ALPHARJM: REWARD-JUMP MEMORY FOR STOCHASTIC RETURN-GUIDED ALPHA DISCOVERY

Anonymous authors
Paper under double-blind review

ABSTRACT

Formulaic alpha discovery is a pool-dependent symbolic search problem in which informative feedback is observed primarily when a complete expression is evaluated. This delayed feedback creates two coupled difficulties: the retained alpha pool does not preserve the full history of realized evaluation feedback, and the value of an intermediate construction action is uncertain because its consequence depends on the formula eventually completed. We introduce **AlphaRJM**, which addresses these difficulties through **Reward-Jump Memory**, an event-driven latent state that remains fixed during token construction and updates only at terminal evaluation events using the realized pool reward and evaluation outcome, and an **action-conditioned SDE return critic** that represents future discounted discovery returns with stochastic particles. The particles guide action selection through their mean and uncertainty and are learned using a distributional Bellman objective combining energy-distance matching, mean calibration, and jump regularization. Empirically, AlphaRJM delivers strong and stable gains across multiple equity universes, forecasting horizons, and random seeds, while ablations confirm the complementary roles of persistent evaluation history, stochastic return modeling, and distributional supervision.

1 INTRODUCTION

Formulaic alphas are interpretable symbolic expressions that transform historical market variables into predictive signals. Automating their discovery is challenging because the space of candidate expressions is combinatorial and, more importantly, the usefulness of a candidate depends on the alpha pool already constructed. A moderately predictive formula that contributes complementary information may be more valuable than a stronger but redundant one. AlphaGen formalizes this interaction by rewarding a candidate according to its marginal contribution to the combined alpha ensemble (Yu et al., 2023), turning formula discovery into a pool-dependent sequential decision problem with sparse and evolving feedback.

Recent methods improve different aspects of this search process. AlphaQCM introduces distributional reinforcement learning and uncertainty-aware exploration through learned return quantiles and Quantile Conditional Moments (QCM) (Zhu & Zhu, 2025; Zhang & Zhu, 2026). AlphaSAGE instead strengthens expression representation and search diversity through structure-aware graph encoding, Generative Flow Networks (GFlowNets), and dense multi-faceted rewards (Chen et al., 2026). These advances improve uncertainty modeling, structural understanding, and exploration, but they do not explicitly maintain a persistent representation of the sequence of terminal rewards and evaluation outcomes produced during search.

This distinction matters because terminal feedback can be informative even when the retained pool remains unchanged. For example, a rejected candidate provides a different evaluation outcome from an accepted or replaced one, while an invalid construction produces another distinct terminal event. Consequently, two search trajectories may arrive at the same retained pool after different sequences of realized rewards and categorical outcomes. Conditioning subsequent decisions only on the current pool therefore discards part of the evaluation history accumulated during search. This motivates a persistent state that evolves with terminal evaluation feedback while remaining unchanged during intermediate token construction.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Delayed terminal evaluation creates a second, closely related difficulty. At an intermediate token step, the consequence of selecting an action is not observed immediately: its utility depends on the expression that will eventually be completed and on the pool against which that expression will be evaluated. The future discovery return associated with an action is therefore naturally uncertain rather than a single deterministic quantity. A distributional representation can preserve this uncertainty and provide both a central estimate and a dispersion signal for action selection.

To address these two coupled challenges, we propose AlphaRJM, a formulaic alpha-discovery framework that combines Reward-Jump Memory with an action-conditioned stochastic differential equation (SDE) return critic. Reward-Jump Memory is an event-driven persistent latent state that remains unchanged during token-level formula construction and updates only when a terminal evaluation event occurs. Each update is driven by the realized pool reward and categorical evaluation outcome, allowing this evaluation history to persist across formula boundaries independently of whether the retained pool changes.

Complementing this persistent history, the action-conditioned SDE critic models the uncertain future discovery return of each candidate construction action. Conditioned on the encoded partial expression, encoded alpha pool, persistent memory, and candidate action, the critic defines a state–action-specific stochastic return process. Independent trajectories generate particles representing plausible future discounted discovery returns, whose empirical mean and uncertainty directly guide action selection. The SDE is used solely as an internal stochastic return critic for symbolic search rather than as a model of asset prices or alpha signals. Its particle distribution is trained through a distributional Bellman objective combining energy-distance matching, mean calibration, and jump regularization.

Empirically, AlphaRJM achieves strong and stable gains across multiple equity universes, forecasting horizons, and random seeds, while controlled ablations support the complementary roles of persistent evaluation history, stochastic return modeling, and distributional supervision.

Contributions. Our main contributions are:

- We introduce **Reward-Jump Memory**, an event-driven persistent state that updates only after terminal evaluations using the realized pool reward and categorical outcome, allowing evaluation history to persist across formula episodes.
- We develop an **action-conditioned SDE return critic** that represents uncertain future discovery returns with stochastic particles and uses their mean and uncertainty to guide symbolic action selection.
- We train the resulting framework through **distributional Bellman learning** with sequence-consistent memory replay, and demonstrate its effectiveness across multiple equity universes, forecasting horizons, and random seeds.

2 Background and Related Work

Formulaic alpha discovery searches for interpretable symbolic expressions whose usefulness depends on the existing alpha pool. AlphaGen formalizes this interaction by rewarding a candidate according to its marginal contribution to the combined ensemble (Yu et al., 2023). AlphaRJM adopts this pool-dependent setting while additionally maintaining a persistent representation of realized terminal evaluation feedback across successive formula episodes.

Distributional reinforcement learning models the distribution of discounted returns rather than only their expectation (Bellemare et al., 2017; Dabney et al., 2018). AlphaQCM applies this principle to formulaic alpha discovery through learned return quantiles and Quantile Conditional Moments for uncertainty-aware exploration (Zhu & Zhu, 2025; Zhang & Zhu, 2026). AlphaRJM retains the distributional perspective but conditions future returns on the current expression, alpha pool, persistent evaluation history, and candidate action, and represents the resulting return law using particles generated by an action-conditioned SDE.

A complementary line of work improves symbolic representation and exploration. AlphaSAGE combines structure-aware relational graph representations with Generative Flow Networks (GFlowNets) and dense multi-faceted rewards to improve expression modeling and search

diversity (Chen et al., 2026; Schlichtkrull et al., 2018; Bengio et al., 2021; Malkin et al., 2022). AlphaRJM addresses a different aspect of the search process: retaining realized reward–outcome information across formula boundaries and using it to condition subsequent distributional action evaluation.

Neural SDEs provide flexible mechanisms for representing and sampling stochastic dynamics (Kidger et al., 2021), while energy-distance and proper-scoring objectives support sample-based distribution learning (Gneiting & Raftery, 2007; Székely & Rizzo, 2013). In AlphaRJM, the SDE is used solely as an internal stochastic return critic rather than as a model of asset prices or alpha signals.

Overall, AlphaRJM complements prior work on pool-dependent rewards, distributional exploration, and structure-aware symbolic generation by coupling persistent terminal evaluation history with a sampleable distribution of future discovery returns.

3 METHODOLOGY

AlphaRJM augments pool-dependent symbolic alpha discovery with two coupled components: an event-driven persistent memory that carries terminal evaluation feedback across formula episodes, and an action-conditioned stochastic return critic that represents uncertain future discovery returns. We first formalize the symbolic search state and terminal-event structure, then introduce Reward-Jump Memory, stochastic return-guided action selection, and distributional Bellman learning.

3.1 FRAMEWORK AND SEARCH STATE

We follow the grammar-based symbolic construction setting used in prior formulaic alpha-discovery methods (Yu et al., 2023; Zhu & Zhu, 2025; Chen et al., 2026). Let t index primitive token-level decisions and let $n \in \{0, \dots, N-1\}$ denote the number of completed formula episodes before the current decision, where N is the total formula-generation budget.

At step t, the search context contains a partial expression x_t and the current retained alpha pool $\mathcal{P}_t$. Let $\mathcal{A}_t$ denote the set of actions exposed by the current symbolic construction mask. An action

$$a_t \in \mathcal{A}_t \tag{1}$$

appends an operator, market feature, temporal token, constant, or terminal symbol to the current expression. The exact grammar and construction-mask protocol are given in Appendix B.2.

Let $\delta_t \in \{0, 1\}$ indicate whether action a_t produces a terminal formula evaluation. Thus, $\delta_t = 0$ denotes an intermediate construction step and $\delta_t = 1$ terminates and evaluates the current formula episode. In the reported environment every terminated formula attempt produces such an evaluation event, so δ_t also serves as the formula-episode termination indicator.

At terminal events, let o_t denote the realized categorical evaluation outcome, with

$$o_t \in \mathcal{O} := \{\text{accepted}, \text{replaced}, \text{rejected}, \text{invalid}\}. \tag{2}$$

The outcome is irrelevant when $\delta_t = 0$.

Following the synergistic pool formulation of AlphaGen (Yu et al., 2023), the reward is the marginal change in training-set ensemble Information Coefficient (IC):

$$r_t = \begin{cases} Q(\mathcal{P}_{t+1}) - Q(\mathcal{P}_t), & \delta_t = 1, \\ 0, & \delta_t = 0, \end{cases} \tag{3}$$

where $Q(\mathcal{P})$ denotes the IC of the weighted alpha ensemble represented by pool $\mathcal{P}$. Hence the same candidate may have different utility under different retained pools.

Let E_ϕ denote the partial-expression encoder with parameters ϕ and P_ψ the permutation-invariant pool encoder with parameters ψ. Their outputs are

$$e_t = E_\phi(x_t), \qquad p_t = P_\psi(\mathcal{P}_t), \tag{4}$$

where e_t represents the current expression and p_t the retained pool. AlphaRJM augments these observable representations with a persistent latent memory H_t and defines the search state

$$s_t = (e_t, p_t, H_t). \tag{5}$$

The two history-bearing components have distinct roles: p_t summarizes the formulas currently retained in the ensemble, whereas H_t summarizes the sequence of realized reward–outcome feedback from previous terminal evaluations.

3.2 Reward-Jump Memory

At the beginning of the search stream, the persistent state is initialized from the initial expression and pool representations:

$$H_0 = \tanh(W_0[e_0; p_0] + b_0)\,. \tag{6}$$

Here $[\cdot\,;\cdot]$ denotes vector concatenation. The implementation details of this one-time initialization are provided in Appendix B.4.

For a terminal event, the realized reward and outcome are encoded into an event mark

$$\chi_t = \mathcal{M}_\eta(r_t, o_t), \tag{7}$$

where $\mathcal{M}_\eta$ is the learned event encoder. The event mark and current memory are then transformed into a bounded candidate jump:

$$\begin{aligned} &u_t = \mathcal{C}_\eta([H_t; \chi_t]), \\ &g_t = \mathrm{sigmoid}(W_g u_t + b_g), \qquad \zeta_t = \tanh(W_\zeta u_t + b_\zeta), \\ &J_t = J_{\max}\, g_t \odot \zeta_t, \end{aligned} \tag{8}$$

where $\mathcal{C}_\eta$ is the learned jump-context network, g_t is an element-wise gate, ζ_t is a signed direction vector, $\odot$ denotes the Hadamard product, and $J_{\max} > 0$ bounds the magnitude of every jump coordinate. We use η collectively for the learned parameters of the event encoder, jump-context network, and jump heads.

The persistent state evolves according to

$$H_{t+1} = \begin{cases} H_t + J_t, & \delta_t = 1, \\ H_t, & \delta_t = 0. \end{cases} \tag{9}$$

Thus H_t is exactly constant throughout intermediate token construction and changes only after terminal evaluation feedback. Importantly, completion of a formula ends its Bellman episode but not the memory stream: the post-event state H_{t+1} is retained when construction of the next formula begins. This produces a fast token-construction timescale and a slower event-driven memory timescale.

3.3 Stochastic Return-Guided Alpha Discovery

Reward-Jump Memory carries past evaluation feedback, while action selection requires estimating the uncertain future consequence of each available construction action. AlphaRJM represents this uncertainty through an action-conditioned scalar SDE return critic.

For candidate action a, let $A_\omega(a)$ denote its learned action embedding with parameters ω. The critic conditioning vector is

$$q_{t,a} = [e_t; p_t; H_t; A_\omega(a)]. \tag{10}$$

A neural conditioning network maps $q_{t,a}$ to four scalars: an initial return state $z_{0,t,a}$, a long-run level $\mu_{t,a}$, a positive mean-reversion rate $\kappa_{t,a}$, and a bounded diffusion scale $\sigma_{t,a} \in [\sigma_{\min}, \sigma_{\max}]$, where $0 < \sigma_{\min} \le \sigma_{\max} < \infty$. For compactness, θ denotes the full online SDE-critic parameter collection, including the action-embedding parameters ω.

These quantities define the conditional return process

$$dZ_\tau^{t,a} = \kappa_{t,a}\big(\mu_{t,a} - Z_\tau^{t,a}\big)\,d\tau + \sigma_{t,a}\,dB_\tau, \qquad Z_0^{t,a} = z_{0,t,a}, \tag{11}$$

where $\tau \in [0,1]$ is an internal diffusion coordinate and B_τ is a standard one-dimensional Brownian motion. The scalar mean-reverting form provides an inexpensive sampleable return law with separately controlled location, reversion, and stochastic dispersion. Importantly, $Z_\tau^{t,a}$ is not a stock-price process or an alpha signal; it represents the stochastic discounted reinforcement-learning return associated with choosing action a in search state s_t. Its conditional law is characterized in Appendix F.

Simulating M independent trajectories of Eq. 11 and taking their terminal values at $\tau = 1$ yields the particle representation

$$\mathcal{Z}_\theta(s_t, a) = \left\{ Z_{t,a}^{(1)}, \ldots, Z_{t,a}^{(M)} \right\}, \tag{12}$$

where $Z_{t,a}^{(m)}$ denotes the terminal value of trajectory m. The implementation uses Euler–Maruyama integration as detailed in Appendix B.5.

The empirical mean and variance of these particles are

$$\overline{Z}_{t,a} = \frac{1}{M} \sum_{m=1}^{M} Z_{t,a}^{(m)}, \qquad \widehat{V}_{t,a} = \frac{1}{M} \sum_{m=1}^{M} \left(Z_{t,a}^{(m)} - \overline{Z}_{t,a} \right)^2. \tag{13}$$

AlphaRJM converts these statistics into the uncertainty-aware action score

$$S_t(a) = \overline{Z}_{t,a} + c_n \sqrt{\widehat{V}_{t,a} + \varepsilon_{\text{num}}}, \tag{14}$$

where $\varepsilon_{\text{num}} > 0$ is a numerical stabilizer and $c_n \geq 0$ is the uncertainty coefficient at formula episode n. The coefficient decreases during search, placing greater emphasis on uncertain actions early and increasingly emphasizing expected return later.

Action selection additionally uses an episode-dependent ϵ_n-greedy rule:

$$a_t = \begin{cases} \text{sample uniformly from } \mathcal{A}_t, & \text{with probability } \epsilon_n, \\ \arg\max_{a \in \mathcal{A}_t} S_t(a), & \text{with probability } 1 - \epsilon_n. \end{cases} \tag{15}$$

The schedules of c_n and ϵ_n, together with the exact construction mask, are specified in Appendix B.6.

3.4 Distributional Bellman Learning

For the executed action a_t, let

$$Z_t^{(m)} := Z_{t,a_t}^{(m)} \sim \mathcal{Z}_\theta(s_t, a_t), \qquad m = 1, \ldots, M, \tag{16}$$

denote particles from the online critic. Let $\bar{\theta}$ denote the parameters of a periodically synchronized target copy of the complete SDE return critic.

After the environment transition, the next state is

$$s_{t+1} = (e_{t+1}, p_{t+1}, H_{t+1}), \tag{17}$$

where H_{t+1} follows Eq. 9. For a nonterminal transition, the greedy next action is selected using the online stochastic score:

$$a_{t+1}^\star = \arg\max_{a \in \mathcal{A}_{t+1}} S_{t+1}(a). \tag{18}$$

The target critic independently generates

$$\widetilde{Z}_{t+1}^{(m)} \sim \mathcal{Z}_{\bar{\theta}}(s_{t+1}, a_{t+1}^\star), \qquad m = 1, \ldots, M, \tag{19}$$

and the distributional Bellman particles are

$$Y_t^{(m)} = r_t + \gamma(1 - \delta_t) \widetilde{Z}_{t+1}^{(m)}, \tag{20}$$

where $\gamma \in [0, 1)$ is the discount factor. When $\delta_t = 1$, the Bellman target therefore reduces to the realized pool reward and does not bootstrap across the formula boundary. The updated memory nevertheless persists into the next formula episode, separating episodic return learning from persistent evaluation history.

We match the predicted and Bellman-target particle distributions using the empirical one-dimensional energy distance:

$$\mathcal{L}_{\text{ED}} = \frac{2}{M^2} \sum_{m,m'=1}^{M} \left| Z_t^{(m)} - Y_t^{(m')} \right| - \frac{1}{M^2} \sum_{m,m'=1}^{M} \left| Z_t^{(m)} - Z_t^{(m')} \right| - \frac{1}{M^2} \sum_{m,m'=1}^{M} \left| Y_t^{(m)} - Y_t^{(m')} \right|. \tag{21}$$

Energy distance directly compares distributions through samples (Székely & Rizzo, 2013); in one dimension it is closely related to CRPS/energy-score objectives used in probabilistic prediction (Gneiting & Raftery, 2007). We therefore refer to Eq. 21 as a CRPS-style distributional objective.

Because finite particle sets can introduce sampling noise in the predicted central tendency, we additionally use

$$\mathcal{L}_{\text{mean}} = \text{Huber}\left(\frac{1}{M}\sum_{m=1}^{M} Z_t^{(m)}, \frac{1}{M}\sum_{m=1}^{M} Y_t^{(m)}\right), \tag{22}$$

where Huber denotes the Huber loss.

We further regularize the magnitude of memory jumps at actual evaluation events. Let $\mathcal{L}_{\text{jump}}$ denote the event-masked squared-ℓ_2 jump penalty; the exact replay-minibatch aggregation used by the implementation is given in Eq. 39 of Appendix B.7. The complete objective is

$$\mathcal{L} = \lambda_{\text{ED}}\mathcal{L}_{\text{ED}} + \lambda_{\text{mean}}\mathcal{L}_{\text{mean}} + \beta_J\mathcal{L}_{\text{jump}}, \tag{23}$$

where $\lambda_{\text{ED}} \geq 0$, $\lambda_{\text{mean}} \geq 0$, and $\beta_J \geq 0$ control the three loss components.

Finally, the persistence of H_t makes independently shuffled transition replay unsuitable: arbitrary reordering would break the event sequence that defines the memory trajectory. AlphaRJM therefore trains on contiguous transition sequences. Each sampled sequence starts from its stored pre-transition memory, uses a burn-in prefix to reconstruct the persistent trajectory under the current model, and applies the return-learning objective to the subsequent unroll. The expression encoder, pool encoder, Reward-Jump Memory, action embedding, and online SDE return critic are optimized jointly, while the target SDE critic is synchronized periodically. Full architecture, discretization, exploration, replay, and optimization details are given in Appendix B; theoretical properties are given in Appendix F.

4 Experiments and Results

4.1 Experiment Setting

Evaluation Metrics. Following established formulaic alpha discovery protocols (Yu et al., 2023; Zhu & Zhu, 2025; Chen et al., 2026), we evaluate predictive performance using four correlation-based metrics: Information Coefficient (IC), Information Coefficient Information Ratio (ICIR), Rank Information Coefficient (RIC), and Rank Information Coefficient Information Ratio (RICIR). Higher values indicate better performance for all metrics. Detailed definitions and evaluation settings are provided in Appendix D.2.

Datasets. We evaluate all methods on three representative Chinese equity universes: CSI300, CSI500, and CSI800. For all datasets, we use a common chronological split, with January 1, 2010–December 31, 2020 for training, January 1, 2021–December 31, 2021 for validation, and January 1, 2022–December 31, 2024 for testing. This fixed split is used consistently across all methods and random seeds. Additional protocol details are provided in Appendix D.1, and the AlphaRJM configurations are reported in Appendix B.9. We use h to denote the forecasting horizon; the standard benchmark uses $h = 20$, while the long-horizon experiment uses $h = 42$.

Baselines. We compare AlphaRJM with representative baselines spanning three model families: (1) conventional predictive models, including a multilayer perceptron (MLP) (Murtagh, 1991) and Light Gradient Boosting Machine (LightGBM) (Ke et al., 2017); (2) continuous-time neural models, including Neural ODE (ordinary differential equation) (Chen et al., 2018) and Neural SDE (Kidger et al., 2021); and (3) formulaic alpha discovery methods based on reinforcement learning, including AlphaGen (Yu et al., 2023), AlphaQCM (Zhu & Zhu, 2025), and AlphaSAGE (Chen et al., 2026). All baselines are evaluated under the same benchmark protocol whenever applicable; implementation details are provided in Appendix D.3.

4.2 Overall Performance and Robustness

Table 1 reports IC, ICIR, RIC, and RICIR on CSI300, CSI500, and CSI800, averaged over four random seeds (0–3). AlphaRJM achieves the best mean result in 10 of the 12 dataset–metric comparisons and ranks second in the remaining two. It leads both IC and ICIR on all three universes

Table 1: Performance comparison on CSI300, CSI500, and CSI800 using correlation-based evaluation metrics. Results are reported as mean ± standard deviation over four random seeds (0–3). Higher values are better for all metrics. Best and second-best performances are shown in **bold** and underlined, respectively; rankings are determined from the unrounded mean values.

Dataset	Method	IC	ICIR	RIC	RICIR
CSI300	MLP	0.030 ± 0.017	0.218 ± 0.119	0.035 ± 0.024	0.269 ± 0.186
	LightGBM	0.024 ± 0.003	0.242 ± 0.028	0.020 ± 0.001	0.191 ± 0.010
	Neural ODE	0.037 ± 0.004	0.260 ± 0.028	0.032 ± 0.006	0.214 ± 0.040
	Neural SDE	0.032 ± 0.005	0.239 ± 0.040	0.026 ± 0.005	0.188 ± 0.035
	AlphaGen	$\underline{0.041 \pm 0.011}$	$\underline{0.302 \pm 0.080}$	$\underline{0.046 \pm 0.021}$	$\underline{0.326 \pm 0.156}$
	AlphaQCM	0.039 ± 0.007	0.276 ± 0.051	0.043 ± 0.009	0.301 ± 0.073
	AlphaSAGE	0.033 ± 0.016	0.246 ± 0.080	0.037 ± 0.017	0.266 ± 0.088
	AlphaRJM (ours)	$\mathbf{0.042 \pm 0.007}$	$\mathbf{0.307 \pm 0.058}$	$\mathbf{0.046 \pm 0.003}$	$\mathbf{0.331 \pm 0.038}$
CSI500	MLP	0.031 ± 0.011	0.269 ± 0.088	0.034 ± 0.018	0.299 ± 0.162
	LightGBM	0.027 ± 0.003	0.311 ± 0.027	0.031 ± 0.007	0.364 ± 0.077
	Neural ODE	0.029 ± 0.002	0.239 ± 0.014	0.028 ± 0.002	0.218 ± 0.018
	Neural SDE	0.025 ± 0.005	0.215 ± 0.030	0.024 ± 0.007	0.186 ± 0.055
	AlphaGen	0.036 ± 0.005	0.273 ± 0.025	0.041 ± 0.007	0.339 ± 0.061
	AlphaQCM	0.035 ± 0.007	0.296 ± 0.053	0.042 ± 0.012	0.360 ± 0.085
	AlphaSAGE	$\underline{0.041 \pm 0.007}$	$\underline{0.336 \pm 0.039}$	$\underline{0.059 \pm 0.016}$	$\underline{0.497 \pm 0.061}$
	AlphaRJM (ours)	$\mathbf{0.046 \pm 0.014}$	$\mathbf{0.379 \pm 0.159}$	$\mathbf{0.059 \pm 0.015}$	$\mathbf{0.523 \pm 0.173}$
CSI800	MLP	0.015 ± 0.007	0.176 ± 0.082	0.018 ± 0.013	0.206 ± 0.152
	LightGBM	0.026 ± 0.003	$\underline{0.305 \pm 0.035}$	0.025 ± 0.004	0.286 ± 0.045
	Neural ODE	0.026 ± 0.002	0.237 ± 0.024	0.024 ± 0.003	0.197 ± 0.033
	Neural SDE	0.028 ± 0.003	0.264 ± 0.020	0.028 ± 0.003	0.239 ± 0.022
	AlphaGen	0.028 ± 0.004	0.258 ± 0.054	0.039 ± 0.007	0.344 ± 0.088
	AlphaQCM	$\underline{0.029 \pm 0.017}$	0.260 ± 0.147	0.041 ± 0.019	0.332 ± 0.150
	AlphaSAGE	0.025 ± 0.020	0.208 ± 0.159	$\mathbf{0.045 \pm 0.015}$	$\mathbf{0.396 \pm 0.082}$
	AlphaRJM (ours)	$\mathbf{0.036 \pm 0.007}$	$\mathbf{0.339 \pm 0.027}$	$\underline{0.043 \pm 0.007}$	$\underline{0.371 \pm 0.048}$

Table 2: IC and RIC performance on CSI300 and CSI500 for the 42-trading-day forecasting horizon ($h = 42$) using random seed 0. Higher values are better. Best and second-best results are shown in **bold** and underlined, respectively.

Method	CSI300		CSI500	
	IC	RIC	IC	RIC
MLP	**0.046**	0.056	0.041	0.050
LightGBM	0.027	0.026	0.031	0.035
Neural ODE	0.014	0.014	0.028	0.033
Neural SDE	0.007	0.005	0.019	0.015
AlphaGen	0.021	0.027	0.034	0.044
AlphaQCM	<u>0.045</u>	<u>0.062</u>	<u>0.044</u>	<u>0.052</u>
AlphaSAGE	0.025	0.043	0.033	0.050
AlphaRJM (ours)	**0.046**	**0.066**	**0.051**	**0.065**

and all four metrics on CSI300 and CSI500; on CSI800, it achieves the best IC and ICIR and ranks second on RIC and RICIR. Despite stochastic variation across seeds, the method retains strong mean performance overall, indicating that the gains are not driven by a single favorable initialization.

Figure 1 provides a complementary view on CSI500. AlphaRJM maintains the strongest cumulative-wealth trajectory for most of the 2022–2024 test period and recovers strongly after major market declines, including the late-2024 rebound. The CSI500 index remains substantially below the learned strategies over most of the evaluation period, supporting the persistence of the predictive advantage observed in Table 1.

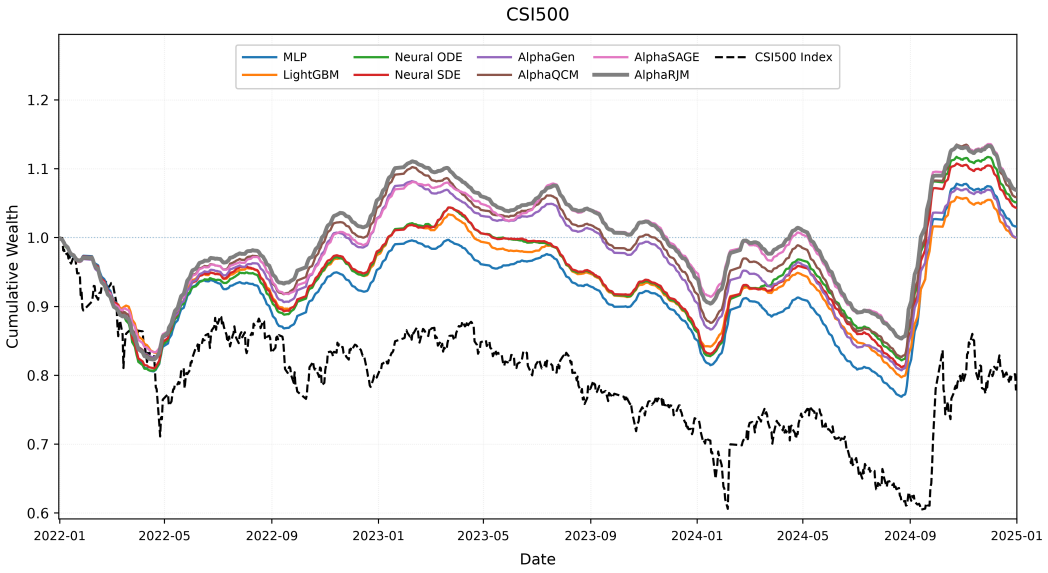


Figure 1: Cumulative wealth curves on CSI500 over the test period, averaged across random seeds 0–3.

Controlled ablations in Appendix E.1 isolate the contributions of the SDE return critic, distributional supervision, and persistent memory.

4.3 Long-Horizon Generalization

At the 42-trading-day forecasting horizon ($h = 42$), Table 2 reports the long-horizon results. AlphaRJM ranks first on both IC and RIC for CSI300 and CSI500, achieving $(0.046, 0.066)$ and $(0.051, 0.065)$, respectively. Overall, AlphaRJM ranks first in four of the four long-horizon dataset–metric comparisons.

Figure 4 in Appendix E.3 shows that this advantage also extends to cumulative wealth on CSI500: AlphaRJM maintains a clear lead for much of the test period and reaches the highest cumulative-wealth level during the late-2024 recovery. These results suggest that its predictive effectiveness is not limited to the standard forecasting horizon.

4.4 Sensitivity Analysis

We evaluate AlphaRJM on CSI300 by varying the number of SDE particles, integration steps, energy-distance (CRPS-style) loss weight λ_{ED}, and maximum diffusion scale $\sigma_{\max}$ (Fig. 2). Fewer particles generally yield stronger correlation metrics, while performance is relatively insensitive to the number of integration steps. The energy-distance weight introduces a trade-off across IC- and rank-based metrics, and moderate $\sigma_{\max}$ gives the most balanced results, whereas excessive diffusion degrades performance. Overall, AlphaRJM exhibits smooth and stable behavior across the tested configurations. The exact one-factor-at-a-time settings are given in Appendix C.

5 Conclusion

We introduced AlphaRJM, a formulaic alpha-discovery framework that combines persistent terminal-evaluation history with stochastic return-guided symbolic search. Reward-Jump Memory remains fixed during token-level construction and updates only after terminal evaluations using the realized pool reward and categorical outcome, allowing evaluation history to persist across formula episodes even when the retained pool is unchanged. An action-conditioned SDE return critic complements this memory by generating particles that represent uncertain future discounted discov-

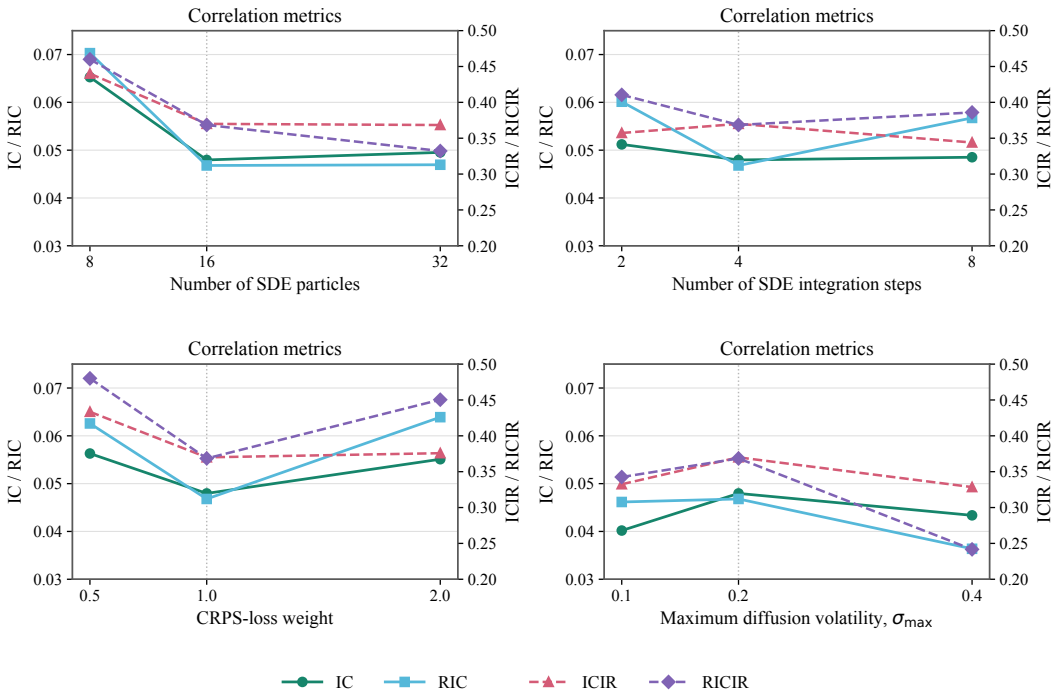


Figure 2: Sensitivity analysis of AlphaRJM on CSI300 with respect to the number of SDE particles, number of SDE integration steps, energy-distance (CRPS-style) loss weight λ_{ED}, and maximum diffusion scale $\sigma_{\max}$. The figure reports IC, RIC, ICIR, and RICIR, with the dotted vertical lines indicating the default configurations.

ery returns; their empirical mean and uncertainty guide action selection, while their distribution is learned through a CRPS-style distributional Bellman objective.

Across CSI300, CSI500, and CSI800, AlphaRJM achieves strong and stable correlation-based performance relative to conventional, continuous-time, and formulaic alpha-discovery baselines. The improvement extends to the 42-trading-day forecasting setting, while cumulative-wealth results provide complementary evidence of out-of-sample effectiveness. Ablation experiments support the complementary contributions of persistent evaluation history, stochastic return modeling, and distributional supervision, and sensitivity analysis shows stable behavior across the tested configurations. Overall, the results indicate that preserving terminal reward–outcome history and modeling future discovery returns distributionally provide complementary signals for pool-dependent symbolic alpha discovery.

REFERENCES

Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International conference on machine learning*, pp. 449–458. Pmlr, 2017.

Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. *Advances in neural information processing systems*, 34:27381–27394, 2021.

Binqi Chen, Hongjun Ding, Ning Shen, Taian Guo, Jinsheng Huang, Luchen Liu, and Ming Zhang. Alphasage: Structure-aware alpha mining via gflownets for robust exploration. In *International Conference on Learning Representations*, volume 2026, pp. 2955–2982, 2026.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Will Dabney, Georg Ostrovski, David Silver, and Rémi Munos. Implicit quantile networks for distributional reinforcement learning. In *International conference on machine learning*, pp. 1096–1105. PMLR, 2018.

Rick Durrett. *Probability: theory and examples*, volume 49. Cambridge university press, 2019.

Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102(477):359–378, 2007.

Desmond J Higham. An algorithmic introduction to numerical simulation of stochastic differential equations. *SIAM review*, 43(3):525–546, 2001.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

Patrick Kidger, James Foster, Xuechen Li, and Terry J Lyons. Neural sdes as infinite-dimensional gans. In *International conference on machine learning*, pp. 5453–5463. PMLR, 2021.

Nikolay Malkin, Moksh Jain, Emmanuel Bengio, Chen Sun, and Yoshua Bengio. Trajectory balance: Improved credit assignment in gflownets. *Advances in Neural Information Processing Systems*, 35:5955–5967, 2022.

Xuerong Mao. *Stochastic differential equations and applications*. Elsevier, 2007.

Fionn Murtagh. Multilayer perceptrons for classification and regression. *Neurocomputing*, 2(5-6): 183–197, 1991.

Bernt Oksendal. *Stochastic differential equations: an introduction with applications*. Springer Science & Business Media, 2013.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pp. 593–607. Springer, 2018.

Gábor J Székely and Maria L Rizzo. Energy statistics: A class of statistics based on distances. *Journal of statistical planning and inference*, 143(8):1249–1272, 2013.

Shuo Yu, Hongyan Xue, Xiang Ao, Feiyang Pan, Jia He, Dandan Tu, and Qing He. Generating synergistic formulaic alpha collections via reinforcement learning. In *Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 5476–5486, 2023.

Ningning Zhang and Ke Zhu. Quantiled conditional variance, skewness, and kurtosis by cornish–fisher expansion. *The Annals of Applied Statistics*, 20(1):131–147, 2026.

Zhoufan Zhu and Ke Zhu. Alphaqcm: Alpha discovery in finance with distributional reinforcement learning. In *Forty-second International Conference on Machine Learning*, 2025.

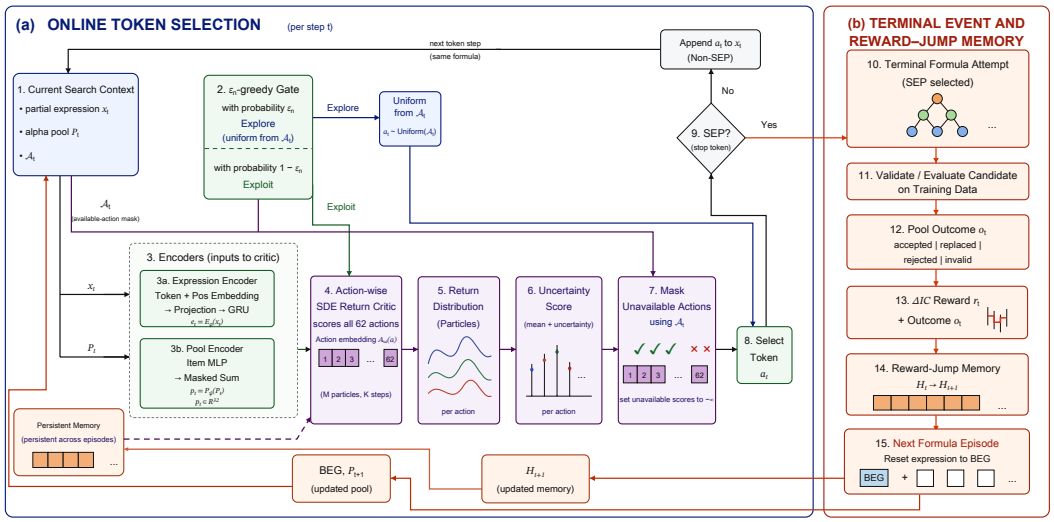


Figure 3: Architecture overview of AlphaRJM. The ϵ_n-greedy policy either samples uniformly from $\mathcal{A}_t$, the action set exposed by the current construction mask, or exploits the action-conditioned SDE return critic using the encoded expression e_t, encoded pool p_t, persistent memory H_t, and candidate action. Terminal evaluations produce reward–outcome feedback (r_t, o_t) that updates Reward-Jump Memory before the next formula episode.

A The Use of Large Language Models (LLMs)

LLMs were used solely as auxiliary tools for language refinement, grammatical correction, improving clarity and presentation, and limited assistance in writing and debugging selected portions of the code. All methodological development, experiments, analyses, results, and scientific conclusions were independently conducted, carefully reviewed, and verified by the authors.

B AlphaRJM Implementation Details

This appendix specifies the implementation corresponding to the methodology in Section 3. Architectural dimensions, symbolic-construction details, numerical SDE integration, exploration schedules, replay, and optimization settings are given here to keep the main text focused on the method itself.

B.1 Architecture Overview

Figure 3 summarizes the complete search loop. At an exploitation step, the partial expression and retained alpha pool are encoded, while the persistent memory H_t directly conditions the action-wise SDE return critic. At an exploration step, the ϵ_n-greedy branch samples directly from the currently available action set $\mathcal{A}_t$. Terminal formula evaluations produce a pool reward r_t and outcome o_t, after which Reward-Jump Memory is updated before construction of the next formula episode.

B.2 Symbolic Construction and Pool Evaluation

AlphaRJM uses the same expression grammar family as the controlled formulaic-alpha benchmark. The selectable vocabulary contains 62 actions, while the beginning-of-expression token (BEG) and padding token (PAD) are input-only. Expressions contain at most

$$L_{\max} = 20$$

tokens in the benchmark state representation. The terminal separator action, denoted SEP, completes the current formula attempt whenever selected.

The available action set $\mathcal{A}_t$ in Section 3.1 is the set exposed by the benchmark construction mask. The mask first applies the expression-builder legality rules. In addition, we reproduce the released AlphaSAGE length-dependent early-stop rule. Whenever SEP is currently available and the maximum length has not been reached, all non-SEP actions are masked with probability

$$p_{\mathrm{stop},t} = p_{\mathrm{mask}} \frac{\ell_t^{\mathrm{state}}}{L_{\max}}, \tag{24}$$

where ℓ_t^{state} is the current state length including BEG. The reported runs use $p_{\mathrm{mask}} = 1$. At the maximum length, SEP is made available unconditionally so that the attempt terminates. The mask is cached for the current state, ensuring that action selection and environment execution use the same realization.

A terminal action increments the formula-episode counter and evaluates the completed expression on the training data. A syntactically or numerically invalid terminal attempt receives outcome $\mathrm{invalid}$ and reward zero. Otherwise, the candidate is considered for inclusion in an alpha pool of capacity 50. Pool quality is the weighted-ensemble training IC $Q(\mathcal{P})$, and the reward is exactly the marginal improvement defined in Eq. 3.

Pool coefficients are jointly optimized using the controlled benchmark mechanism. If the pool is not full, a valid inserted candidate receives outcome $\mathrm{accepted}$. If the pool is full, the candidate and existing members are jointly reweighted. If the candidate obtains the smallest absolute coefficient, it receives outcome $\mathrm{rejected}$ and the pre-candidate pool is restored exactly. Otherwise, the corresponding existing member is removed and the candidate receives outcome $\mathrm{replaced}$. A valid candidate whose IC statistics cannot be used by the pool is likewise treated as $\mathrm{rejected}$.

Thus the reported environment realizes the four outcomes in Eq. 2. A valid accepted or replaced candidate may have a zero-valued incremental reward; no separate "terminal-zero" outcome is emitted by the reported search environment.

B.3 Expression and Pool Encoders

The partial-expression encoder E_ϕ uses learned token and positional embeddings, each of dimension 32. Their concatenation is projected to dimension 32 and passed through a gated recurrent unit (GRU). The final valid hidden state is the expression representation

$$e_t \in \mathbb{R}^{32}. \tag{25}$$

Completed formulas stored in the alpha pool reuse the token embeddings, position embeddings, input projection, and GRU of the partial-expression encoder. Their final hidden states are projected to 24-dimensional formula embeddings. For pool member j, let

$$f_j \in \mathbb{R}^{24}$$

denote this embedding.

The pool-item representation concatenates f_j with five scalar quantities:

$$[f_j;\, w_j;\, \mathrm{IC}_j;\, \ell_j;\, \rho_j;\, m_j], \tag{26}$$

where w_j is the ensemble coefficient, IC_j is the standalone training IC, ℓ_j is the formula token length, ρ_j is the maximum absolute redundancy with the other retained formulas, and $m_j \in \{0, 1\}$ is the pool-slot occupancy mask.

Each occupied pool item is mapped to a 32-dimensional hidden representation. The item representations are summed across the pool, yielding a permutation-invariant aggregate, and a final projection produces

$$p_t \in \mathbb{R}^{32}. \tag{27}$$

A learned 32-dimensional embedding represents the empty pool.

The persistent memory dimension is

$$H_t \in \mathbb{R}^8, \tag{28}$$

and each action has a learned embedding

$$A_\omega(a) \in \mathbb{R}^{16}. \tag{29}$$

Consequently, Eq. 10 has dimension

$$\dim(q_{t,a}) = 32 + 32 + 8 + 16 = 88. \tag{30}$$

B.4 Reward-Jump Memory Architecture

At the beginning of the complete search stream, Eq. 6 maps the initial expression and pool representations to $H_0 \in \mathbb{R}^8$. In the reported implementation this initialization is evaluated once without gradient tracking, and the resulting H_0 is detached before the search begins. Consequently, no training loss is backpropagated through the one-time initialization map (W_0, b_0).

The implementation reserves five coordinates for categorical outcome encoding. The four realized outcomes in Eq. 2 occupy four of these coordinates; the fifth slot corresponds to a reserved terminal-zero code that is not emitted by the reported environment. Thus

$$[r_t; \mathrm{onehot}_5(o_t)] \in \mathbb{R}^6 \tag{31}$$

at a realized terminal event.

The event encoder implements

$$\mathrm{LayerNorm}(6) \to \mathrm{Linear}(6, 96) \to \mathrm{SiLU} \to \mathrm{Linear}(96, 32) \to \tanh,$$

producing

$$\chi_t \in \mathbb{R}^{32}.$$

The eight-dimensional current memory and 32-dimensional event mark are concatenated to form a 40-dimensional jump input. The context network is

$$\mathrm{LayerNorm}(40) \to \mathrm{Linear}(40, 96) \to \mathrm{SiLU} \to \mathrm{Linear}(96, 96) \to \mathrm{SiLU}\,.$$

Separate heads then produce

$$g_t \in (0, 1)^8, \qquad \zeta_t \in (-1, 1)^8.$$

The experiments use

$$J_{\max} = 0.02,$$

so Eq. 8 becomes

$$J_t = 0.02\, g_t \odot \zeta_t, \tag{32}$$

which guarantees

$$\|J_t\|_\infty \le 0.02.$$

The event-driven jump depends on the current memory together with the realized reward–outcome event mark. The expression and pool encodings do not enter Eq. 8 directly. Reward-Jump Memory contains no continuous drift, diffusion, or Brownian component; between terminal events, Eq. 9 leaves H_t exactly unchanged.

B.5 SDE Return Critic

The action-conditioned critic receives the 88-dimensional vector $q_{t,a}$ from Eq. 10. A two-layer SiLU network with hidden width d_{critic} maps this vector to four scalar heads:

$$z_{0,t,a}, \qquad \mu_{t,a}, \qquad \widehat{\kappa}_{t,a}, \qquad \widehat{\sigma}_{t,a}.$$

The positive mean-reversion rate in Eq. 11 is parameterized as

$$\kappa_{t,a} = \mathrm{softplus}(\widehat{\kappa}_{t,a}) + 10^{-4}, \tag{33}$$

and the diffusion scale is

$$\sigma_{t,a} = \sigma_{\min} + (\sigma_{\max} - \sigma_{\min})\,\mathrm{sigmoid}(\widehat{\sigma}_{t,a}), \tag{34}$$

with

$$\sigma_{\min} = 10^{-3}.$$

Eq. 11 is simulated using K Euler–Maruyama steps over $\tau \in [0, 1]$. Define

$$\Delta\tau = \frac{1}{K}. \tag{35}$$

For fixed (t, a), particle m is initialized by

$$Z_0^{(m)} = z_{0,t,a}$$

and evolves as

$$\begin{aligned} Z_{k+1}^{(m)} = Z_k^{(m)} + \kappa_{t,a}\big(\mu_{t,a} - Z_k^{(m)}\big)\Delta\tau \\ + \sigma_{t,a}\sqrt{\Delta\tau}\,\varepsilon_k^{(m)}, \end{aligned} \tag{36}$$

where

$$\varepsilon_k^{(m)} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1), \qquad k = 0, \ldots, K-1, \quad m = 1, \ldots, M.$$

After K integration steps,

$$Z_{t,a}^{(m)} := Z_K^{(m)}$$

is the terminal return particle appearing in Eq. 12. Independent Gaussian increments are used across particles.

B.6 Exploration Strategy

The uncertainty coefficient in Eq. 14 is linearly annealed according to

$$c_n = c_0 + (c_N - c_0)\min\Big(\frac{n}{N}, 1\Big), \qquad c_0 = 1, \quad c_N = 0. \tag{37}$$

The ϵ_n-greedy probability in Eq. 15 follows

$$\epsilon_n = \epsilon_0 + (\epsilon_N - \epsilon_0)\min\Big(\frac{n}{N}, 1\Big), \qquad \epsilon_0 = 1, \quad \epsilon_N = 0.05. \tag{38}$$

The reported runs use $N = 10{,}000$ completed formula episodes.

On the random branch, an action is sampled uniformly from $\mathcal{A}_t$ without invoking the SDE critic. On the exploitation branch, the implementation vectorizes the SDE critic over all 62 selectable actions, computes their particle scores, masks actions not contained in $\mathcal{A}_t$ to $-\infty$, and selects the maximizer.

B.7 Contiguous Replay and Optimization

The replay buffer stores primitive transitions together with the persistent memory present immediately before each action and identifiers for the corresponding current and next pool snapshots. Its capacity is 50,000 transitions.

Training samples contiguous sequences of length

$$L_{\text{seq}} = 16.$$

The first

$$L_{\text{burn}} = 8$$

positions serve as burn-in, after which the remaining eight positions contribute directly to the SDE return-learning objective. Starting from the stored pre-transition memory of each sampled sequence, Eq. 9 is unrolled chronologically so that later learning positions use a memory trajectory reconstructed under the current jump-network parameters.

For completeness, the implementation aggregates jump regularization over all event-bearing positions of the full 16-step memory unroll, including burn-in. Let B_{batch} be the replay batch size, let $\delta_{b,k}$ and $J_{b,k}$ denote the terminal indicator and candidate jump for sequence $b \in \{1, \ldots, B_{\text{batch}}\}$ at replay position k, and define

$$\mathcal{K}_{\text{evt}} = \left\{ k : \sum_{b=1}^{B_{\text{batch}}} \delta_{b,k} > 0 \right\}.$$

When $\mathcal{K}_{\text{evt}} \neq \emptyset$, the implemented jump penalty is

$$\mathcal{L}_{\text{jump}} = \frac{1}{|\mathcal{K}_{\text{evt}}|} \sum_{k \in \mathcal{K}_{\text{evt}}} \frac{\sum_{b=1}^{B_{\text{batch}}} \delta_{b,k} \|J_{b,k}\|_2^2}{\sum_{b=1}^{B_{\text{batch}}} \delta_{b,k}}. \tag{39}$$

If no evaluation event occurs in the sampled unroll, this term is zero.

Replay optimization begins after 2,048 stored primitive transitions. The replay batch size is

$$B_{\text{batch}} = 16,$$

and one gradient update is triggered every four primitive actions.

The discount factor is

$$\gamma = 0.99.$$

Trainable online components are optimized jointly with AdamW and gradient clipping at maximum norm 10. The target SDE return critic is synchronized with the online critic every 50 optimizer updates.

For the default AlphaRJM configuration,

$$\lambda_{\text{ED}} = 1, \qquad \lambda_{\text{mean}} = 0.1, \qquad \beta_J = 10^{-4}. \tag{40}$$

B.8 AlphaRJM Pseudocode

Algorithm 1 summarizes the complete search and optimization procedure using the notation introduced in Section 3.

B.9 AlphaRJM Hyperparameter Settings

Table 3 reports the AlphaRJM settings that vary across the reported universes. Parameters not listed in the table follow the common implementation settings above.

Table 3: AlphaRJM configurations used in the reported experiments. LR denotes the AdamW learning rate, WD denotes weight decay, d_{critic} is the hidden width of the SDE critic, M is the number of return particles, and K is the number of Euler–Maruyama steps.

Preset	**LR**	**WD**	$\mathbf{d}_{\text{critic}}$	**M**	**K**	$\boldsymbol{\lambda}_{\mathbf{ED}}$	$\boldsymbol{\lambda}_{\mathbf{mean}}$	$\boldsymbol{\beta}_{\boldsymbol{J}}$
CSI300/CSI500	3.0×10^{-4}	1.0×10^{-4}	64	16	4	1.0	0.1	1.0×10^{-4}
CSI800/CSI1000	1.442×10^{-4}	2.144×10^{-6}	32	32	4	1.0	0.1	1.0×10^{-4}

The first preset is used for CSI300 and CSI500; the second is used for the reported CSI800 experiment and the CSI1000 appendix experiment. Each $h = 42$ run reuses the corresponding universe's $h = 20$ AlphaRJM configuration rather than introducing long-horizon-specific model tuning. All reported configurations use

$$\sigma_{\max} = 0.20.$$

C Sensitivity Configuration

Sensitivity analysis is conducted on CSI300 with seed 0 using a one-factor-at-a-time protocol. Starting from the default configuration, we vary:

$$M \in \{8, 16, 32\}, \tag{41}$$
$$K \in \{2, 4, 8\}, \tag{42}$$
$$\lambda_{\text{ED}} \in \{0.5, 1.0, 2.0\}, \tag{43}$$
$$\sigma_{\max} \in \{0.10, 0.20, 0.40\}. \tag{44}$$

The default operating point is

$$(M, K, \lambda_{\text{ED}}, \sigma_{\max}) = (16, 4, 1.0, 0.20).$$

All other parameters remain fixed while one quantity is varied.

Algorithm 1: AlphaRJM

Input: Formula-episode budget N
Output: Final alpha pool $\mathcal{P}$

Initialize $\mathcal{P} \leftarrow \emptyset$ and replay buffer $\mathcal{D} \leftarrow \emptyset$
Initialize online model parameters and target SDE critic $\bar{\theta} \leftarrow \theta$
Encode the initial context and initialize H_0 by Eq. 6
while *completed formula episodes* $< N$ **do**
  Compute $e_t \leftarrow E_\phi(x_t)$ and $p_t \leftarrow P_\psi(\mathcal{P}_t)$
  Obtain the current available action set $\mathcal{A}_t$
  if *sample ϵ_n-greedy exploration branch* **then**
    Sample a_t uniformly from $\mathcal{A}_t$
  else
    foreach $a \in \mathcal{A}_t$ **do**
      Generate $\mathcal{Z}_\theta(s_t, a)$ by Eqs. 11–12
      Compute $S_t(a)$ by Eq. 14
    $a_t \leftarrow \arg\max_{a \in \mathcal{A}_t} S_t(a)$
  Execute a_t and observe $(x_{t+1}, \mathcal{P}_{t+1}, r_t, \delta_t)$
  if $\delta_t = 1$ **then**
    Observe terminal outcome o_t
    Compute event mark χ_t by Eq. 7
    Compute J_t by Eq. 8
    $H_{t+1} \leftarrow H_t + J_t$
    Start the next formula episode while retaining H_{t+1}
  else
    $H_{t+1} \leftarrow H_t$
  Store the transition, pre-action memory H_t, and pool-snapshot identifiers in $\mathcal{D}$
  if *replay conditions in Appendix B.7 are satisfied* **then**
    Sample contiguous sequences from $\mathcal{D}$
    Reconstruct the memory trajectory through burn-in
    foreach *learning position in the replay unroll* **do**
      Generate online particles $Z_t^{(m)} \sim \mathcal{Z}_\theta(s_t, a_t)$
      Select $a_{t+1}^\star$ by Eq. 18
      Generate independent target particles $\widetilde{Z}_{t+1}^{(m)} \sim \mathcal{Z}_{\bar{\theta}}(s_{t+1}, a_{t+1}^\star)$
      Form Bellman particles $Y_t^{(m)}$ by Eq. 20
    Compute the objective in Eq. 23
    Update the online model parameters
    Periodically synchronize $\bar{\theta} \leftarrow \theta$
return $\mathcal{P}$

D Experiment Details

This section complements the experimental summary in Section 4. We first specify the common benchmark protocol, then define the reported metrics, and finally describe the baseline implementations and computational environment. AlphaRJM-specific architecture and optimization settings are given in Appendix B, with the universe-specific settings summarized in Table 3.

D.1 Benchmark Protocol

The main benchmark uses CSI300, CSI500, and CSI800; CSI1000 is included as an additional large-universe experiment in Appendix E.2. All methods use the same chronological partitions: January 1, 2010–December 31, 2020 for training, January 1–December 31, 2021 for validation, and January 1, 2022–December 31, 2024 for testing. Training and model selection therefore use no observations from the test period.

Let $C_{i,d}$ denote the closing price of stock i on trading date d. For a forecasting horizon h, the prediction target is

$$y_{i,d}^{(h)} = \frac{C_{i,d+h}}{C_{i,d}} - 1, \tag{45}$$

implemented in the benchmark as `Ref(CLOSE,-h)/CLOSE-1`. The main experiments use $h = 20$, while the long-horizon study changes only the target horizon to $h = 42$.

The formula-discovery methods share the same expression grammar and an alpha pool capacity of 50. AlphaQCM, AlphaSAGE, and AlphaRJM are trained for 10,000 terminal formula episodes. AlphaGen retains its native PPO step-based training and uses 50,000 requested primitive-action steps; because PPO collects complete 2,048-step rollouts, the completed runs contain 51,200 executed primitive actions. We preserve each method's native optimization unit rather than rewriting its training procedure solely to express all budgets in the same counter.

The main $h = 20$ results are aggregated over random seeds $0, 1, 2, 3$. The $h = 42$ comparison uses seed 0, and the CSI1000 appendix table uses seeds 0 and 1. Apart from these explicitly stated reporting choices, the dataset split, target construction, universe definition, pool protocol, and downstream evaluation are shared across compared methods whenever applicable.

D.2 Evaluation Metrics

Let $\widehat{y}_{i,d}^{(h)}$ denote the predicted alpha score corresponding to the target in Eq. 45. For each test date d, the Information Coefficient (IC) and Rank Information Coefficient (RIC) are the cross-sectional Pearson and rank correlations, respectively:

$$\mathrm{IC}_d = \mathrm{Corr}_i\left(\widehat{y}_{i,d}^{(h)}, y_{i,d}^{(h)}\right), \tag{46}$$

$$\mathrm{RIC}_d = \mathrm{Corr}_i\left(\mathrm{rank}(\widehat{y}_{i,d}^{(h)}), \mathrm{rank}(y_{i,d}^{(h)})\right). \tag{47}$$

Here Corr_i is computed across stocks at a fixed date, whereas $\mathbb{E}_d$ and Var_d denote the empirical mean and variance across test dates. The four reported statistics are

$$\mathrm{IC} = \mathbb{E}_d[\mathrm{IC}_d], \qquad \mathrm{ICIR} = \frac{\mathbb{E}_d[\mathrm{IC}_d]}{\sqrt{\mathrm{Var}_d(\mathrm{IC}_d)}}, \tag{48}$$

$$\mathrm{RIC} = \mathbb{E}_d[\mathrm{RIC}_d], \qquad \mathrm{RICIR} = \frac{\mathbb{E}_d[\mathrm{RIC}_d]}{\sqrt{\mathrm{Var}_d(\mathrm{RIC}_d)}}. \tag{49}$$

Higher values indicate stronger predictive performance or temporal stability.

Cumulative wealth. For the complementary portfolio visualization, stocks are ranked by their predicted alpha score on each trading date d, and $\mathcal{T}_d$ denotes the top 20% of valid stocks. Following the evaluation implementation, the daily contribution used for wealth accumulation is

$$R_d = \frac{1}{h}\frac{1}{|\mathcal{T}_d|}\sum_{i\in\mathcal{T}_d} y_{i,d}^{(h)}. \tag{50}$$

Starting from $\mathcal{W}_0 = 1$, cumulative wealth is

$$\mathcal{W}_d = \mathcal{W}_{d-1}(1 + R_d) = \prod_{d'=1}^{d}(1 + R_{d'}). \tag{51}$$

When multiple seeds are shown, the wealth trajectory is computed separately for each seed and the plotted curve is their pointwise mean,

$$\overline{\mathcal{W}}_d = \frac{1}{N_{\mathrm{seed}}}\sum_{\nu=1}^{N_{\mathrm{seed}}} \mathcal{W}_d^{(\nu)}, \tag{52}$$

where N_{seed} is the number of seeds and ν indexes the seed. The same portfolio rule is used for all compared methods.

D.3 Baseline Implementations

The common protocol above is applied without redefining the internal learning mechanism of each baseline. Our unified benchmark was assembled using the public reference implementations of AlphaSAGE, AlphaQCM, and AlphaGen, respectively:[1] For these formula-discovery baselines, method-specific architectures and learner settings are retained from the released implementations, while the shared data, target, grammar, pool, and evaluation protocol follow Appendix D.1.

MLP and LightGBM. The multilayer perceptron and LightGBM baselines use the tabular feature construction of the reference AlphaSAGE benchmark. They provide conventional direct-prediction comparators under the same materialized features, labels, and chronological partitions.

Neural ODE and Neural SDE. These are controlled direct-prediction baselines using the same materialized features, labels, normalization, and train/validation/test partitions as the other predictive models. Their continuous-time dynamics map market features to the prediction target directly; they are therefore distinct from AlphaRJM's SDE return critic, whose state models the distribution of future symbolic-search returns.

AlphaGen. AlphaGen (Yu et al., 2023) is the pool-synergy reinforcement-learning baseline. We retain its PPO-based symbolic search and pool optimization from the reference implementation; only the shared benchmark interface and the training budget specified in Appendix D.1 are imposed.

AlphaQCM. AlphaQCM (Zhu & Zhu, 2025) is the distributional reinforcement-learning baseline. Its released IQN/QCM learner and uncertainty-guided action scoring are retained, while it operates on the common symbolic environment and pool protocol used in our benchmark.

AlphaSAGE. AlphaSAGE (Chen et al., 2026) is the structure- and diversity-aware formula-discovery baseline. We retain its graph-based expression representation and GFlowNet-oriented search components from the public implementation under the same benchmark protocol.

D.4 Computational Environment

All reported experiments were executed locally on CPU using an Apple Mac mini with an Apple M4 processor, 16 GB unified memory, and a 256 GB SSD. The reproducibility environment uses Python 3.12.13, PyTorch 2.13.0, Qlib 0.9.8.dev32, LightGBM 4.7.0, scikit-learn 1.9.0, NumPy 2.5.1, and pandas 2.3.3.

E Additional Results

E.1 Ablation Study

Table 4 isolates the three components highlighted in the main text. The *w/o SDE Return Critic* variant replaces the SDE return-distribution critic with the QCM/IQN return critic while retaining event-conditioned memory, and produces the largest decline in both IC and ICIR. The *w/o Persistent Memory* variant removes recurrent use of historical memory and also lowers both metrics. The *w/o Distributional Loss* variant removes the energy-distance term while retaining the SDE dynamics and mean-calibration objective; it causes a moderate reduction in IC and a larger decrease in ICIR. Overall, the full model achieves the highest IC and ICIR, supporting the complementary roles of the SDE return critic, distributional supervision, and persistent memory.

[1] `https://github.com/BerkinChen/AlphaSAGE`, `https://github.com/ZhuZhouFan/AlphaQCM`, and `https://github.com/RL-MLDM/alphagen`.

Table 4: Ablation study of **AlphaRJM** on CSI300 using random seed 0. Higher values are better. Best and second-best results are shown in **bold** and underlined, respectively.

Model	IC	ICIR
AlphaRJM	**0.0480**	**0.3700**
w/o SDE Return Critic	0.0362	0.2687
w/o Distributional Loss	0.0453	0.3085
w/o Persistent Memory	0.0434	0.3053

E.2 Additional Results on CSI1000

To further evaluate AlphaRJM on a broader equity universe, we report additional results on CSI1000 using the same chronological data split, prediction target, evaluation metrics, and controlled comparison protocol described in Appendix D.1. Results are reported over random seeds 0 and 1. As shown in Table 5, AlphaRJM achieves the highest IC and RIC on CSI1000, reaching 0.076 ± 0.003 and 0.102 ± 0.009, respectively. It also obtains the second-highest RICIR (0.648 ± 0.013), while AlphaQCM and LightGBM achieve the strongest ICIR and RICIR, respectively. These results provide additional evidence that the predictive advantage of AlphaRJM is retained when the evaluation is extended to the larger CSI1000 universe, particularly for the cross-sectional correlation metrics IC and RIC.

Table 5: Performance comparison on CSI1000 using correlation-based evaluation metrics. Results are reported as mean $\pm$ standard deviation over two random seeds (0–1). Higher values are better for all metrics. Best and second-best performances are shown in **bold** and underlined, respectively.

Method	IC	ICIR	RIC	RICIR
MLP	0.049 ± 0.004	0.464 ± 0.034	0.063 ± 0.008	0.556 ± 0.076
LightGBM	0.053 ± 0.000	0.556 ± 0.006	0.063 ± 0.000	$\mathbf{0.668 \pm 0.027}$
Neural ODE	0.054 ± 0.001	0.513 ± 0.020	0.066 ± 0.002	0.585 ± 0.008
Neural SDE	0.054 ± 0.000	0.510 ± 0.000	0.063 ± 0.001	0.574 ± 0.006
AlphaGen	0.072 ± 0.010	0.513 ± 0.038	0.089 ± 0.011	0.627 ± 0.035
AlphaQCM	0.070 ± 0.008	$\mathbf{0.563 \pm 0.002}$	0.078 ± 0.016	0.610 ± 0.080
AlphaSAGE	0.056 ± 0.007	0.517 ± 0.032	0.068 ± 0.004	0.579 ± 0.018
AlphaRJM (ours)	$\mathbf{0.076 \pm 0.003}$	0.505 ± 0.003	$\mathbf{0.102 \pm 0.009}$	0.648 ± 0.013

E.3 Long-Horizon Cumulative Wealth on CSI500.

Figure 4 shows the cumulative wealth trajectories on CSI500 for the $h = 42$ forecasting horizon using random seed 0. AlphaRJM maintains the highest wealth trajectory for most of the test period and shows a strong recovery following the market decline in mid-2024. By the end of the evaluation period, AlphaRJM remains above the other learned strategies, while the CSI500 index remains substantially below its initial level. Together with the corresponding IC and RIC results, this visualization provides additional evidence that the long-horizon predictive signals discovered by AlphaRJM translate into persistent cross-sectional portfolio performance. Since the figure is based on a single random seed, it is presented as a complementary illustration rather than as a separate statistical comparison.

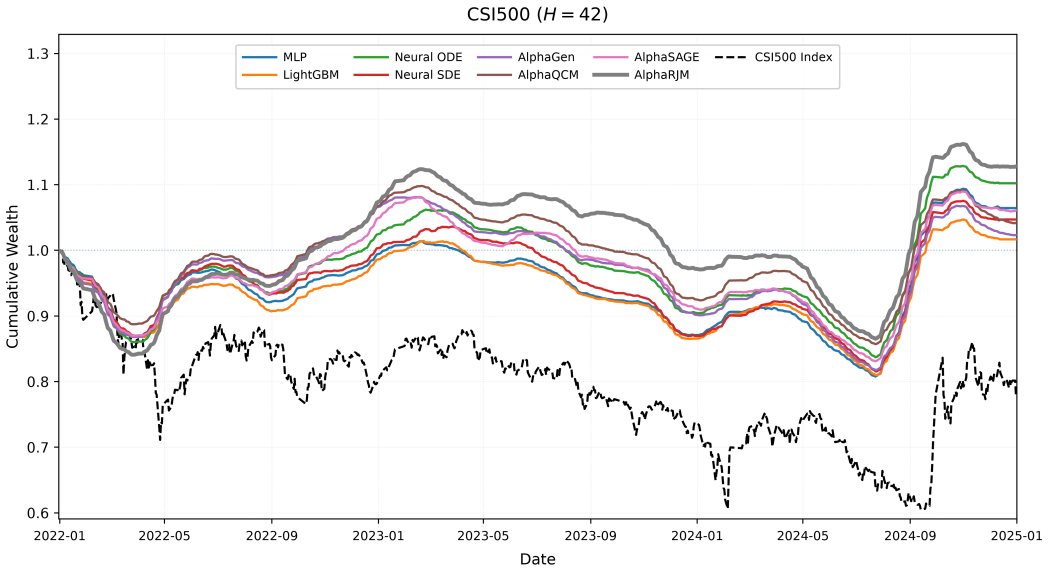


Figure 4: Cumulative wealth curves on CSI500 for the $h = 42$ forecasting horizon using random seed 0.

F Theoretical Properties of AlphaRJM

This appendix establishes theoretical properties of the two central components of AlphaRJM. We first show that Reward-Jump Memory is exactly constant between terminal evaluation events and has bounded variation over finite search trajectories. We then characterize the conditional law of the action-conditioned SDE return critic using classical results for linear stochastic differential equations (Oksendal, 2013; Mao, 2007). Finally, we derive the moments of the finite-step Euler–Maruyama particles used by the implementation (Higham, 2001) and establish consistency of their empirical statistics using the strong law of large numbers (Durrett, 2019).

F.1 Event-Driven Memory Dynamics

The following result formalizes the event-driven persistence of Reward-Jump Memory and bounds its variation over a finite search trajectory. Here $\|\cdot\|_\infty$ denotes the vector maximum norm.

Proposition F.1 (Event-Driven Persistence and Bounded Memory Variation)**.** *For Reward-Jump Memory defined by Eqs. 8–9,*

$$H_{t+1} = H_t \qquad \textit{whenever} \qquad \delta_t = 0. \tag{53}$$

Whenever $\delta_t = 1$,

$$\|H_{t+1} - H_t\|_\infty \le J_{\max}. \tag{54}$$

Moreover, for every integer $T \ge 1$, *the cumulative variation satisfies*

$$\sum_{t=0}^{T-1} \|H_{t+1} - H_t\|_\infty \le J_{\max} \sum_{t=0}^{T-1} \delta_t. \tag{55}$$

Consequently,

$$\|H_T - H_0\|_\infty \le J_{\max} \sum_{t=0}^{T-1} \delta_t. \tag{56}$$

Thus Reward-Jump Memory is exactly constant between terminal evaluation events, while both its cumulative variation and net displacement over any finite search prefix are controlled by the number of such events.

Proof. When $\delta_t = 0$, Eq. 9 gives

$$H_{t+1} = H_t,$$

and therefore

$$H_{t+1} - H_t = 0,$$

which proves Eq. 53.

Now suppose that $\delta_t = 1$. Then Eq. 9 gives

$$H_{t+1} - H_t = J_t.$$

From Eq. 8,

$$J_t = J_{\max}\, g_t \odot \zeta_t.$$

Since g_t is obtained componentwise through a sigmoid function and ζ_t through a hyperbolic tangent, for every coordinate i,

$$0 < (g_t)_i < 1, \qquad -1 < (\zeta_t)_i < 1.$$

Hence

$$\|g_t\|_\infty \le 1, \qquad \|\zeta_t\|_\infty \le 1.$$

Moreover,

$$\begin{aligned}\|g_t \odot \zeta_t\|_\infty &= \max_i |(g_t)_i(\zeta_t)_i| \\ &\le \left(\max_i |(g_t)_i|\right)\left(\max_i |(\zeta_t)_i|\right) \\ &= \|g_t\|_\infty \|\zeta_t\|_\infty \le 1.\end{aligned}$$

Therefore,

$$\begin{aligned}\|J_t\|_\infty &= J_{\max}\|g_t \odot \zeta_t\|_\infty \\ &\le J_{\max}.\end{aligned}$$

Since $H_{t+1} - H_t = J_t$ when $\delta_t = 1$, we obtain

$$\|H_{t+1} - H_t\|_\infty \le J_{\max},$$

which proves Eq. 54.

Combining the cases $\delta_t = 0$ and $\delta_t = 1$ gives, for every t,

$$\|H_{t+1} - H_t\|_\infty \le J_{\max}\delta_t.$$

Summing over $t = 0, \dots, T-1$ yields

$$\sum_{t=0}^{T-1} \|H_{t+1} - H_t\|_\infty \le J_{\max} \sum_{t=0}^{T-1} \delta_t,$$

which establishes Eq. 55.

Finally, by telescoping,

$$H_T - H_0 = \sum_{t=0}^{T-1} (H_{t+1} - H_t).$$

Applying the triangle inequality for the ℓ_∞ norm,

$$\begin{aligned}\|H_T - H_0\|_\infty &= \left\| \sum_{t=0}^{T-1} (H_{t+1} - H_t) \right\|_\infty \\ &\le \sum_{t=0}^{T-1} \|H_{t+1} - H_t\|_\infty \\ &\le J_{\max} \sum_{t=0}^{T-1} \delta_t.\end{aligned}$$

Thus Eq. 56 follows. □

Proposition F.1 formalizes the separation between termination of an individual formula episode and termination of the longer memory stream. A terminal evaluation may modify H_t, but the resulting state is retained while subsequent formulas are constructed.

F.2 SDE RETURN CRITIC

For a fixed search state s_t and available action $a \in \mathcal{A}_t$, the process in Eq. 11 is a scalar linear mean-reverting diffusion. The following result specializes classical linear-SDE theory (Oksendal, 2013; Mao, 2007) to the return critic used by AlphaRJM.

Theorem F.2 (Well-Posedness and Conditional Law of the SDE Return Critic)**.** *For any fixed search state s_t and available action $a \in \mathcal{A}_t$, the SDE return critic in Eq. 11 admits a unique strong solution on $\tau \in [0,1]$. The solution is*

$$Z_\tau^{t,a} = \mu_{t,a} + \big(z_{0,t,a} - \mu_{t,a}\big)e^{-\kappa_{t,a}\tau} + \sigma_{t,a}\int_0^\tau e^{-\kappa_{t,a}(\tau-\tau')}\,dB_{\tau'}. \tag{57}$$

Consequently,

$$\mathbb{E}\left[Z_\tau^{t,a} \mid s_t, a\right] = \mu_{t,a} + \big(z_{0,t,a} - \mu_{t,a}\big)e^{-\kappa_{t,a}\tau}, \tag{58}$$

and

$$\mathrm{Var}\left(Z_\tau^{t,a} \mid s_t, a\right) = \frac{\sigma_{t,a}^2}{2\kappa_{t,a}}\left(1 - e^{-2\kappa_{t,a}\tau}\right). \tag{59}$$

For every $\tau \in (0,1]$,

$$Z_\tau^{t,a} \mid (s_t,a) \sim \mathcal{N}\Bigg(\mu_{t,a} + \big(z_{0,t,a} - \mu_{t,a}\big)e^{-\kappa_{t,a}\tau}, \frac{\sigma_{t,a}^2}{2\kappa_{t,a}}\left(1 - e^{-2\kappa_{t,a}\tau}\right)\Bigg). \tag{60}$$

At $\tau = 0$, the process reduces to the deterministic initial value $Z_0^{t,a} = z_{0,t,a}$. In particular, conditional on (s_t,a), $Z_\tau^{t,a}$ has finite moments of every finite order for all $\tau \in [0,1]$.

Proof. Fix (s_t,a). Conditional on this state–action pair, the critic outputs $z_{0,t,a}$, $\mu_{t,a}$, $\kappa_{t,a}$, and $\sigma_{t,a}$ are fixed finite scalars. By construction,

$$\kappa_{t,a} > 0, \qquad 0 < \sigma_{\min} \le \sigma_{t,a} \le \sigma_{\max} < \infty.$$

Define the drift and diffusion coefficients by

$$b_{t,a}(z) = \kappa_{t,a}(\mu_{t,a} - z), \qquad \varsigma_{t,a}(z) = \sigma_{t,a}.$$

For any $z, z' \in \mathbb{R}$,

$$|b_{t,a}(z) - b_{t,a}(z')| = \kappa_{t,a}|z - z'|,$$

so the drift is globally Lipschitz. Moreover,

$$|b_{t,a}(z)| \le \kappa_{t,a}\big(|\mu_{t,a}| + |z|\big),$$

and hence it satisfies a linear-growth bound. Since the diffusion coefficient is constant in z,

$$|\varsigma_{t,a}(z) - \varsigma_{t,a}(z')| = 0,$$

and its boundedness implies the corresponding linear-growth condition. Therefore, standard existence and uniqueness results for stochastic differential equations imply that Eq. 11 admits a unique strong solution on $[0,1]$ (Oksendal, 2013; Mao, 2007).

To obtain the solution explicitly, rewrite Eq. 11 as

$$dZ_\tau^{t,a} + \kappa_{t,a}Z_\tau^{t,a}\,d\tau = \kappa_{t,a}\mu_{t,a}\,d\tau + \sigma_{t,a}\,dB_\tau.$$

For fixed (s_t,a), $\kappa_{t,a}$ is constant with respect to the internal diffusion coordinate τ. Using the integrating factor $e^{\kappa_{t,a}\tau}$ and Itô's product rule,

$$\begin{aligned} d\big(e^{\kappa_{t,a}\tau}Z_\tau^{t,a}\big) &= e^{\kappa_{t,a}\tau}\,dZ_\tau^{t,a} + \kappa_{t,a}e^{\kappa_{t,a}\tau}Z_\tau^{t,a}\,d\tau \\ &= \kappa_{t,a}\mu_{t,a}e^{\kappa_{t,a}\tau}\,d\tau + \sigma_{t,a}e^{\kappa_{t,a}\tau}\,dB_\tau, \end{aligned}$$

where the terms involving $Z_\tau^{t,a}$ cancel.

Integrating over $[0, \tau]$ and using $Z_0^{t,a} = z_{0,t,a}$ gives

$$\begin{aligned} e^{\kappa_{t,a}\tau} Z_\tau^{t,a} &= z_{0,t,a} + \kappa_{t,a}\mu_{t,a} \int_0^\tau e^{\kappa_{t,a}\tau'} \, d\tau' \\ &\quad + \sigma_{t,a} \int_0^\tau e^{\kappa_{t,a}\tau'} \, dB_{\tau'} \\ &= z_{0,t,a} + \mu_{t,a} \left(e^{\kappa_{t,a}\tau} - 1\right) + \sigma_{t,a} \int_0^\tau e^{\kappa_{t,a}\tau'} \, dB_{\tau'}. \end{aligned}$$

Multiplying by $e^{-\kappa_{t,a}\tau}$ and rearranging yields

$$\begin{aligned} Z_\tau^{t,a} &= \mu_{t,a} + (z_{0,t,a} - \mu_{t,a}) \, e^{-\kappa_{t,a}\tau} \\ &\quad + \sigma_{t,a} \int_0^\tau e^{-\kappa_{t,a}(\tau - \tau')} \, dB_{\tau'}, \end{aligned}$$

which is Eq. 57.

Conditional on (s_t, a), the integrand $e^{-\kappa_{t,a}(\tau-\tau')}$ is deterministic. Hence

$$\int_0^\tau e^{-\kappa_{t,a}(\tau-\tau')} \, dB_{\tau'}$$

is a Gaussian random variable with mean zero. Taking conditional expectation in Eq. 57 therefore gives

$$\mathbb{E}\left[Z_\tau^{t,a} \mid s_t, a\right] = \mu_{t,a} + (z_{0,t,a} - \mu_{t,a}) \, e^{-\kappa_{t,a}\tau},$$

which proves Eq. 58.

Since the deterministic terms do not contribute to the conditional variance, Itô isometry gives

$$\begin{aligned} \mathrm{Var}\left(Z_\tau^{t,a} \mid s_t, a\right) &= \sigma_{t,a}^2 \int_0^\tau e^{-2\kappa_{t,a}(\tau-\tau')} \, d\tau' \\ &= \sigma_{t,a}^2 \int_0^\tau e^{-2\kappa_{t,a}u} \, du \\ &= \frac{\sigma_{t,a}^2}{2\kappa_{t,a}} \left(1 - e^{-2\kappa_{t,a}\tau}\right), \end{aligned}$$

where $u = \tau - \tau'$. This proves Eq. 59.

Finally, conditional on (s_t, a), Eq. 57 is the sum of deterministic terms and a Gaussian Itô integral. Therefore $Z_\tau^{t,a} \mid (s_t, a)$ is Gaussian with the mean and variance derived above, which establishes Eq. 60. For $\tau > 0$, all finite moments follow from Gaussianity, while at $\tau = 0$ the process equals the deterministic initial value $z_{0,t,a}$. □

Theorem F.2 gives a direct interpretation of the critic parameters. Conditional on (s_t, a), the mean moves from $z_{0,t,a}$ toward $\mu_{t,a}$ at a rate controlled by $\kappa_{t,a}$, while $\sigma_{t,a}$ determines stochastic dispersion. These quantities characterize the internal reinforcement-learning return used for symbolic search and do not describe an asset-price process.

F.3 Euler–Maruyama Particles and Action Scoring

The implementation samples the return process using the Euler–Maruyama scheme in Eq. 36 (Higham, 2001). The next result characterizes the finite-K terminal particles actually used in the action score of Eq. 14.

Corollary F.3 (Euler–Maruyama Particle Moments and Score Consistency)**.** *Fix a search state s_t, an available action $a \in \mathcal{A}_t$, formula episode n (and hence c_n), and an integer number of Euler–Maruyama integration steps $K \geq 1$. Under this fixed conditioning, consider independent Euler–Maruyama trajectories generated according to Eq. 36, and let $Z_{t,a}^{(m)} = Z_K^{(m)}$ denote the terminal*

state of trajectory m. Then the terminal particles are independent and identically distributed, and each is conditionally Gaussian with

$$\mathbb{E}\left[Z_{t,a}^{(m)} \mid s_t, a\right] = \mu_{t,a} + \left(z_{0,t,a} - \mu_{t,a}\right)\left(1 - \frac{\kappa_{t,a}}{K}\right)^K, \tag{61}$$

and

$$\mathrm{Var}\left(Z_{t,a}^{(m)} \mid s_t, a\right) = \frac{\sigma_{t,a}^2}{K} \sum_{k=0}^{K-1} \left(1 - \frac{\kappa_{t,a}}{K}\right)^{2k}. \tag{62}$$

For the particle statistics $\overline{Z}_{t,a}$ and $\widehat{V}_{t,a}$ defined in Eq. 13,

$$\overline{Z}_{t,a} \xrightarrow[M\to\infty]{\text{a.s.}} \mathbb{E}\left[Z_{t,a}^{(m)} \mid s_t, a\right], \tag{63}$$

and

$$\widehat{V}_{t,a} \xrightarrow[M\to\infty]{\text{a.s.}} \mathrm{Var}\left(Z_{t,a}^{(m)} \mid s_t, a\right). \tag{64}$$

Consequently,

$$\begin{aligned} S_t(a) \xrightarrow[M\to\infty]{\text{a.s.}} \mu_{t,a} + \left(z_{0,t,a} - \mu_{t,a}\right)\left(1 - \frac{\kappa_{t,a}}{K}\right)^K \\ + c_n\left[\frac{\sigma_{t,a}^2}{K}\sum_{k=0}^{K-1}\left(1 - \frac{\kappa_{t,a}}{K}\right)^{2k} + \varepsilon_{\text{num}}\right]^{1/2}. \end{aligned} \tag{65}$$

Proof. Fix (s_t, a) and K. Using $\Delta\tau = 1/K$ in Eq. 36, the Euler–Maruyama recursion becomes

$$Z_{k+1}^{(m)} = Z_k^{(m)} + \frac{\kappa_{t,a}}{K}\left(\mu_{t,a} - Z_k^{(m)}\right) + \frac{\sigma_{t,a}}{\sqrt{K}}\varepsilon_k^{(m)},$$

where $\varepsilon_k^{(m)} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1)$. Subtracting $\mu_{t,a}$ from both sides gives

$$Z_{k+1}^{(m)} - \mu_{t,a} = \left(1 - \frac{\kappa_{t,a}}{K}\right)\left(Z_k^{(m)} - \mu_{t,a}\right) + \frac{\sigma_{t,a}}{\sqrt{K}}\varepsilon_k^{(m)}.$$

For brevity, define

$$\rho_{t,a} := 1 - \frac{\kappa_{t,a}}{K}.$$

Iterating the recursion from $Z_0^{(m)} = z_{0,t,a}$ yields

$$\begin{aligned} Z_K^{(m)} - \mu_{t,a} = \rho_{t,a}^K\left(z_{0,t,a} - \mu_{t,a}\right) \\ + \frac{\sigma_{t,a}}{\sqrt{K}}\sum_{k=0}^{K-1}\rho_{t,a}^{K-1-k}\varepsilon_k^{(m)}. \end{aligned}$$

Equivalently,

$$\begin{aligned} Z_K^{(m)} = \mu_{t,a} + \rho_{t,a}^K\left(z_{0,t,a} - \mu_{t,a}\right) \\ + \frac{\sigma_{t,a}}{\sqrt{K}}\sum_{k=0}^{K-1}\rho_{t,a}^{K-1-k}\varepsilon_k^{(m)}. \end{aligned}$$

Conditional on (s_t, a), all coefficients in this expression are fixed. Since the variables $\varepsilon_k^{(m)}$ are independent standard Gaussians, their weighted sum is Gaussian. Hence $Z_K^{(m)} \mid (s_t, a)$ is Gaussian. Moreover, $\mathbb{E}[\varepsilon_k^{(m)}] = 0$, so

$$\begin{aligned} \mathbb{E}\left[Z_K^{(m)} \mid s_t, a\right] &= \mu_{t,a} + \rho_{t,a}^K\left(z_{0,t,a} - \mu_{t,a}\right) \\ &= \mu_{t,a} + \left(1 - \frac{\kappa_{t,a}}{K}\right)^K\left(z_{0,t,a} - \mu_{t,a}\right), \end{aligned}$$

which proves Eq. 61.

The deterministic terms do not contribute to the conditional variance. Using independence and $\mathrm{Var}(\varepsilon_k^{(m)}) = 1$,

$$\begin{aligned}\mathrm{Var}\left(Z_K^{(m)} \mid s_t, a\right) &= \frac{\sigma_{t,a}^2}{K} \sum_{k=0}^{K-1} \rho_{t,a}^{2(K-1-k)} \\ &= \frac{\sigma_{t,a}^2}{K} \sum_{j=0}^{K-1} \rho_{t,a}^{2j} \\ &= \frac{\sigma_{t,a}^2}{K} \sum_{j=0}^{K-1} \left(1 - \frac{\kappa_{t,a}}{K}\right)^{2j},\end{aligned}$$

where the second equality follows from the reindexing $j = K-1-k$. This establishes Equation (62).

For fixed (s_t, a) and K, the terminal particles $Z_{t,a}^{(1)}, \ldots, Z_{t,a}^{(M)}$ are conditionally independent and identically distributed because each is generated from the same state–action-conditioned critic parameters using an independent noise trajectory. Their conditional distribution is Gaussian by the preceding derivation, and therefore

$$\mathbb{E}\left[\left(Z_{t,a}^{(m)}\right)^2 \mid s_t, a\right] < \infty.$$

Hence, by the strong law of large numbers (Durrett, 2019),

$$\overline{Z}_{t,a} = \frac{1}{M} \sum_{m=1}^{M} Z_{t,a}^{(m)} \xrightarrow{\text{a.s.}} \mathbb{E}\left[Z_{t,a}^{(m)} \mid s_t, a\right],$$

which proves Eq. 63.

Since the conditional second moment is finite, the strong law also applies to $\left(Z_{t,a}^{(m)}\right)^2$, giving

$$\frac{1}{M} \sum_{m=1}^{M} \left(Z_{t,a}^{(m)}\right)^2 \xrightarrow{\text{a.s.}} \mathbb{E}\left[\left(Z_{t,a}^{(m)}\right)^2 \mid s_t, a\right].$$

From Eq. 13,

$$\widehat{V}_{t,a} = \frac{1}{M} \sum_{m=1}^{M} \left(Z_{t,a}^{(m)}\right)^2 - \overline{Z}_{t,a}^2.$$

Since

$$\overline{Z}_{t,a}^2 \xrightarrow{\text{a.s.}} \left(\mathbb{E}\left[Z_{t,a}^{(m)} \mid s_t, a\right]\right)^2,$$

it follows that

$$\begin{aligned}\widehat{V}_{t,a} \xrightarrow{\text{a.s.}} &\mathbb{E}\left[\left(Z_{t,a}^{(m)}\right)^2 \mid s_t, a\right] \\ &- \left(\mathbb{E}\left[Z_{t,a}^{(m)} \mid s_t, a\right]\right)^2 \\ =\ &\mathrm{Var}\left(Z_{t,a}^{(m)} \mid s_t, a\right),\end{aligned}$$

which proves Eq. 64.

Finally, for fixed n, define

$$f(z, v) = z + c_n \sqrt{v + \varepsilon_{\text{num}}}.$$

Since $\varepsilon_{\text{num}} > 0$, the mapping $f : \mathbb{R} \times [0, \infty) \to \mathbb{R}$ is continuous. Therefore, by Eqs. 63 and (64) and continuity of f,

$$\begin{aligned}S_t(a) &= f\left(\overline{Z}_{t,a}, \widehat{V}_{t,a}\right) \\ &\xrightarrow{\text{a.s.}} \mathbb{E}\left[Z_{t,a}^{(m)} \mid s_t, a\right] + c_n \sqrt{\mathrm{Var}\left(Z_{t,a}^{(m)} \mid s_t, a\right) + \varepsilon_{\text{num}}}.\end{aligned}$$

Using Eq. 14, this is precisely Eq. 65. □

Corollary F.3 shows that, for fixed K, the empirical particle mean and variance consistently estimate the corresponding moments of the finite-step Euler–Maruyama return distribution. Their Monte Carlo estimation errors vanish almost surely as $M \to \infty$, while the underlying finite-K discretized return process remains unchanged.