updates (Sketch)

Let hidden mode $m_t \in \{1, \dots, K\}$ follow a Markov chain with transition matrix Π. Conditional on $m_t = k$,

$$x_{t+1} \;=\; \phi_k x_t + \mu_k + \sigma_k\, \varepsilon^{(k)}_{t+1}, \qquad \varepsilon^{(k)}_{t+1} \sim \text{Student-}t_{\nu_k}.$$

E–step (forward–backward). Compute $\gamma_t(k) = \Pr(m_t = k \mid x_{0:T})$ and $\xi_t(i,j) = \Pr(m_t = i, m_{t+1} = j \mid x_{0:T})$. Define residuals $e^{(k)}_{t+1} = x_{t+1} - \phi_k x_t - \mu_k$ and weights

$$w^{(k)}_{t+1} \;=\; \frac{\nu_k + 1}{\nu_k + \big(e^{(k)}_{t+1}/\sigma_k\big)^2}.$$

M–step (weighted t-regression). Update transitions:

$$\Pi^{\text{new}}_{ij} \;=\; \frac{\sum_{t=0}^{T-1} \xi_t(i,j)}{\sum_{t=0}^{T-1} \gamma_t(i)}.$$

Update AR(1) parameters for each k by weighted least squares:

$$\phi^{\text{new}}_k,\ \mu^{\text{new}}_k = \arg\min_{\phi,\mu} \sum_{t=0}^{T-1} \gamma_t(k)\, w^{(k)}_{t+1} \big(x_{t+1} - \phi x_t - \mu\big)^2,$$

$$(\sigma^2_k)^{\text{new}} = \frac{\sum_{t=0}^{T-1} \gamma_t(k)\, w^{(k)}_{t+1}\, \big(e^{(k)}_{t+1}\big)^2}{\sum_{t=0}^{T-1} \gamma_t(k)}.$$

Update ν_k by solving the 1-D equation

$$-\psi\Big(\frac{\nu_k}{2}\Big) + \log\Big(\frac{\nu_k}{2}\Big) + 1 - \frac{\sum_t \gamma_t(k)\,\big(\log w^{(k)}_{t+1} - w^{(k)}_{t+1}\big)}{\sum_t \gamma_t(k)} \;=\; 0,$$

where $\psi(\cdot)$ is the digamma function (solve by Newton or bisection on $\nu_k \in [3, 50]$).

C Particle Filter with Systematic Resampling

State per particle $\theta = (d, m)$. At tick t with observation x_t:

1. **Predict:** For each particle i,

$$\begin{aligned} m^{(i)}_t &\sim \Pi(\, m^{(i)}_{t-1}, \cdot\,), \\ d^{(i)}_t &= a_{m^{(i)}_t}\, d^{(i)}_{t-1} + b_{m^{(i)}_t} + \eta^{(i)}_t, \quad \eta^{(i)}_t \sim \mathcal{N}(0, \sigma^2_{d,m}). \end{aligned}$$

2. **Weight:** Using SR–OU emission as a proxy,

$$w^{(i)}_t \propto \frac{1}{\hat\sigma_{m^{(i)}_t}} \exp\!\left(-\frac{1}{2}\frac{\big(x_t - \phi\, x_{t-1} - d^{(i)}_t\big)^2}{\hat\sigma^{\,2}_{m^{(i)}_t}}\right),$$

   then normalize.

3. **ESS and resample:** If ESS $= (\sum_i (w_t^{(i)})^2)^{-1} < 0.5N$, do *systematic resampling*: draw $u \sim \mathrm{Unif}(0, 1/N)$, set thresholds $u_j = u + (j-1)/N$, traverse cumulative weights to select ancestors.
4. **Jitter:** After resampling, add small $\mathcal{N}(0, \sigma_\mathrm{j}^2)$ jitter to d_t to avoid collapse.
5. **Summaries:** Report $\hat{x}_{t+1|t}$, $\widehat{\mathrm{Var}}_t$ by mapping particles to $(\hat{\phi}, \hat{\mu}, \hat{\sigma})$ as in §4.

D Full-Information DP Upper Bound (Sketch)

For one good with full state (x, d, m, τ, q, C) known, finite–horizon value satisfies

$$V_t(s) \;=\; \max_{a \in \mathcal{A}(s)} \Big\{ R_t(s,a) + \gamma\, \mathbb{E}\big[V_{t+1}(S') \mid s, a\big] \Big\}.$$

Linearity of R_t in $v = r + x$ and monotone next–state laws in x imply a *band* structure: $\exists\, \ell_t(\cdot), u_t(\cdot)$ with buy if $x \le \ell_t$, sell if $x \ge u_t$, hold otherwise (monotone optimal policies under supermodularity/submodularity arguments). With integer inventory and costs, the optimal n jumps to boundaries (an (s, S)–type policy). Summing per–good DP values yields an *upper bound* for implementable controllers that face a shared budget (coupling relaxed).

E Symbol Glossary

Symbol	Meaning
$v_t^{(g)}$	Price of good g at tick t (in *\$econds*)
$r^{(g)}$	Resting value (deterministic, bank–level shifted)
$x_t^{(g)}$	Deviation $v_t^{(g)} - r^{(g)}$
$d_t^{(g)}$	Drift term (latent, updated each tick)
$m_t^{(g)}$	Mode / regime index
$\tau_t^{(g)}$	Mode time–to–live counter
$q_t^{(g)}$	Inventory (units)
$q_{\max}^{(g)}$	Capacity for good g
C_t	Cookies on hand (budget)
$\kappa(b)$	Proportional buy overhead with b brokers
$F(b)$	Buy factor $1 + \kappa(b)$
ϕ, μ, σ	SR–OU AR(1) parameters
Π	Mode transition matrix (RSO–HMM)
γ	Per–tick discount factor
$\mathrm{Idx}^{(g)}$	Edge–per–cost index for allocation
$\bar{x}$	OU stationary mean $\mu/(1-\phi)$

F Default Hyperparameters (for Reproducibility)

Setting	Default
Discount γ	0.999 (or day half–life: $\gamma = \exp(-\ln 2/1440)$)
OU fit window H	120 ticks; ridge $\lambda = 10^{-4}$
Volatility estimate	MAD×1.4826 on AR(1) residuals
Particles N	512; resample if ESS/N < 0.5
Band guardrails	$\lvert u - \ell \rvert \geq 0.25\,\sigma$; $\ell \geq 1 - r$
Broker check cadence	Every 60 ticks; break–even with 1200 *\$econds* cost
RNG streams	`price-local`, `price-global`, `mode`, `trade-ties`