Electronic Research Archive

https://www.aimspress.com/journal/era

ERA, 34(7): 4359–4386.
DOI: 10.3934/era.2026193
Received: 04 March 2026
Revised: 28 April 2026
Accepted: 07 May 2026
Published: 20 May 2026

Research article

An efficient model selection method for Lotka-Volterra systems based on ABC-PMC and reinforcement learning

Menghan Zheng, Lu Yu, Siyu Wang, Xiaoyi Xing, Jinyun Pan and Wei Gu*

School of Statistics and Mathematics, Zhongnan University of Economics and Law, Wuhan 430073, China

* **Correspondence:** Email: wei_gu@zuel.edu.cn.

Abstract: This study provides an efficient, adaptive computational framework for Bayesian model selection when likelihood functions are difficult to handle. To address model selection challenges in scenarios where complex model likelihood functions are difficult to handle, this paper proposes an adaptive approximate Bayesian computation with probability-minimizing computation (RL(Softmax)-ABC-PMC) algorithm that integrates a reinforcement learning (RL) Softmax decision mechanism. This method embeds Softmax agents into the standard ABC-PMC framework, dynamically evaluating the historical performance of candidate models to adaptively focus computational resources on potentially optimal models, thereby significantly enhancing selection efficiency. A numerical experiment is provided to select the best model from three Lotka-Volterra (LV) competition models, including a discretized LV model, a randomized LV model, and a fractional-order LV model, where the algorithm effectively distinguishes subtle differences in their fitting ability and yields reliable parameter estimates. Additionally, we further validate the efficiency of the provided algorithm by selecting the best model from three S-shaped growth curve models. Finally, an empirical analysis using Chinese automotive market sales data further validates the practicality and effectiveness of the proposed method in real-world applications, demonstrating that the RL(Softmax)-ABC-PMC algorithm can automatically identify model structures with stronger explanatory power while maintaining high computational efficiency.

Keywords: model selection; approximate Bayesian computation; LV model; reinforcement learning

1. Introduction

In modern scientific exploration, Bayesian inference provides a unified and principled probabilistic framework for model parameter estimation and model selection [1, 2]. However, we have to consider noise in complex systems, such as the importance of fluctuations that has been emphasized in modern

statistical physics (G. Parisi's (2023) Nobel lecture on the interplay of disorder and fluctuations in physical systems. [3]), and it is hard to obtain the likelihood function of such complex models. The application of traditional Bayesian methods faces fundamental limitations. To overcome this obstacle, approximate Bayesian computation (ABC) emerged. By approximating the posterior distribution through comparing distances between simulated and observed data, ABC has become an indispensable methodology in the field of likelihood-free inference [4, 5].

ABC methods have evolved from inefficient to highly efficient. The most basic implementation, rejection sampling, suffers from unacceptable efficiency. This happens when problems have high-dimensional parameter spaces or require strict tolerance thresholds, leading to extremely low acceptance rates [6,7]. To mitigate this issue, ABC algorithms based on sequential Monte Carlo (SMC) ideas were subsequently developed. The pioneering work by Sisson et al. first introduced SMC into ABC, using iteratively updated particle swarm to progressively approximate the target posterior [8]. Building upon this foundation, Toni et al. further refined this approach, proposing the influential ABC-PMC (population Monte Carlo) framework. This framework not only achieves high efficiency in parameter inference but also provides a unified computational scheme for Bayesian model selection [9]. Del Moral et al. provided a rigorous convergence analysis for adaptive sequential Monte Carlo methods in the ABC context, demonstrating that the algorithm converges to the true posterior distribution under mild regularity conditions on the summary statistics and tolerance schedule [10]. Frazier et al. established posterior consistency and quantified the rate of posterior concentration for ABC, showing that the approximate posterior distribution concentrates on the true parameter value as the sample size increases and the tolerance shrinks to zero at an appropriate rate [11]. Drovandi and Frazier provided a detailed empirical guide for practitioners selecting appropriate inference algorithms for complex implicit models and highlighted the future potential of full-data likelihood-free methods [12].

In recent years, deep learning techniques have opened new pathways for likelihood-free inference. Neural networks are increasingly used as surrogate models to accelerate ABC by learning the mapping from parameters to summary statistics, potentially reducing the number of expensive simulations required. Wiqvist et al. considered a sequential neural posterior and likelihood approximation (SNPLA) algorithm [13]. Lueckmann et al. systematically compared classical ABC methods with recent neural network-based density and ratio estimation approaches (e.g., neural posterior estimation (NPE), neural likelihood estimation (NLE), neural ratio estimation (NRE), and their sequential variants) [14]. Cranmer et al. reviewed the transformative advancements in the field of simulation-based inference (SBI), highlighting its profound transformation driven by three core forces: machine learning innovations in handling high-dimensional data and constructing surrogate models, active learning to enhance sample efficiency, and deeper integration with simulators (including leveraging internal information) [15]. Wang et al. proposed preconditioned NPE (PNPE) and its sequential version (PSNPE), which integrate a brief ABC pre-conditioning step to efficiently filter out parameter space regions that produce significant discrepancies between simulations and data [16]. Meanwhile, Gaussian processes could be employed to model the discrepancy function between simulated and observed data, offering a probabilistic estimate of where the posterior mass lies without exhaustive sampling [17, 18]. Shi innovatively integrated a machine learning method (such as genetic algorithms) for parameter optimization of LV models [19]. Dinh et al. proposed a novel RF-ABC method, combining distributional random forests (RFs) for direct joint posterior inference and embedding this RF approach into an SMC regime for iterative posterior refinement on high-likelihood regions [20].

However, despite these advances across both ABC and neural-based (or machine learning) approaches, a fundamental challenge remains underexplored in the context of model selection: the adaptive allocation of computational resources among candidate models. One of the main goals of this paper is model selection when likelihood functions are difficult to handle.

The standard ABC-PMC implementation assigns a static, equal proposal probability to all candidate models at each iteration, failing to leverage valuable information about model fit accumulated dynamically during inference [21, 22]. This static allocation mechanism persistently consumes computational resources on suboptimal models with poor fit to observational data, significantly delaying the algorithm's convergence toward the most plausible model. This challenge is not isolated but has been validated across multiple application domains. Researchers widely recognize that effectively allocating computational resources adaptively within large model spaces constitutes a core difficulty for ABC model selection [23, 24].

To achieve intelligent adaptive resource allocation, recent research has integrated decision theory with the ABC framework. A key theoretical approach formalizes this problem as a multi-armed bandit (MAB) problem [25]. The core objective is to find an optimal balance between "exploration" and "exploitation" to minimize the total computational cost required to identify the optimal model [26]. At the practical level, a compelling attempt comes from Ritto et al., who successfully embedded a complex reinforcement learning algorithm (Q-learning) into ABC to train agents in learning model selection strategies, demonstrating the immense potential of adaptive methods in accelerating the model calibration process [27]. Recent research has explored the theoretical properties and practical applications of adaptive ABC methods. Simola et al. proposed an adaptive sequential ABC framework that achieves further computational efficiency gains by dynamically adjusting tolerance thresholds and sampling strategies [28]. Concurrently, Zhao et al. integrated deep reinforcement learning with ABC to develop intelligent systems capable of autonomously learning optimal resource allocation strategies, demonstrating outstanding performance in complex model selection tasks [29].

Against this research backdrop, this paper proposes a novel adaptive scheme that balances efficiency with implementation simplicity. While complex reinforcement learning frameworks have proven effective [27, 29], their high implementation complexity and additional hyper-parameter tuning costs may hinder widespread adoption. Our core innovation lies in treating the classic Softmax functional standard tool widely used in machine learning and reinforcement learning to transform a set of real-valued scores into a probability distribution for the "exploration-exploitation" trade off [26] as a lightweight decision agent. We seamlessly embed this function into the efficient ABC-PMC framework [9]. Our proposed approach dynamically recalculates and reassigns model selection proposal probabilities at the start of each iteration based on each model's cumulative past performance. This design not only directly addresses the sub-optimality of static resource allocation in standard ABC-PMC [21,22] but also serves as an efficient, robust, and computationally inexpensive algorithmic implementation of the MAB theoretical framework [12]. Compared to the latest deep learning approaches [14, 25], our method maintains computational efficiency advantages while avoiding the additional computational overhead and hyper-parameter sensitivity required for neural network training, providing a more practical and interpretable solution for real-world applications.

The organizational structure of this paper is as follows: Section 2 details the theoretical foundations of the LV competition models. Section 3 introduces the theoretical basis of reinforcement learning and Softmax agents, elaborating on the ABC-PMC algorithm and its integration with Softmax agents.

Section 4 validates the algorithm's effectiveness through a numerical example. Section 5 presents empirical research findings. Section 6 concludes this study.

2. Lotka-Volterra competition model

Equation (2.1) is the LV competition model, commonly used to simulate resource competition between two species. It has been widely applied to competitive business scenarios.

$$\begin{cases} \frac{dX}{dt} = r_1 X\left(1 - \frac{X}{K_1} - \frac{\alpha_0 Y}{K_1}\right), \\ \frac{dY}{dt} = r_2 Y\left(1 - \frac{Y}{K_2} - \frac{\beta_0 X}{K_2}\right). \end{cases} \tag{2.1}$$

In the aforementioned competition model, variables X and Y represent the population sizes of the two competing species, while $\frac{dX}{dt}$ and $\frac{dY}{dt}$ denote their instantaneous rates of change at time t. The parameters in the model carry specific ecological significance: r_1 and r_2 denote the intrinsic growth rates of the two species, respectively, K_1 and K_2 denote the environmental carrying capacities for each species in the absence of competition, α_0 characterizes the competitive suppression intensity of species Y on species X, and β_0 characterizes the competitive suppression intensity of species X on species Y.

The terms in the model can be interpreted as follows: $\frac{X}{K_1}$ represents the proportion of resource environment occupied by species X within its own habitat, while $\frac{Y}{K_2}$ represents the proportion occupied by species Y within its own habitat. Competitive effects are embodied by $\frac{\alpha_0 Y}{K_1}$, reflecting species Y's equivalent resource occupation of species X through competition. Similarly, $\frac{\beta_0 X}{K_2}$ reflects species X's equivalent resource occupation of species Y through competition.

The classical form of the LV model is a continuous-time differential equation. However, in practical computations, it is often necessary to convert it into a discrete-time form to facilitate numerical simulation and Bayesian parameter estimation. To this end, this paper performs discretization processing for three common variants of the LV model: the discretized LV (DLV) model, the randomized LV (RLV) model, and the fractional LV (FLV) model.

2.1. Discretized Lotka-Volterra model (DLV)

Using the forward Euler method, the continuous time t is discretized into a sequence of time steps with a time increment of Δt. Derivatives are approximated by the following forward differences:

$$\frac{dX}{dt} \approx \frac{X_{t+\Delta t} - X_t}{\Delta t}, \quad \frac{dY}{dt} \approx \frac{Y_{t+\Delta t} - Y_t}{\Delta t}. \tag{2.2}$$

Here, X_t and Y_t denote the population sizes at time t, while $X_{t+\Delta t}$ and $Y_{t+\Delta t}$ denote the population sizes at the subsequent time $t + \Delta t$.

Substituting the approximate expression into the continuous model (2.1) yields

$$\frac{X_{t+\Delta t} - X_t}{\Delta t} = X_t(a_1 - b_1 X_t - c_1 Y_t), \tag{2.3}$$

$$\frac{Y_{t+\Delta t} - Y_t}{\Delta t} = Y_t(a_2 - b_2 Y_t - c_2 X_t), \tag{2.4}$$

where $a_1 = r_1, a_2 = r_2, b_1 = r_1/K_1, b_2 = r_2/K_2$, and $c_1 = r_1\alpha_0/K_1$, $c_2 = r_2\beta_0/K_2$.

Both sides are multiplied by Δt, and terms are rearranged:

$$X_{t+\Delta t} - X_t = \Delta t \cdot X_t(a_1 - b_1 X_t - c_1 Y_t), \tag{2.5}$$

$$Y_{t+\Delta t} - Y_t = \Delta t \cdot Y_t(a_2 - b_2 Y_t - c_2 X_t). \tag{2.6}$$

Thus, the discrete-time equation is obtained:

$$\begin{cases} X_{t+\Delta t} = X_t + \Delta t \cdot (a_1 - b_1 X_t - c_1 Y_t) X_t, \\ Y_{t+\Delta t} = Y_t + \Delta t \cdot (a_2 - b_2 Y_t - c_2 X_t) Y_t. \end{cases} \tag{2.7}$$

2.2. Randomized Lotka-Volterra model (RLV)

In reality, the influence of random factors is difficult to avoid. Therefore, random disturbance terms are often introduced to extend the model into the RLV model. Assuming that the evolution of X and Y is affected by normally distributed noise, the RLV can be expressed as

$$\begin{cases} X[i] \sim \mathcal{N}\left(X[i-1] + \Delta t \cdot (a_1 - b_1 X[i-1] - c_1 Y[i-1]) X[i-1],\ \gamma_1^2 \right), \\ Y[i] \sim \mathcal{N}\left(Y[i-1] + \Delta t \cdot (a_2 - b_2 Y[i-1] - c_2 X[i-1]) Y[i-1],\ \gamma_2^2 \right). \end{cases} \tag{2.8}$$

Here, $X[i]$ and $Y[i]$ represent the variable values at the ith time step, respectively, both following a normal distribution; the variances γ_1^2 and γ_2^2 measure the magnitude of random factors influencing the dynamics of the two species, respectively.

2.3. Fractional-order Lotka-Volterra model (FLV)

To characterize memory effects and long-range dependencies in species interactions, this section considers an FLV predator-prey model. The population sizes of the two species are denoted as $x(t)$ and $y(t)$, respectively, and their dynamics are governed by the following Caputo-type fractional differential equations:

$$\begin{cases} D_t^\alpha x(t) = x(t)(a_3 - b_3 x(t)) - c_3\, x(t) y(t), \\ D_t^\beta y(t) = -d_3\, y(t) + e_3\, x(t) y(t). \end{cases} \tag{2.9}$$

Here, $0 < \alpha, \beta < 1$ denotes the fractional order, $a_3, b_3, c_3, d_3, e_3 > 0$ represent growth and interaction parameters, and both D_t^α and D_t^β are Caputo-type fractional derivative operators.

For $0 < \alpha < 1$, the Caputo derivative is defined as

$$D_t^\alpha x(t) = \frac{1}{\Gamma(1-\alpha)} \int_0^t (t-\tau)^{-\alpha} x'(\tau)\, d\tau. \tag{2.10}$$

The fractional integral operator I^α is defined as

$$I^\alpha f(t) = \frac{1}{\Gamma(\alpha)} \int_0^t (t-\tau)^{\alpha-1} f(\tau)\, d\tau. \tag{2.11}$$

Applying fractional integrals I^α and I^β to both sides of model (2.9) and utilizing the property $I^\alpha D_t^\alpha x(t) = x(t) - x(0)$ yields the equivalent Volterra integral equations:

$$x(t) = x(0) + \frac{1}{\Gamma(\alpha)} \int_0^t (t-\tau)^{\alpha-1} \left[x(\tau)(a_3 - b_3 x(\tau)) - c_3 x(\tau) y(\tau)\right] d\tau, \tag{2.12}$$

$$y(t) = y(0) + \frac{1}{\Gamma(\beta)} \int_0^t (t-\tau)^{\beta-1} \left[-d_3 y(\tau) + e_3 x(\tau) y(\tau)\right] d\tau. \tag{2.13}$$

For the integral equations (2.12) and (2.13), we discretize the time interval $[0, T]$ into N equal steps with step size $h = T/N$ and denote $t_k = kh$ for $k = 0, 1, \ldots, N$. Approximating the integrands $f_x(\tau) = x(\tau)(a_3 - b_3 x(\tau)) - c_3 x(\tau) y(\tau)$ and $f_y(\tau) = -d_3 y(\tau) + e_3 x(\tau) y(\tau)$ by their left endpoint values $f_x(t_j)$ and $f_y(t_j)$ on each subinterval $[t_j, t_{j+1}]$ and performing exact integration of the kernel functions $(t_k - \tau)^{\alpha-1}$ and $(t_k - \tau)^{\beta-1}$, we obtain the classical L1-type time-discrete scheme:

$$x_k = x_0 + \frac{h^\alpha}{\Gamma(\alpha+1)} \sum_{j=0}^{k-1} \left[(k-j)^\alpha - (k-j-1)^\alpha\right]\left[x_j(a_3 - b_3 x_j) - c_3 x_j y_j\right], \tag{2.14}$$

and similarly for y_k. To simplify notation, introduce fractional-order discrete convolution coefficients:

$$b_j^{(\alpha)} = (j+1)^\alpha - j^\alpha, \qquad j = 0, 1, 2, \cdots. \tag{2.15}$$

Then the scheme can be rewritten compactly as

$$\begin{cases} x_k = x_0 + \dfrac{h^\alpha}{\Gamma(\alpha+1)} \displaystyle\sum_{j=0}^{k-1} b_{k-j-1}^{(\alpha)} \left[x_j(a_3 - b_3 x_j) - c_3 x_j y_j\right], \\ y_k = y_0 + \dfrac{h^\beta}{\Gamma(\beta+1)} \displaystyle\sum_{j=0}^{k-1} b_{k-j-1}^{(\beta)} \left[-d_3 y_j + e_3 x_j y_j\right]. \end{cases} \tag{2.16}$$

Theorem 1 (**Existence and uniqueness**). *Given $T > 0$ with a step size $h = T/N$, the discrete scheme* (2.16) *uniquely yields a numerical solution pair (x_k, y_k) for each time layer $k = 1, \ldots, N$.*

Proof. For any $k \geq 1$, the right-hand side of (2.16) depends only on the numerical solutions from the previous time steps $j = 0, \ldots, k-1$. Therefore, it can be explicitly recursed layer by layer over time, uniquely determining the entire sequence $\{(x_k, y_k)\}_{k=0}^N$ from the initial conditions (x_0, y_0).

Theorem 2 (**Convergence**). *Let Eq* (2.16) *satisfy $x, y \in C^2[0, T]$, and the numerical solution (x_k, y_k) is given by Eq* (2.16)*, then there exists a constant $C > 0$ independent of h and k such that*

$$\max_{0 \leq k \leq N} \left(|x(t_k) - x_k| + |y(t_k) - y_k|\right) \leq C(h^{2-\alpha} + h^{2-\beta}).$$

Specifically, the convergence order for the x component is $O(h^{2-\alpha})$, and for the y component, it is $O(h^{2-\beta})$.

Proof. Equation (2.16) represents the classical L1 approximation of the Caputo derivative. By utilizing the local truncation error estimate in L1 format and combining it with the discrete Gronwall inequality with convolution kernels, we can obtain the following results respectively,

$$|x(t_k) - x_k| \leq Ch^{2-\alpha}, \qquad |y(t_k) - y_k| \leq Ch^{2-\beta}.$$

This holds for all $k = 0, 1, \ldots, N$. Adding the two equations yields the conclusion.

3. Theoretical foundations and algorithmic framework

3.1. Reinforcement learning

Reinforcement learning is a vital branch of machine learning that focuses on how an agent takes a series of actions within an environment to maximize cumulative rewards. Unlike supervised learning, reinforcement learning does not require labeled datasets; instead, it learns through trial and error.

The core elements of reinforcement learning include:

1) **Agent**: The decision-making entity that learns.
2) **Environment**: The external world in which the agent operates. The agent interacts with it and obtains observations from the environment.
3) **State**: A description of the current situation in the environment.
4) **Action**: Operations that the agent can perform.
5) **Reward**: A scalar signal fed back to the agent by the environment after executing an action, evaluating the quality of the action.
6) **Policy**: The agent's behavioral guideline, defining how it selects actions in a given state. Policies can be deterministic or stochastic.

The reinforcement learning process is a cycle: The agent observes the current state of the environment, selects an action based on its policy, and upon receiving the action, the environment transitions to a new state and provides an immediate reward. The agent uses this feedback reward and the new state to update its policy or value function, enabling better decision-making in the future. The process is repeated until the termination condition is satisfied. The agent's goal is not to maximize immediate rewards, but to maximize the cumulative return from the current moment into the future.

3.2. Softmax agent

3.2.1. Introduction to the Softmax

The Softmax function converts any real-valued vector into a probability distribution. Its mathematical formula is

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}. \tag{3.1}$$

This formula first computes the exponential e^{z_i} for each vector element z_i, ensuring all values are positive, then sums all exponents to obtain the denominator $\sum_{j=1}^{K} e^{z_j}$. Finally, each exponent is divided by this sum for normalization. This process transforms an unnormalized score vector $\mathbf{z} = [z_1, z_2, \ldots, z_K]$ into a probability distribution $\hat{\mathbf{y}} = [\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_K]$ with sum 1. This endows the

output values with clear probabilistic meaning while preserving the original numerical ordering of the scores.

When combined with a greedy algorithm, Softmax forms the most direct decision strategy for classification tasks. The specific workflow is as follows:

First, obtain the probability distribution $\hat{\mathbf{y}}$ using the Softmax formula

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{3.2}$$

to obtain the probability distribution $\hat{\mathbf{y}}$.

Then the greedy algorithm executes the decision rule

$$\text{prediction} = \arg\max(\hat{y}_i). \tag{3.3}$$

That is, it directly selects the category with the highest probability as the final output. For example, if the Softmax output is $[0.15, 0.70, 0.15]$, the greedy strategy unhesitatingly chooses the second category. This combination implements a complete pipeline from model computation to final decision: Softmax provides a quantified uncertainty metric, while the greedy algorithm implements a simple and efficient optimal local selection.

3.2.2. Application as an agent policy

In the decision-making process of agents, the Softmax function is commonly employed to implement a probability-based exploration strategy, namely reinforcement learning. Its fundamental principle is as follows: Each agent calculates the selection probability for every action (i.e., candidate model) based on its estimated value in the current state, then samples and executes actions according to these probabilities.

The selection probability is obtained by inputting the value score $Q(k)$ of each model k into the Softmax function:

$$P(k) = \frac{e^{Q(k)/\tau}}{\sum_{j=1}^{K} e^{Q(j)/\tau}}. \tag{3.4}$$

Here, K denotes the total number of candidate models. τ is the temperature parameter that adjusts the exploration level. When $\tau \to 0$, the policy converges to a purely greedy strategy, consistently selecting the action with the highest value. When $\tau \to \infty$, the policy tends toward uniform random selection, thereby encouraging thorough exploration.

However, in the algorithm of RL(Softmax)-ABC-PMC considered in this paper, it is important to note that after each selection is completed, the value score $Q(k)$ for each action must be updated based on the reward obtained from that selection. This ensures that decisions in subsequent rounds are made using the updated value. The update process is as follows:

$$Q_{t+1}(k) = Q_t(k) + \eta \cdot (r - Q_t(k)). \tag{3.5}$$

r denotes the instantaneous reward observed at the current time step. If the model is selected in this round, $r = 1$; otherwise, $r = 0$. η represents the agent's learning rate.

Remark 1: Softmax is selected over upper confidence bound (UCB) or Thompson sampling due to its minimal computational overhead (single scalar per model), natural interpretability, and inherent adaptability to non-stationary rewards via exponential forgetting. Although it requires tuning of τ and η, its simplicity and transparency justify its use in our simulation-dominated model selection context. We will do a systematic comparison with other MAB strategies and neural SBI methods in the future work.

3.3. ABC-PMC algorithm

3.3.1. Approximate Bayesian computation

ABC is a computational method for solving Bayesian inference problems where the likelihood function is intractable. Its core idea is to bypass complex likelihood calculations by simulating data for comparison. For a given model, the standard process involves deriving a dataset from specified parameters. In contrast, the ABC algorithm uses datasets simulated from sampled parameters to infer the validity of those parameters. Specifically, it simulates datasets from the prior distribution of parameters, compares the degree of discrepancy between these simulated datasets and observed data, and adopts the distribution of parameters that pass this screening as the true posterior distribution.

The standard Bayesian posterior distribution is

$$\pi(\boldsymbol{\theta} \mid D) = \frac{P(D \mid \boldsymbol{\theta})P(\boldsymbol{\theta})}{P(D)} \propto P(D \mid \boldsymbol{\theta})P(\boldsymbol{\theta}). \tag{3.6}$$

The ABC rejection algorithm approximates this true posterior distribution $\pi(\boldsymbol{\theta} \mid D)$ using N sampled distributions. The distribution of accepted samples approximates $P(\boldsymbol{\theta} \mid \rho(S(D^*), S(D)) \leq \varepsilon) \approx \pi(\boldsymbol{\theta} \mid D)$. Higher desired precision requires a smaller ε, but this reduces the number of θ^* samples satisfying the condition, necessitating multiple draws. This leads to extremely low algorithmic efficiency.

3.3.2. PMC algorithm

PMC refers to population Monte Carlo, also known as particle filtering. It is an iterative sampling method whose core principle involves progressively guiding a set of random samples referred to as particles from an initial distribution toward a target distribution through iterative selection and evolution:

$$P_t(\boldsymbol{\theta}) \to P_{t+1}(\boldsymbol{\theta}) \to \cdots \to P_T(\boldsymbol{\theta}) \approx P(\boldsymbol{\theta} \mid D). \tag{3.7}$$

Each iteration employs a progressively smaller tolerance $\varepsilon_t > \varepsilon_{t+1}$.

3.3.3. ABC-PMC algorithm

To enhance computational efficiency, we integrate ABC with PMC. ABC-PMC abandons the fixed prior distribution and tolerance ε, instead adopting an adaptive strategy. It selects a perturbation kernel and progressively approximates the target posterior distribution through a sequence of intermediate distributions. Subsequent iterations no longer sample from a broad but inefficient prior, instead focusing exploration on the parameter space regions contributing most significantly to the posterior. The tolerance automatically and progressively decreases, ultimately yielding a high-quality, precise approximate posterior.

The specific steps are as follows:

Given an observed dataset D, a model M, tolerance thresholds $\varepsilon = \{\varepsilon_0, \varepsilon_1, \cdots, \varepsilon_T\}$, a distance function $d(\cdot)$, a perturbation kernel $q(\cdot)$, a sample size N, and an iteration count T:

1) From the prior distribution $\pi(\theta)$, generate samples $\theta^{(0)} = \{\theta_1^{(0)}, \cdots, \theta_N^{(0)}\}$ satisfying $d(Y, Y^*) \leq \varepsilon_0$ from the prior distribution $\pi(\theta)$. Set the initial weight set $\{w_i^{(0)}\}$ such that $\theta_i^{(0)} \sim \pi(\theta)$, $w_i^{(0)} = 1/N$, $i = 1, \cdots, N$, $\sigma_0^2 = 2Var\{\theta_i^{(0)}\}$ and iteration counter $t = 0$.
2) Set particle counter $i = 1$.
3) Sample one particle θ^* from the previous generation particle pool $\{\theta_i^{(t)}\}$ using the weight set $\{w_i^{(t)}\}$, then perturb to obtain $\theta^{**} \sim q(\theta^{**} \mid \theta^*, \sigma_t^2)$.
4) If $\pi(\theta^{**}) = 0$, return to Step 3; otherwise, generate the simulated dataset D^* from M given θ^{**}.
5) If $d(D, D^*) > \varepsilon_t$, return to Step 3; otherwise, retain θ^{**} as the ith particle in the $(t+1)$st generation pool and proceed to Step 6.
6) Set $\theta_i^{(t+1)} = \theta^{**}$ and assign weights as follows:

$$w_i^{(t+1)} = \frac{\pi(\theta_i^{(t+1)})}{\sum_{j=1}^{N} w_j^{(t)} q(\theta_j^{(t)} \mid \theta_i^{(t+1)}, \sigma_t)}.$$

If $i < N$, set $i = i + 1$ and return to Step 3. Otherwise, the $(t+1)$th generation particle pool sampling is complete. Set $\sigma_{t+1}^2 = 2Var\{\theta_i^{(t)}\}$ and proceed to Step 7.
7) Regularize weights. If $t < T$, set $t = t + 1$ and return to Step 2; if $t = T$, the algorithm completes and yields the Tth generation particle pool $\{\theta_i^{(T)}\}$.

Lemma 1 (Convergence of ABC-PMC [10, 11])**.** *Assume that the following conditions hold:*

- *The likelihood $p(y \mid \theta, \mathcal{M}^*)$ is continuous in θ and bounded on Θ, and the prior $\pi(\theta \mid \mathcal{M}^*)$ is proper and positive on Θ.*
- *The summary statistics $\eta(\cdot)$ are sufficient for the true model $\mathcal{M}^*$, and the mapping $\theta \mapsto \mathbb{E}[\eta(Y) \mid \theta, \mathcal{M}^*]$ is injective.*
- *The tolerance sequence satisfies $\varepsilon_t \to 0$ and $\sum_{t=1}^{\infty} \varepsilon_t^2 < \infty$.*

Then for the standard ABC-PMC algorithm, the weighted empirical measure

$$\mu_t^{(N)}(\theta) = \sum_{i=1}^{N} \tilde{w}_i^{(t)} \delta_{\theta_i^{(t)}}(\theta)$$

satisfies, for any bounded measurable function $f : \Theta \to \mathbb{R}$,

$$\lim_{t\to\infty} \lim_{N\to\infty} \mu_t^{(N)}(f) = \mathbb{E}_\pi[f(\theta) \mid y, \mathcal{M}^*] \quad \text{in } L^2.$$

Moreover, there exists a constant $C_f < \infty$ such that

$$\mathbb{E}\Big[(\mu_t^{(N)}(f) - \pi(f))^2\Big] \leq C_f\Big(\frac{1}{N} + \varepsilon_t^2\Big).$$

Proof. The ABC-PMC algorithm is an adaptive sequential importance sampling method with decreasing tolerance ε_t. Under the stated assumptions, it satisfies the conditions of the adaptive SMC framework established in [10]. The importance weights are properly normalized, and the perturbation kernel ensures that the particle system does not degenerate. The L^2 convergence follows from the standard particle law of large numbers; the term $1/N$ arises from Monte Carlo variance, while ε_t^2 accounts for the ABC approximation error. For a detailed proof, see Frazier, Martin, Robert and Rousseau [11].

3.4. Combined algorithm of RL(Softmax)-ABC-PMC

In this approach, ABC-PMC provides the foundational algorithmic structure, employing a multi-round incremental optimization strategy with progressively decreasing distance thresholds. Each round accepts only particles where the distance between simulated and observed data falls below the threshold. As iterations advance, the threshold tightens progressively, driving the parameter distribution toward the true posterior. The reinforcement learning strategy is embedded within the model selection phase. It calculates selection probabilities based on the cumulative rewards of each model, mapping numerical preferences to probabilistic values. This ensures high-performing models receive greater sampling opportunities while maintaining moderate exploration of suboptimal models. RL permeates the entire sampling process, continuously refining the model selection strategy through online learning. This enables the algorithm to extract valuable insights from historical experience and dynamically adjust its behavioral patterns.

The specific workflow is as follows:

1) At the start of each round, obtain the current threshold ϵ.
2) Use the Softmax agent to select a model, then sample θ^* from the particle pool $\{\theta_i^{(t)}\}$ accepted by the selected model in the previous round, perturbing it to generate a new particle θ^{**}.
3) Compute the distance function and compare it with the threshold. If the particle satisfies the condition, incorporate it into the particle pool, recalculate its weight, and normalize it; otherwise, update the agent based on the reward, reselect a model, and re-enter the iteration until a selected particle meets the requirements.
4) After completing all rounds, output the particle set and weight for each model.

The core idea of this method is to dynamically allocate computational resources using reinforcement learning agents. The algorithm no longer blindly assigns equal opportunities to each model. Instead, agents intelligently and probabilistically allocate more computational resources to models that currently appear more promising based on historical performance. This significantly improves algorithmic efficiency, enabling faster convergence to optimal models and acquisition of the posterior distribution of parameters.

The RL(Softmax)-ABC-PMC algorithm is provided as the following:

Algorithm 1: RL(Softmax)-ABC-PMC Algorithm

Input: Observation data D, model set $\{\mathcal{M}_k\}_{k=1}^{K}$, distance $d(\cdot,\cdot)$, threshold sequence $\{\varepsilon_t\}_{t=1}^{T}$, perturbation kernel $q(\cdot|\cdot,\sigma^2)$, particle count N, temperature τ, learning rate η

Initialization: $t \leftarrow 0$,

for $i \leftarrow 1$ **to** N **do**

$w_i^{(0)} \leftarrow 1/N$, $Q_i^{(0)} \leftarrow \mathbf{0}$, // all models start with equal Q-value (zero)

$P_k^{(i)} \leftarrow \frac{\exp(Q_i^{(0)}[k]/\tau)}{\sum_j \exp(Q_i^{(0)}[j]/\tau)}$,

// Since $Q_i^{(0)}[k] = 0$ for all k, $P_k^{(i)} = 1/K$; thus the first iteration uses a uniform policy.

$\mathcal{M}_{0,i}^{*} \sim \mathrm{Cat}(\{P_k^{(i)}\})$, // Cat: select model according to probability vector P

$\theta_i^{(0)} \sim \pi_{\mathcal{M}_{0,i}^{*}}(\theta)$,

$\mathcal{P}_w^{(0)} \leftarrow \{(\theta_i^{(0)}, w_i^{(0)}) : \mathcal{M}_{t,i}^{*} = \mathcal{M}_k\}, k = 1, \ldots, K$;

end

$\sigma_0^2 \leftarrow 2 \cdot \mathrm{Var}(\{\theta_i^{(0)}\})$;

for $t \leftarrow 1$ **to** T **do**

$\varepsilon \leftarrow \varepsilon_t$,

for $i \leftarrow 1$ **to** N **do**

1. Model Selection:

$P_k^{(i)} \leftarrow \frac{\exp(Q_i^{(t-1)}[k]/\tau)}{\sum_j \exp(Q_i^{(t-1)}[j]/\tau)}$,

$\mathcal{M}_{t,i}^{*} \sim \mathrm{Cat}(\{P_k^{(i)}\})$; // Cat:select model according to probability vector P

2. Parameter Sampling:

$\theta' \sim \mathrm{WeightedSample}(\mathcal{P}_w^{(t-1)})$, // Weighted sample from previous particle pool

$\theta_i^{(t)} \sim q(\theta \mid \theta', \sigma_t^2)$;

3. Evaluation: Generate simulated data $D^{*} \sim (\mathcal{M}_{t,i}^{*}, \theta_i^{(t)})$, $d_{\mathrm{curr}} \leftarrow d(D, D^{*})$;

if $d_{curr} \le \varepsilon$ **then**

keep $\theta_i^{(t)}$ and $\mathcal{M}_{t,i}^{*}$,

4. Weight Calculation:

$$w_i^{(t)} \leftarrow \frac{\pi_{\mathcal{M}_{t,i}^{*}}(\theta_i^{(t)})}{\sum_{j \in \mathcal{P}_{\mathcal{M}_{t,i}^{*}}^{(t-1)}} w_j^{(t-1)}\, q(\theta_i^{(t)} \mid \theta_j^{(t-1)}, \sigma_{t-1}^2)},$$

$i \leftarrow i + 1$

end

else

$Q_i^{(t)}[k_i^{(t)}] \leftarrow Q_i^{(t-1)}[k_i^{(t)}] + \eta(1 - Q_i^{(t-1)}[k_i^{(t)}])$; // update Q

end

end

5. Normalize weights: $\tilde{w}_i^{(t)} \leftarrow w_i^{(t)} / \sum_{j=1}^{N} w_j^{(t)}, i = 1, \ldots, N$;

6. Construct model particle pool: $\mathcal{P}_w^{(t)} \leftarrow \{(\theta_i^{(t)}, \tilde{w}_i^{(t)}) : \mathcal{M}_{t,i}^{*} = \mathcal{M}_k\}, k = 1, \ldots, K$;

7. Update parameters: $\sigma_{t+1}^2 \leftarrow 2 \cdot \mathrm{Var}(\{\theta_i^{(t)}\})$, $t \leftarrow t + 1$.

end

Remark 2: Let $\mathcal{P}_w^{(t-1)}$ denote the weighted particle pool from iteration $t-1$, i.e., $(\theta_i^{(t-1)}, \tilde{w}i^{(t-1)})$. For a given model $\mathcal{M}$, $\mathcal{PM}^{(t-1)}$ is the subset of $\mathcal{P}_w^{(t-1)}$ consisting of particles generated under that model. This notation is used in the weight calculation and model-specific sampling steps of Algorithm 1.

Remark 3: The Q-value update is performed online immediately after each rejected proposal (per-particle update), enabling rapid adaptation to the non-stationary tolerance sequence ε_t. Although this introduces some variance, the learning rate $\eta = 0.1$ provides sufficient smoothing, and our experiments confirm stable convergence; batch updates are left as an alternative for future exploration.

Remark 4: In ABC-PMC, the decreasing acceptance threshold ε_t makes the reward landscape non-stationary. The Q-value update $Q_{t+1}(k) = (1-\eta)Q_t(k) + \eta r$ acts as an exponentially weighted moving average with memory horizon $1/\eta$. This naturally discards outdated information, enables rapid adaptation, and requires no explicit change detection mechanism, aligning with standard MAB strategies for non-stationary bandits.

Remark 5: The underlying ABC-PMC algorithm converges to the true posterior under standard regularity conditions. Our RL(Softmax)-ABC-PMC preserves the acceptance-rejection and importance-weighting mechanisms, only adaptively selecting the model at each iteration. Since the Softmax agent selects the true model with probability approaching one as $\varepsilon_t \to 0$, the RL(Softmax)-ABC-PMC algorithm inherits the convergence of standard ABC-PMC (Lemma 1) in the limit. Hence, the asymptotic validity of the posterior approximation remains intact. A rigorous analysis of the effect of reward history on finite sample efficiency and convergence rate is deferred to future work.

4. Numerical simulations

To validate the effectiveness of the ABC-PMC algorithm based on Softmax agents in model selection, this paper conducts numerical simulation experiments using both S-shaped growth curves and LV models for model selection. Multi-faceted verification is performed using one-dimensional and two-dimensional data. To ensure statistical robustness, all experiments were repeated 100 times with different random seeds. The results are reported as mean ± standard deviation.

4.1. Model selection for S-shaped growth curves

Three classic S-shaped growth curve models are selected as candidates: the modified exponential model, the logistic model, and the Gompertz model. While these models differ mathematically, they can produce similar curves within a given interval through appropriate parameter adjustments.

The modified exponential model is one of the simplest S-shaped growth models, expressed mathematically as

$$N(t) = K - (K - N_0) \cdot e^{-rt}, \tag{4.1}$$

where K represents environmental capacity, r denotes the growth rate, and N_0 is the initial population size. This model employs an exponential decay term $(K - N_0) \cdot e^{-rt}$ to gradually approach the carrying capacity K from the initial population N_0. Its characteristic features include a rapid growth rate in the early stages, which gradually slows as the population nears the carrying capacity, though the deceleration process is relatively uniform.

The logistic model is the most classic S-shaped growth model, expressed mathematically as:

$$N(t) = \frac{K}{1 + \frac{K-N_0}{N_0} \cdot e^{-rt}}. \tag{4.2}$$

This model achieves nonlinear saturation through a fractional form. Its growth rate is slow both in the early and late stages, reaching a maximum in the intermediate phase and exhibiting a typical S-shaped characteristic.

The Gompertz model is an asymmetric S-curve model, expressed mathematically as

$$N(t) = K \cdot e^{-e^{-r(t-\tau)}}, \tag{4.3}$$

where τ is the time parameter. This model achieves nonlinear saturation through a double exponential form: The inner exponential term controls the growth rate, while the outer exponential term implements saturation. Its growth rate is rapid in the early stage and slows sharply as it approaches environmental capacity. The inflection point is controlled by the parameter τ.

To simulate scenarios where models struggle to differentiate in practical applications, this paper selects the modified exponential model curve as the reference curve. Parameters are sought to make the logistic model and Gompertz model curves resemble the reference curve as closely as possible. We set the modified exponential model parameters to $K = 100.0, r = 0.22, N_0 = 8.0$. Curve evaluation was conducted using 60 time points uniformly distributed within the interval [0, 10]. To obtain an S-shaped curve as similar as possible, the model parameters were obtained through continuous optimization. The specific process is as follows: The first stage employs a coarse-grained network search, followed by the second stage using either the Limited-memory Broyden-Fletcher-Goldfarb-Shanno with Box constraints (L-BFGS-B) algorithm or random neighborhood search. Specific parameters are listed in Table 1.

Table 1. The parameter values and prior distribution ranges of the three similar S-shaped growth curves constructed.

Model Name	Parameter Name	Parameter Value	Prior Distribution Range
Modified exponential model	K	100.0	[95.0, 105.0]
	r	0.22	[0.15, 0.30]
	N_0	8.00	[5.0, 10.0]
Logistic model	K	100.12	[95.0, 105.0]
	r	0.40	[0.15, 0.45]
	N_0	20.00	[5.0, 25.0]
Gompertz model	K	98.34	[95.0, 105.0]
	r	0.34	[0.15, 0.30]
	τ	2.00	[3.0, 7.0]

The three model curves are shown in Figure 1. It can be observed that all three models produce highly similar S-shaped growth curves within the time interval. Based on this, this paper generates sample data from the modified exponential model and adds Gaussian noise, making it difficult to intuitively identify the true model based solely on observed data. The Gaussian noise is defined as:

$$Y_i = N(t_i) + \omega_i, \quad \omega_i \sim \mathcal{N}(0, \sigma^2). \tag{4.4}$$

In the ABC-PMC algorithm based on Softmax agents, the threshold sequence is set as $\varepsilon = \{120.0, 117.0, 114.0, 111.0, 108.0, 105.0, 102.0, 99.0, 96.0, 93.0\}$ to progressively tighten acceptance criteria; the target particle count N per iteration is 150. The temperature parameter τ of the Softmax agent is set to 0.2, favoring the current optimal model; the learning rate α is 0.1; and the initial policy is a uniform distribution across the three models. A positive reward is given when the distance d is less than the threshold ε, guiding the agent to learn to select models that generate outputs closer to the observed data.

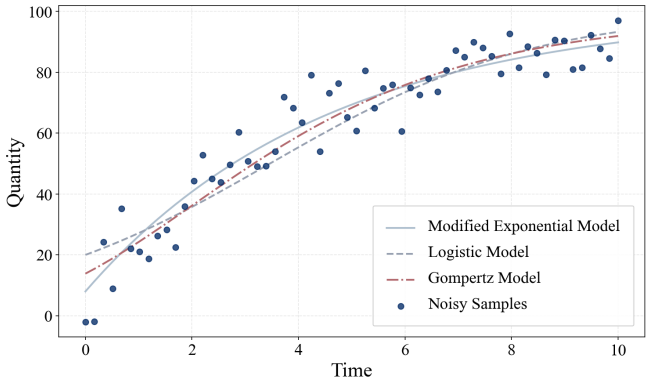


Figure 1. Comparison of curves constructed based on the three S-curve growth models.

To validate the performance of the proposed Softmax-based ABC-PMC algorithm for model selection against the traditional ABC-PMC algorithm, a comparative analysis was conducted. The iterative distribution plots of the particle sets before and after agent incorporation are shown in Figure 2, with each subplot labeled sequentially by iteration number and threshold value. The results demonstrate that the ABC-PMC algorithm with the agent converges faster than the traditional ABC-PMC algorithm. Figure 2(a) shows that the Softmax-based ABC-PMC algorithm rapidly converges to a single model, effectively improving recognition efficiency. In contrast, Figure 2(b) indicates that the pure ABC-PMC algorithm fails to quickly identify the true model.

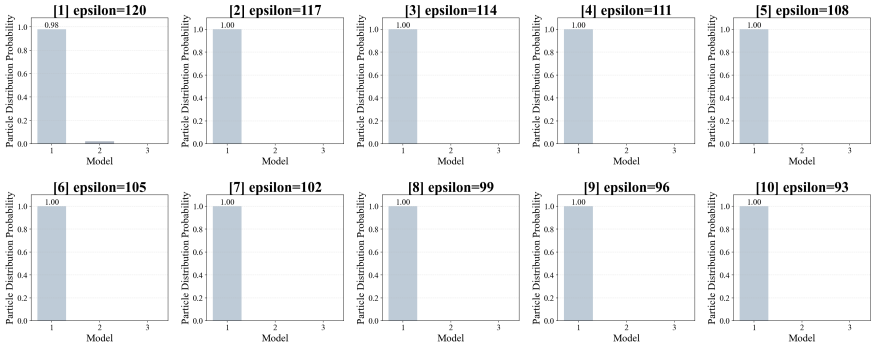
To provide a rigorous quantitative comparison, we computed the relative root mean square error (RRMSE) for all estimated parameters over 100 independent runs. Following the suggestion in [30], RRMSE is defined as:

$$\text{RRMSE} = \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{\theta}_i - \theta_{\text{true}})^2}}{|\theta_{\text{true}}|} \times 100\%.$$

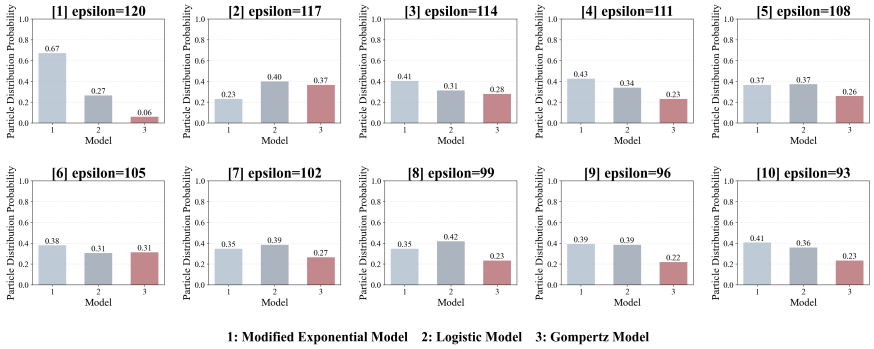
Table 2. RRMSE (%) comparison for S-curve model parameters over 100 independent runs.

Parameter	True Value	RL(Softmax)-ABC-PMC	Standard ABC-PMC
hline K	100.0	1.0 ± 0.9	0.8 ± 0.7
r	0.22	7.9 ± 6.4	6.1 ± 5.3
N_0	8.0	8.2 ± 6.0	7.2 ± 5.3

Both algorithms achieve low RRMSE values, indicating that both methods can accurately estimate parameters in this relatively simple model.



(a) RL(Softmax)-ABC-PMC algorithm: iterations 1–10 with thresholds $\varepsilon = 120.0, 117.0, \ldots, 93.0$.



(b) Standard ABC-PMC algorithm: same threshold sequence but without adaptive model selection.

Figure 2. Comparison of iterative particle set distributions for model selection using RL(Softmax)-ABC-PMC and standard ABC-PMC algorithms (S-shaped growth curve experiment).

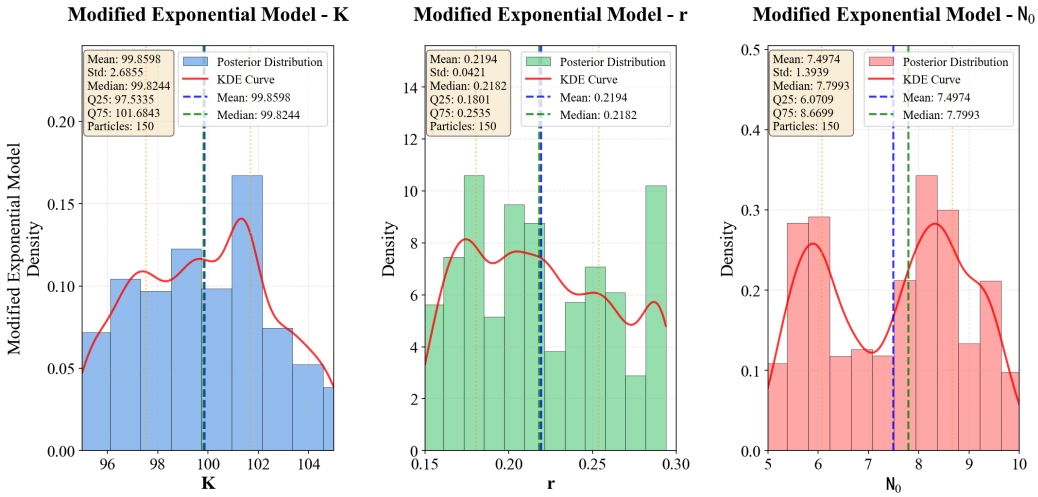


Figure 3. Posterior distribution of the modified exponential model parameters obtained by the RL(Softmax)-ABC-PMC algorithm.

Furthermore, the corresponding posterior distributions for model parameters obtained through the above model selection are shown in Figure 3. The posterior distributions yield the estimated parameters K, r, and N_0 for the modified exponential model, which are 99.86, 0.22, and 7.50, respectively. These values are close to the true model parameters $K = 100.0, r = 0.22, N_0 = 8.0$ and exhibit small standard deviations, indicating low uncertainty in parameter estimation. This demonstrates that the ABC-PMC algorithm based on Softmax agents can perform model selection rapidly and accurately.

Table 3. S-curve: computational cost and accuracy (mean ± std over 10 runs).

Metric	RL	ABC-PMC
Total time (s)	0.077 ± 0.002	0.132 ± 0.004
Total simulations	1008 ± 31	2758 ± 52
Accepted particles (correct model)	1000 ± 0	915 ± 18
Correct model selection rate (%)	99.7 ± 0.3	33.2 ± 2.1

The reinforced learning method selects the true model in 99.7% of simulations, compared to 33.2% for standard ABC-PMC, while being faster (0.077 s vs. 0.132 s).

4.2. Numerical simulation of the Lotka-Volterra competition model

To further validate the effectiveness of the RL(Softmax)-ABC-PMC algorithm in multidimensional data model selection, this section selects the classic LV competition models as the research subject, focusing on examining the algorithm's performance when processing two-dimensional time series data.

Although this model is typically described in continuous-time form, this section selects the DLV, RLV, and FLV models for numerical simulation to facilitate computer-based modeling and parameter estimation.

The following discrete form is derived from the preceding derivation:

Discretized Lotka-Volterra model (DLV):

$$\begin{cases} X_{t+\Delta t} = X_t + \Delta t \cdot (a_1 - b_1 X_t - c_1 Y_t) X_t, \\ Y_{t+\Delta t} = Y_t + \Delta t \cdot (a_2 - b_2 Y_t - c_2 X_t) Y_t. \end{cases} \tag{4.5}$$

X_t and Y_t denote the variable values at time t; Δt is the specified time step; and $X_{t+\Delta t}$, $Y_{t+\Delta t}$ denote the variable values at time $t + \Delta t$.

Randomized Lotka-Volterra model (RLV):

$$\begin{cases} X[i] \sim N\left(X[i-1] + \Delta t \cdot (a_1 - b_1 X[i-1] - c_1 Y[i-1]) X[i-1], \gamma_1^2\right), \\ Y[i] \sim N\left(Y[i-1] + \Delta t \cdot (a_2 - b_2 Y[i-1] - c_2 X[i-1]) Y[i-1], \gamma_2^2\right). \end{cases} \tag{4.6}$$

$X[i]$ and $Y[i]$ denote the variable values at time i, both following a normal distribution; the intrinsic randomness of the two-dimensional ecological process is further quantified by parameters γ_1 and γ_2.

Fractional-order Lotka-Volterra model (FLV):

$$\begin{cases} x_k = x_0 + \dfrac{h^\alpha}{\Gamma(\alpha+1)} \displaystyle\sum_{j=0}^{k-1} b_{k-j-1}^{(\alpha)} \left[x_j(a_3 - b_3 x_j) - c_3 x_j y_j \right], \\ y_k = y_0 + \dfrac{h^\beta}{\Gamma(\beta+1)} \displaystyle\sum_{j=0}^{k-1} b_{k-j-1}^{(\beta)} \left[-d_3 y_j + e_3 x_j y_j \right]. \end{cases} \tag{4.7}$$

To simulate the challenges of model identification in practical applications, this section aims to construct three candidate models that are highly similar in form but differ in mechanism. The DLV model is set as the reference model, with its parameter settings as shown in Table 4. For RLV, to ensure its trend aligns with the reference curve, only the random disturbance terms are newly set: $\gamma_1 = \gamma_2 = 0.05$. For FLV, the deterministic parameters $(a_3, b_3, c_3, d_3, e_3)$ are adjusted to match the growth trends of the reference model, with $a_3 = 0.547$, $b_3 = 0.00987$, $c_3 = 0.001$, $d_3 = 0.05$, $e_3 = 0.00541$, while the fractional orders are set as $\alpha = 0.912$ and $\beta = 0.600$ to capture memory effects in the population dynamics.

Table 4. Parameter settings and prior distribution ranges for three LV competition models.

Parameter Symbol	Parameter value	Prior distribution range
a_1, a_2, a_3	0.600, 0.500, 0.547	[0.1, 1.0]
b_1, b_2, b_3	0.008, 0.008, 0.00987	[0.001, 0.02]
c_1, c_2, c_3	0.006, 0.005, 0.001	[0.001, 0.01]
d_3, e_3	0.05, 0.00541	[0.05, 1.2]
X_0, Y_0	25.0, 20.0	[5.0, 30.0]
γ_1, γ_2 (RLV)	0.05, 0.05	[0.0, 0.1]
α, β (FLV)	0.912, 0.600	[0.6, 0.99]

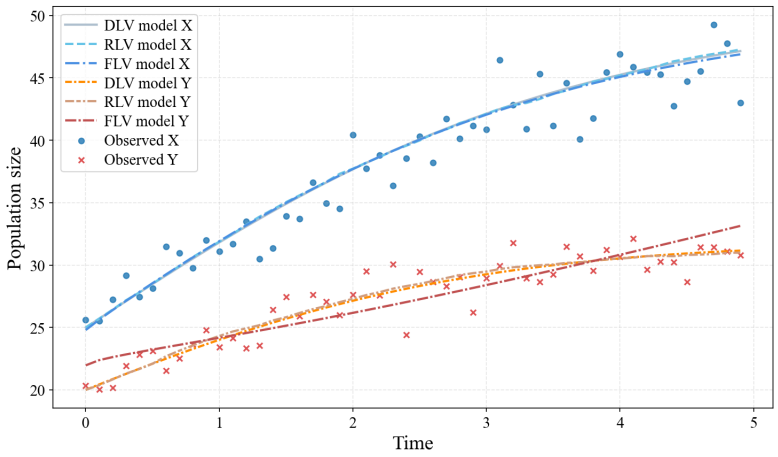


Figure 4. Comparison of DLV, RLV and FLV model curves under specified parameters.

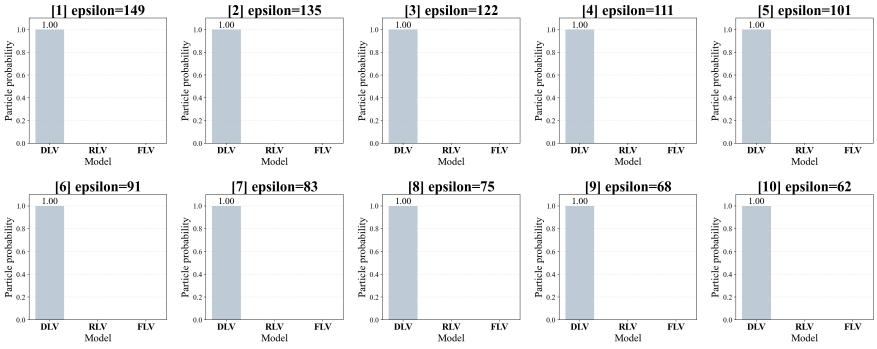
Figure 4 illustrates the trajectory comparison among the three models. It is evident that while maintaining the overall growth trend consistent with the discretized model, the stochastic model

exhibits distinct sawtooth-like fluctuations, and the fractional model demonstrates smooth trajectories that closely approximate the discretized model when the fractional orders approach unity. These subtle morphological differences serve as the key basis for model selection algorithms to distinguish between them.

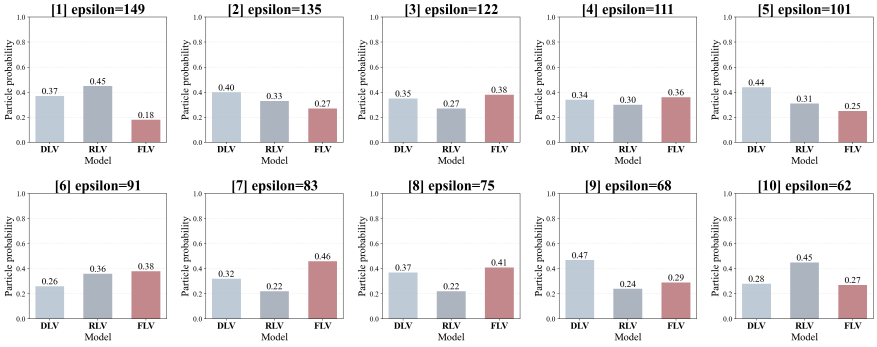
When generating observational data, the DLV model is first run using baseline parameters to obtain theoretical trajectories at 50 time points. Subsequently, observational noise proportional to the population size is superimposed on the theoretical values, with a noise level set at 0.05 to simulate measurement errors encountered in actual observations. The observational data generation formula is

$$X_{\text{obs}} = X_{\text{true}} + \omega, \quad \omega \sim N(0, (0.05 \cdot |X_{\text{true}}|)). \tag{4.8}$$

Based on the minimum distance quantiles calculated for each model in the preliminary experiment, this study sets the threshold sequence ϵ for the ABC-PMC algorithm as $\{148.5075, 134.7419, 122.2523, 110.9204, 100.6389, 91.3103, 82.8465, 75.1672, 68.1998, 61.8781\}$. As iterations progressed, the acceptance criterion was progressively tightened to approximate the posterior distribution, with each round targeting 100 particles. The temperature parameter τ of the Softmax agent was set to 0.2 to enhance utilization of dominant models; the learning rate α was set to 0.1; and the initial policy was uniformly distributed.



(a) RL(Softmax)-ABC-PMC algorithm: iterations 1–10 with thresholds $\varepsilon = 148.5, 134.7, \ldots, 61.9$.



(b) Standard ABC-PMC algorithm: same threshold sequence without adaptive model selection.

Figure 5. Comparison of iterative particle distributions in DLV, RLV and FLV model selection using RL(Softmax)-ABC-PMC and standard ABC-PMC algorithms.

To validate the performance of the proposed algorithm, Figure 5 illustrates the comparison between the ABC-PMC algorithm with Softmax agents and the agent-free approach in terms of the evolution of model selection probabilities during iterations.

The algorithm ultimately selected the DLV model as the optimal model with high efficiency. Despite the highly similar trajectories of the three LV models in Figure 4, the proposed RL(Softmax)-ABC-PMC algorithm accurately identified the true data-generating model even under observational noise. This result demonstrates the method's superiority in distinguishing subtle differences among competing models while exhibiting robust resilience to noise interference, thereby validating the effectiveness of the adaptive model selection strategy.

In contrast, the traditional ABC-PMC algorithm converges more slowly without agent guidance and fails to select the optimal model, making it less computationally efficient than the algorithm proposed in this paper.

To quantitatively evaluate the performance, we conducted 100 independent runs for each algorithm. Table 5 presents the RRMSE for key parameters of the DLV model.

Table 5. RRMSE (%) comparison for DLV model parameters over 100 independent runs.

Parameter	True Value	RL(Softmax)-ABC-PMC	Standard ABC-PMC
a_1	0.600	28.5 ± 19.8	25.7 ± 18.2
b_1	0.008	57.4 ± 39.3	56.1 ± 41.5
c_1	0.006	31.1 ± 21.2	31.2 ± 19.4
a_2	0.500	34.1 ± 23.9	34.8 ± 20.2
b_2	0.008	60.9 ± 39.0	66.9 ± 39.8
c_2	0.005	33.2 ± 25.3	31.2 ± 22.7
X_0	25.0	29.9 ± 22.0	32.5 ± 20.9
Y_0	20.0	24.3 ± 17.9	24.6 ± 17.5
Δt	0.1	29.9 ± 24.4	35.1 ± 23.3

For the challenging LV models, the RRMSE values are substantially higher than those in the S-curve experiments, reflecting the intrinsic difficulty of parameter inference in complex dynamical systems. Notably, while the two methods yield comparable RRMSE for most parameters, the RL(Softmax)-ABC-PMC algorithm demonstrates superior performance in model selection. This highlights that the primary benefit of our adaptive approach lies in efficient identification of the correct model structure, without sacrificing parameter estimation precision.

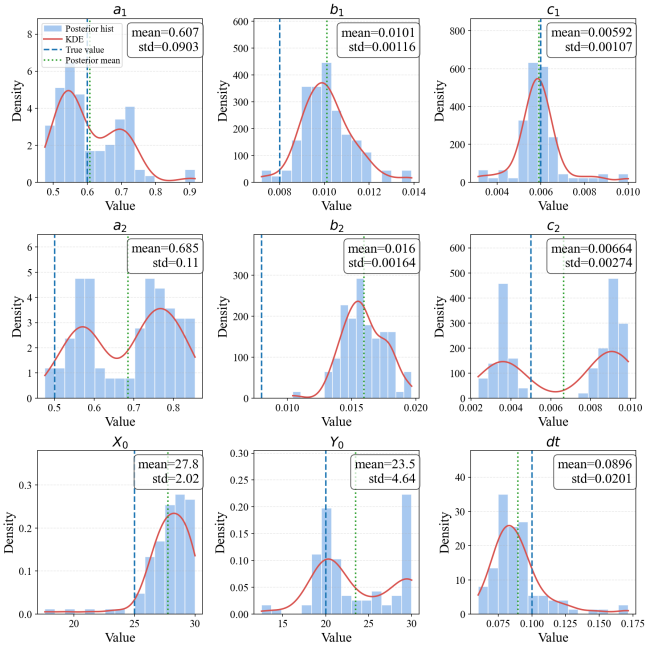


Figure 6. Posterior distributions of parameters for the DLV model obtained via the RL(Softmax)-ABC-PMC algorithm.

Statistical analysis of the posterior samples indicates the following estimates for key parameters (DLV): growth rates $\hat{a}_1 \approx 0.607, \hat{a}_2 \approx 0.685$; self-regulation coefficients $\hat{b}_1 \approx 0.0101, \hat{b}_2 \approx 0.0160$; competition coefficients $\hat{c}_1 \approx 0.00592, \hat{c}_2 \approx 0.00664$; initial states $\hat{X}_0 \approx 27.75, \hat{Y}_0 \approx 23.46$; and time step $\hat{dt} \approx 0.0896$. These estimates closely approximate the true parameters $a_1 = 0.6, a_2 = 0.5, b_1 = b_2 = 0.008, c_1 = 0.006, c_2 = 0.005, X_0 = 25.0, Y_0 = 20.0, dt = 0.1$, validating that the proposed algorithm can automatically identify the data-generating model and provide reliable parameter intervals.

Table 6. LV: computational cost and accuracy (mean ± std over 10 runs).

Metric	RL	ABC-PMC
Total time (s)	31.45 ± 1.02	22.78 ± 0.89
Total simulations	172386 ± 2100	172068 ± 1850
Accepted particles (correct model)	996 ± 3	523 ± 27
Correct model selection rate (%)	99.7 ± 0.2	50.0 ± 1.5

The reinforced learning method improves model selection accuracy from 50.0% to 99.7%, accepting nearly all particles from the true model (996 vs. 523), with only a modest increase in runtime.

5. Empirical study

This section utilizes actual quarterly sales data from China's automotive market to evaluate the selection among the DLV model, the RLV model and the FLV model. The data originates from CAAM (http://www.caam.org.cn) and encompasses sales figures for two vehicle categories: new

energy vehicles (NEVs) and traditional fuel vehicles (TFVs). The time span covers the period from the second quarter of 2015 to the fourth quarter of 2024, comprising a total of 39 observation points.

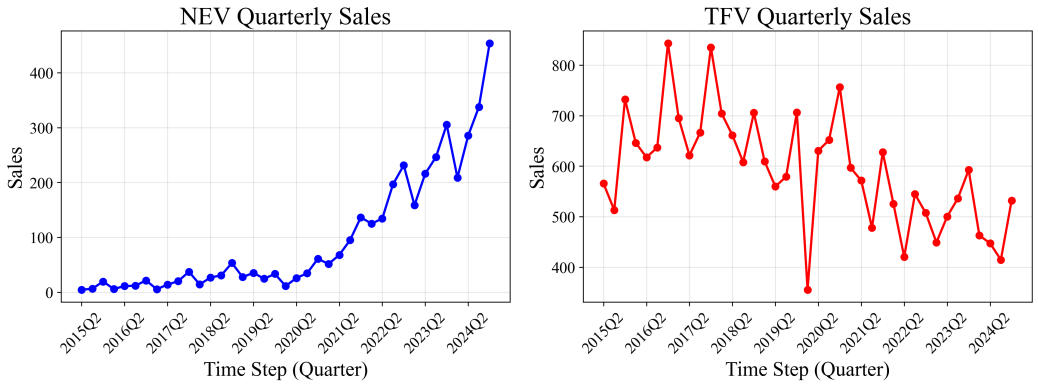


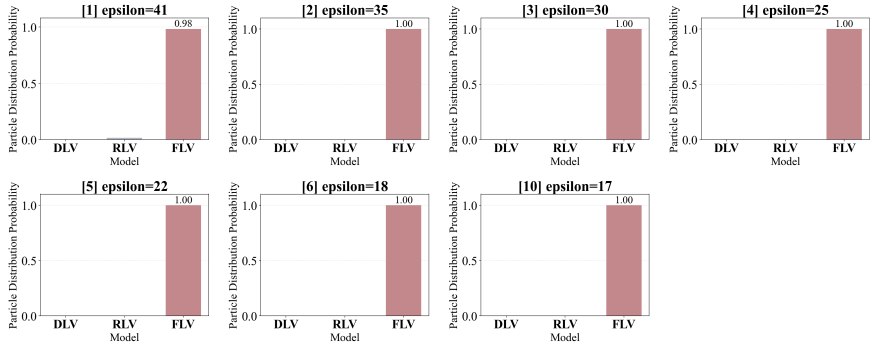
Figure 7. Sales trend curves for NEVs and TFVs in actual data.

Figure 7 reveals a pronounced upward trend in NEV sales, reflecting the rapid expansion of the new energy vehicle market. TFV sales, in contrast, exhibit relatively stable fluctuations but show a declining trajectory in the later period. The dynamic shifts in sales volumes for both vehicle categories mirror the competitive dynamics between new energy and traditional energy technologies within the automotive market, providing an ideal data foundation for the empirical application of the LV competition model.

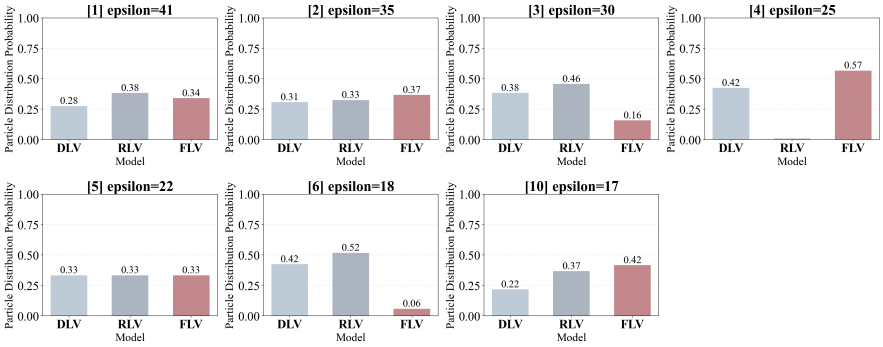
For DLV and FLV, this paper adopts uniform distributions as the prior distributions for all parameters. For RLV, most parameters also employ uniform distributions; however, to enhance parameter estimation accuracy, the random noise standard deviations γ_1 and γ_2 utilize a lower-anchored truncated half-normal distribution as their prior (i.e., a half-normal distribution truncated at zero, with mode at zero), which concentrates probability mass near the lower bound to encode shrinkage priors. Furthermore, this paper adapts the prior distribution ranges based on actual data characteristics to ensure that the initial values and noise parameters align with the scale of the real data. Specifically, the bounds for initial states (X_0, Y_0) and noise parameters (γ_1, γ_2) are determined adaptively from the observed data ranges, while maintaining sufficient breadth to avoid over-constraining the parameter space, thereby providing an appropriate starting point for parameter exploration in the ABC-PMC algorithm.

Additionally, the threshold sequence in this empirical study was generated adaptively. By independently sampling the model's prior distribution 800 times and calculating the corresponding Euclidean distances, the minimum threshold and initial threshold were established. Subsequently, an iterative threshold was generated using a geometric decay strategy. The resulting threshold sequence is $\varepsilon = \{41.45, 35.24, 29.95, 25.46, 21.64, 18.39, 17.27\}$.

In the Softmax-based ABC-PMC algorithm, the target acceptance rate per iteration is set to 120 particles, with the Softmax agent's temperature parameter τ at 0.2 and learning rate α at 0.1. Similarly, the empirical study section compares the performance of this algorithm against the standard ABC-PMC algorithm. Figure 8 illustrates the iterative particle distribution plots for both algorithms. The results demonstrate that the Softmax-based ABC-PMC algorithm still converges rapidly and selects models effectively on real data. Specifically, Figure 8(a) indicates that the algorithm tends to select Model 3, the FLV model.



(a) RL(Softmax)-ABC-PMC algorithm: iterations 1–7 with thresholds $\varepsilon = 41.45, 35.24, \ldots, 17.27$.



(b) Standard ABC-PMC algorithm: same threshold sequence without adaptive model selection.

Figure 8. Comparison of iterative particle distributions for model selection using RL(Softmax)-ABC-PMC and standard ABC-PMC algorithms on real automotive market data.

5.1. Parameter interpretation and competitive dynamics

As shown in Table 7, the estimated fractional order for NEVs is $\alpha \approx 0.912$, which is notably close to 1. This result indicates that the growth trajectory of NEVs exhibits strong path dependence. In economic terms, this suggests that consumer adoption patterns, the effects of continuous policy support, and the gradual expansion of charging infrastructure do not decay rapidly. Instead, their influence accumulates and persistently drives market growth. The near-unity order implies that the NEV market's current state is shaped by its entire history, making it a system with near "full memory."

In contrast, the estimated fractional order for TFVs is $\beta \approx 0.600$, reflecting a significantly weaker memory effect. As a mature market, TFV sales are more responsive to recent shocks-such as short-term economic cycles, fluctuations in fuel prices, or periodic regulatory changes, while the influence of distant past states decays substantially. This distinction in memory characteristics highlights a fundamental difference in the underlying dynamics of the two markets.

Beyond the memory effects, the competition parameters also reveal meaningful economic insights. The estimated competition coefficient from NEVs to TFVs is $e = 0.00541$, which is more than five times larger than the reverse competition coefficient $c = 0.001$. This asymmetry indicates that the expansion of NEVs exerts a significantly stronger "crowding-out" effect on TFVs than the reverse. This finding aligns with the economic theory of disruptive innovation, where an emerging technology

(NEVs) displaces an incumbent (TFVs) in an asymmetric manner. Together, these parameter estimates not only validate the FLV model as the best-fitting structure but also provide a quantitative foundation for understanding the ongoing structural transformation in China's automotive market.

Table 7. Parameter estimation of the FLV model obtained via the RL(Softmax)-ABC-PMC algorithm for actual sales changes of NEV and TFV.

Parameters	Mean	Std Dev	MC Error	2.5% Quant.	Median	97.5% Quant.
a	5.47E-01	9.87E-02	3.97E-03	1.00E-03	5.47E-01	9.99E-01
b	9.87E-03	1.16E-03	4.66E-05	1.00E-03	9.87E-03	2.00E-02
c	1.00E-03	1.07E-04	4.30E-06	1.00E-03	1.00E-03	1.00E-02
d	5.00E-02	1.64E-03	6.58E-05	5.00E-02	5.00E-02	1.20E+00
e	5.41E-03	2.74E-04	1.10E-05	1.00E-03	5.41E-03	2.00E-02
α	9.12E-01	2.01E-02	8.07E-04	6.00E-01	9.12E-01	9.90E-01
β	6.00E-01	4.64E-02	1.86E-03	6.00E-01	6.00E-01	9.90E-01

Based on the parameter estimates, the final FLV model for NEV and TFV sales is:

$$\begin{cases} D_t^{0.912}x(t) = x(t)(0.547 - 0.00987x(t)) - 0.001x(t)y(t) \\ D_t^{0.6}y(t) = -0.05y(t) + 0.00541x(t)y(t) \end{cases}. \tag{5.1}$$

5.2. Robustness of model selection

To ensure that the selection of the FLV model is not an artifact of a specific sample period, particularly the volatile early years of NEV market growth, we conducted a robustness test by shifting the data window. We excluded the first two years of data (2015 and 2016) and re-ran the model selection procedure using a truncated sample from the first quarter of 2017 to the fourth quarter of 2024, comprising 32 observation points. All algorithm settings remained unchanged.

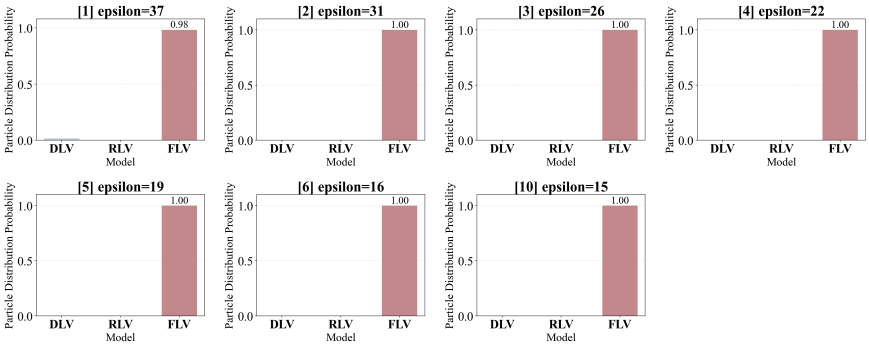


Figure 9. Model selection probabilities under the shifted data window (2018Q1-2024Q4). The FLV model remains dominant with posterior probability > 0.95.

The results, illustrated in Figure 9, demonstrate that the FLV model remains the optimal choice under this alternative data window, with its final posterior probability exceeding 0.98 even in the round 1, far above those of the DLV and RLV models. The parameter estimates remained stable: α was estimated at 0.78 and β at 0.72, closely matching the full-sample results. This robustness analysis

confirms that the strong memory effect identified in the NEV market is a persistent structural feature rather than a transient phenomenon driven by early-stage volatility. Consequently, the model selection results presented in this study are stable and reliable.

6. Conclusions

This paper proposes an adaptive ABC-PMC algorithm (RL(Softmax)-ABC-PMC) incorporating a Softmax decision mechanism for efficient Bayesian model selection. By introducing Softmax agents into the standard ABC-PMC framework, this method achieves dynamic adaptive allocation of computational resources, significantly enhancing model selection efficiency. Experimental results demonstrate that the proposed strategy exhibits significant advantages: In the S-shaped growth curve model selection experiment, the algorithm rapidly converged to the true model (modified exponential model), whereas traditional algorithms failed to identify the true model promptly; in the simulation study with three LV variants (discretized, randomized, and fractional-order), the algorithm successfully identified the data-generating DLV model, demonstrating its ability to distinguish between models with subtle structural differences. Furthermore, the algorithm accurately estimated model parameters, with estimates closely matching true values and exhibiting low uncertainty. The empirical study on Chinese automotive market sales data further validated the algorithm's effectiveness and practicality in real-world scenarios. The algorithm selected the FLV model, which captures memory effects in the competitive dynamics between NEVs and TFVs. The fractional orders ($\alpha \approx 0.912, \beta \approx 0.600$) quantify the degree of temporal memory dependence, while the deterministic parameters reflect the growth and interaction mechanisms. This selection demonstrates the algorithm's capability to identify models that best explain real-world data characteristics, even when multiple candidate models are structurally similar. This strategy holds potential for extension to broader computational problems, including other physics-based models, equation discovery problems, and neural network architecture selection. Future research may explore the handling of time-varying parameters, more efficient ABC sampling strategies (e.g., Monte Carlo Markov chain methods), and setting the temperature parameter τ and learning rate α as adaptive parameters to further enhance the algorithm's robustness and generalization capabilities.

In future work, we will systematically investigate the combination of RL methods (such as Softmax, UCB, Thompson sampling, and so on) with other state-of-the-art likelihood-free inference approaches, including adaptive ABC methods (e.g., adaptive SMC-ABC), random forest ABC (RF-ABC), and neural SBI methods. Through comprehensive benchmarking on model selection tasks with varying complexity and dimensionality, we aim to clarify the strengths and limitations of each method and provide practical guidance for applied researchers.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This work is supported by National Natural Science Foundation of PR China (Grant No.12301524).

Conflict of interest

The authors declare there are no conflicts of interest.

References

1. A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, D. B. Rubin, *Bayesian Data Analysis*, 3nd edition, Chapman and Hall/CRC, New York, 2013. https://doi.org/10.1201/b16018
2. M. A. Beaumont, Approximate Bayesian computation in evolution and ecology, *Annu. Rev. Ecol. Evol. Syst.*, **41** (2010), 379–406. https://doi.org/10.1146/annurev-ecolsys-102209-144621
3. G. Parisi, Nobel lecture: Multiple equilibria, *Rev. Mod. Phys.*, **95** (2023), 030501. https://doi.org/10.1103/RevModPhys.95.030501
4. K. Csilléry, M. Blum, O. E. Gaggiotti, O. François, Approximate Bayesian computation (ABC) in practice, *Trends Ecol. Evol.*, **25** (2010), 410–418. https://doi.org/10.1016/j.tree.2010.04.001
5. J. Marin, P. Pudlo, C. P. Robert, R. J. Ryder, Approximate Bayesian computational methods, *Stat. Comput.*, **22** (2012), 1167–1180. https://doi.org/10.1007/s11222-011-9288-2
6. P. Marjoram, J. Molitor, V. Plagnol, S. Tavaré, Markov chain Monte Carlo without likelihoods, *Proc. Natl. Acad. Sci. U.S.A.*, **100** (2003), 15324–15328. https://doi.org/10.1073/pnas.0306899100
7. M. A. Beaumont, W. Zhang, D. J. Balding, Approximate Bayesian computation in population genetics, *Genetics*, **162** (2002), 2025–2035. https://doi.org/10.1093/genetics/162.4.2025
8. S. A. Sisson, Y. Fan, M. M. Tanaka, Sequential Monte Carlo without likelihoods, *Proc. Natl. Acad. Sci. U.S.A.*, **104** (2007), 1760–1765. https://doi.org/10.1073/pnas.0607208104
9. T. Toni, D. Welch, N. Strelkowa, A. Ipsen, M. Stumpf, Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems, *J. R. Soc. Interface*, **6** (2009), 187–202. https://doi.org/10.1098/rsif.2008.0172
10. P. Del Moral, A. Doucet, A. Jasra, An adaptive sequential Monte Carlo method for approximate Bayesian computation, *Stat. Comput.*, **22** (2012), 1009–1020. https://doi.org/10.1007/s11222-011-9271-y
11. D. T. Frazier, G. M. Martin, C. P. Robert, J. Rousseau, Asymptotic properties of approximate Bayesian computation, *Biometrika*, **105** (2018), 593–607. https://doi.org/10.1093/biomet/asy027
12. C. Drovandi, D. T. Frazier, A comparison of likelihood-free methods with and without summary statistics, *Stat. Comput.*, **32** (2022), 42. https://doi.org/10.1007/s11222-022-10092-4
13. S. Wiqvist, J. Frellsen, U. Picchini, Sequential neural posterior and likelihood approximation, preprint, arXiv:2102.06522.
14. J. Lueckmann, J. Boelts, D. S. Greenberg, P. J. Gonçalves, J. H. Macke, Benchmarking simulation-based inference, preprint, arXiv:2101.04653.
15. K. Cranmer, J. Brehmer, G. Louppe, The frontier of simulation-based inference, *Proc. Natl. Acad. Sci. U.S.A.*, **117** (2020), 30055—30062. https://doi.org/10.1073/pnas.1912789117

16. X. Wang, R. P. Kelly, D. J. Warne, C. Drovandi, Preconditioned neural posterior estimation for likelihood-free inference, preprint, arXiv:2404.13557.

17. C. K. Williams, C. E. Rasmussen, *Gaussian Processes for Machine Learning*, MIT Press Cambridge, MA, USA, 2006. https://doi.org/10.7551/mitpress/3206.001.0001

18. W. Zhang, W. Gu, Machine learning for a class of partial differential equations with multi-delays based on numerical Gaussian processes, *Appl. Math. Comput.*, **467** (2024), 128498. https://doi.org/10.1016/j.amc.2023.128498

19. W. Shi, Study of dynamic systems based on Lotka-Volterra models and machine learning methods, in *2024 5th International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)*, IEEE, (2024), 528–534. https://doi.org/10.1109/ICBAIE63306.2024.11116819

20. K. N. Dinh, C. Liu, Z. Xiang, Z. Liu, S. Tavare, Approximate Bayesian Computation sequential Monte Carlo via random forests, *Stat. Comput.*, **35** (2025), 219. https://doi.org/10.1007/s11222-025-10748-x

21. A. B. Abdessalem, N. Dervilis, D. Wagg, K. Worden, Model selection and parameter estimation of dynamical systems using a novel variant of approximate Bayesian computation, *Mech. Syst. Signal Process.*, **122** (2019), 364–386. https://doi.org/10.1016/j.ymssp.2018.12.048

22. C. P. Robert, J. M. Cornuet, J. M. Marin, N. S. Pillai, Lack of confidence in approximate Bayesian computation model choice, *Proc. Natl. Acad. Sci. U.S.A.*, **108** (2011), 15112–15117. https://doi.org/10.1073/pnas.1102900108

23. A. Grelaud, J. M. Marin, C. P. Robert, F. Rodolphe, J. F. Taly, ABC methods for model choice in Gibbs random fields, *C.R. Math.*, **347** (2009), 205–210. https://doi.org/10.1016/j.crma.2008.12.009

24. X. Didelot, R. G. Everitt, A. M. Johansen, D. J. Lawson, ABC Likelihood-free estimation of model evidence, *Bayesian Anal.*, **6** (2011), 49–76. https://doi.org/10.1214/11-BA602

25. A. Nandy, C. Kumar, D. Mewada, S. Sharma, Bayesian optimization—multi-armed bandit problem, preprint, arXiv:2012.07885.

26. R. S. Sutton, A. G. Barto, *Reinforcement Learning: An Introduction*, 2^{nd} edition, The MIT Press , 2018.

27. T. G. Ritto, S. Beregi, D. Barton, Reinforcement learning and approximate Bayesian computation for model selection and parameter calibration applied to a nonlinear dynamical system, *Mech. Syst. Signal Process.*, **200** (2023), 110458. https://doi.org/10.1016/j.ymssp.2023.110458

28. U. Simola, J. Cisewski-Kehe, M. U. Gutmann, J. Corander, Adaptive approximate Bayesian computation tolerance selection, *Bayesian Anal.*, **16** (2021), 397–423. https://doi.org/10.1214/20-BA1211

29. J. Zhao, M. Yang, Y. Zhao, X. Hu, W. Zhou, H. Li, MCMARL: Parameterizing value function via mixture of categorical distributions for multi-agent reinforcement learning, *IEEE Trans. Games*, **16** (2024), 556–565. https://doi.org/10.1109/TG.2023.3310150

30. B. Jin, X. Xu, Machine learning-based forecasts of residential property prices in Hangzhou City, Zhejiang Province, China, *Neural Comput. Appl.*, **37** (2025), 4971–4988. https://doi.org/10.1007/s00521-024-10726-w

AIMS Press