M2 M2A

Internship Report

Reinforcement with Chemical Reaction Networks and Application to Phototaxis Modeling

Students :
Ruyi Tang
Supervisors:
Prof. Grégoire Sergeant-Perthuis, Prof. David Colliaux
Organizations:
Sorbonne University & Sony CSL
Duration:
April 2025 —October 2025

October 31, 2025

Abstract

Organisms navigate using noisy, partial sensory cues. We study algal phototaxis as sequential decision-making under uncertainty and develop a framework that couples a POMDP with chemical-reaction-network ODEs (CRN-ODEs) to bridge algorithm and biochemistry. We design a POMDP framework that formalizes hidden state, action, transition, observation/emission, and exploitation-exploration reward; we formulate memoryless belief update and one-step-look-ahead algorithm for action scoring; we build biophysical observation models for both single- and multi-source scenes and compare two aggregation schemes. Action values combine exploitation (immediate expected reward) with curiosity (mutual information gain). Since logarithms in mutual information are not mass-action polynomials, we approximate it with a polynomial upper bound and implement matched numerical and CRN-ODE pipelines.

We then learn policies and rewards. First, with a hand-specified exploitation reward we execute a one-step look-ahead policy in simulation and via CRN-ODEs,. Next, using data from trained POMDP policy we apply inverse reinforcement learning on the exploitation component to recover a reward vector and evaluate behavioral fidelity. Finally, we benchmark against two Gillespie/SSA variants (standard instantaneous reorientation and a modified run-while-tumble CTMC). Results show that, despite an initially over-estimated tumble ratio from the learned policy, the alignment-to-light distributions closely match benchmarks and real data; temperature calibration of the soft policy reduces the global tumble ratio to 0.012 (vs. data 0.027), revealing a trade-off between directional alignment and tumble frequency. CRN-ODE simulations reproduce the numerical pipeline's behavior but incur higher runtime due to steady-state integration.

Our contributions are: (i) a one-step POMDP decision module executable by CRN-ODEs (including a curiosity upper bound); (ii) a curated algal-phototaxis dataset and a benchmark across several models, including an end-to-end learned policy. This provides a computational route to link biochemical dynamics with behavioral optimality and sets the stage for task-guided IRL (MMV/MMFE/MCE) and offline-optimal control comparisons in future work.

We thank Pierre Bessière and Jacques Droulez for insightful discussions and comments that shaped the theoretical framework and experimental design.

This internship was funded by AMIES – PEPS project « Generative models of chemotaxis and phototaxis ».

Contents

List of Figures

1. Introduction

Across scales —from insects tracking turbulent plumes to bacteria climbing microscopic gradients —organisms navigate using chemical cues in complex environments. Biological systems must make optimal decisions when facing uncertain and incomplete environmental information. Recent research has demonstrated that probabilistic reasoning is an efficient way for organisms to handle such uncertainty.

Following [7], which introduced the theoretical framework proposing that cell signaling can function as probabilistic computers, it has been proved that molecular-level biochemical cascades can perform Bayesian inference where populations of macromolecules and diffusible messengers can compute any Rational Function with Non-negative Coefficients (RFNC) via elementary biochemical motifs,suggesting a path from abstract inference to biochemical implementations and offering a perspective on intelligent behaviors in biological systems.

Existing research primarily emphasizes theoretical analysis of static probabilistic inference problems; integrating sequential decision-making with partial observations and exploration bonuses within biochemically executable dynamics remains an open direction to explore. Furthermore, existing models fail to adequately consider curiosity-driven mechanisms in organisms' environmental exploration and their impact on learning efficiency.

This research aims to construct an integrated framework combining Partially Observable Markov Decision Processes (POMDP) and Chemical Reaction Networks (CRN), using unicellular algae phototaxis behavior as the research subject to explore biochemical-level reinforcement learning mechanisms. In this framework: POMDP formalizes sequential decision-making under partial observations, CRNs offer biochemical modules and ODE realizations that approximate the learning behavior of the agent. We focus on single- vs multi-source phototaxis scenarios, where observation design is pivotal. For multi-source aggregation, we compare a simple weighted scheme with an alternative aggregator and empirically study their statistical effects. We combine exploitation (immediate expected reward) with exploration (curiosity/information gain), computed both numerically and via CRN-ODE realizations.

Specifically, this research includes two axes: (i) *CRN-based decision modules*—we map belief updates, one-step action scoring (exploitation + curiosity), and observation aggregation into mass-action-implementable CRN-ODE components; and (ii) *data-driven behavior modeling and benchmarking*—we learn policies/rewards from experimental trajectories and benchmark against SSA/Gillespie baselines using predefined metrics (tumble ratio, alignment distributions, CDF/Q–Q). Building on this, our contributions are:

1. an observation and emission design for phototaxis with single-source and multi-source schemes and analysis of their statistical implications;

2. a curiosity (information-gain) objective with a polynomial upper bound suitable for CRN-ODE realization and aligned with the numerical pipeline;

3. policy learning with the constructed POMDP and comparisons with real experimental data and Gillespie-based simulations.

This work offers a computational implementation that aligns biochemical dynamics with behavioral decision-making. By modeling reinforcement learning processes at the molecular level, we can not only gain insights into decision-making mechanisms in unicellular organisms but also potentially provide theoretical foundations for designing novel biologically-inspired artificial intelligence algorithms. Furthermore,after validating the effectiveness of the framework in modeling algae phototaxis, this research has potential applications in synthetic biology, bioengineering, and other related fields that can be explored.

In the following chapters, we address our topic in detail as below:

- Chapter 2 reviews the mathematical framework for decision-making under partial observability and existing related works on CRN-based computation;
- Chapter 3 formulates phototaxis and details single- and multi-source observation and emission design;
- Chapter 4 presents the numerical RL pipeline with curiosity and the CRN-ODE implementation;
- Chapter 5 demonstrates experiments on phototaxis simulation under multiple scenarios and POMDP policy-learning with real data from biological experiments;
- Chapter 6 discusses implications and limitations, outlines future exploration directions

All code and scripts are available at `github.com/giveyourselfaTRY/phototaxis-pomdp-crn`.

2. Related Works

2.1 Brief overview on RL for phototaxis

Reinforcement learning (RL) formalizes sequential decision-making under uncertainty: an agent interacts with an environment, selects actions according to a policy, and receives rewards while trading off exploitation and exploration. In biological setting, often replete with noise and lack full access to the complete observability, reinforcement learning models are modulated, and have proven especially useful for explaining adaptive behavior across levels: from human motor learning [5] and sensory-motor decisions in natural behavior [14] to cellular navigation of chemotaxis and phototaxis [9]. Reinforcement learning and the partially observable perspective have been applied across sophisticated biological decision tasks.

In the decision task of phototaxis navigation, an unicellular organism must infer where the light source is and how to steer in its direction with noisy signal perceived by its eyespot while moving in a changing field underwater, making the task partially observable. Studies have repeatedly shown that decision-theoretic control improves navigation with noisy cues. For instance, in sperm chemotaxis, a strategy that switches steering gains depending on uncertainty outperforms fixed controllers—an explicitly optimal-policy result derived within a Markov decision framework and validated in simulations. This reinforces the value of casting gradient-seeking navigation as optimal control under uncertainty [9]. Beyond chemotaxis, phototaxis at the cellular and community levels has been quantitatively modeled and imaged , highlighting the roles of sensing, motility, and collective effects—ingredients that a partially observable RL framework can capture and modulate via the biophysical observation model, action set, and transition dynamics [12].

In partially observable settings for decision-making and explorative learning tasks, a normative approach is to maintain an explicit belief —a probability distribution over hidden states —and to choose actions over policy as a function of this belief. Several studies adopt this explicit-belief route under MDP/POMDP framework. In the task of human motor decision-making, [5] formalized the problem under POMDP framework and derived a belief-action value to select action and obtained belief from Bayesian inference. They used QMDP to get approximated solutions of POMDP, which is stated as a "belief state" MDP. In the aspect of conservation biology, [11] shows an example of continuous-state POMDP for monitoring task by visualizing policy graphs over beliefs to obtain the optimal POMDP.

In our research on phototaxis task, partial observability is inherent as the algae (agent) does not have direct access to the position of single or multiple light sources, observations are noisy and geometry-dependent, and actions reshape possibilities for the next moves. Following above literature on reinforcement learning and explicit belief, we model phototaxis as a POMDP, perform memoryless version of Bayesian filtering to maintain beliefs and optimize policies in belief space; this enables principled comparison of observation

designs for different experimental scenarios and the incorporation of information gain / curiosity as exploration bonuses alongside task exploitation rewards.

2.2 Chemical Reaction Networks

2.2.1 Overview of chemical reaction networks

A CRN(chemical reaction networks) is a set of chemical reactions composed by species (molecules). In [2] researchers give the following definition of CRN:

Definition 2.2.1 (CRN)**.** A CRN is a sorted pair $\mathcal{N} = (\Lambda, R)$, with Λ a finite set as the *species* of $\mathcal{N}$, and R a finite set of reactions over Λ.

Each reaction is abstracted into reactant r/product p vectors, which can be encoded as columns in a stoichiometry matrix M. From mass-action kinetics and the matrix M, we derive a system of ODEs describing concentration changes over time. Solving the ODE system, especially as $t \to \infty$ yields the steady-state concentrations of species.

To be more exact, in the continuous CRN model, the system state is a concentration vector $\mathbf{c}(t) \in \mathbb{R}^{\Lambda}_{\geq 0}$ evolving over time $t \geq 0$.
Let Λ be the species set, R the reaction set. For each reaction $\alpha \in R$, $\mathbf{r}_\alpha(\cdot)$ and $\mathbf{p}_\alpha(\cdot)$ denote the stoichiometric coefficients of reactants and products, respectively; for any $Y \in \Lambda$, $\mathbf{r}_\alpha(Y)$ is the count of Y as a reactant, and $\mathbf{p}_\alpha(Y)$ as a product. Under mass-action kinetics, the reaction rate is $v_\alpha(c) = k_\alpha \prod_{Y\in\Lambda} c_Y^{\mathbf{r}(Y)}$, where $c_Y = [Y]$ is concentration of Y, k_α is the rate constant.

Thus, for each species X:

$$\frac{d[X]}{dt} = \sum_{\alpha=(\mathbf{r}_\alpha,\mathbf{p}_\alpha)\in R} (\mathbf{p}_\alpha(X) - \mathbf{r}_\alpha(X))v_\alpha(c) = \sum_{\alpha=(\mathbf{r}_\alpha,\mathbf{p}_\alpha)\in R} k_\alpha M_{X,\alpha} \prod_{Y\in\Lambda} [Y]^{\mathbf{r}_\alpha(Y)}$$

where $M_{X,\alpha} = \mathbf{p}_\alpha(X) - \mathbf{r}_\alpha(X)$ denotes the stoichiometry matrix (rows = species, columns = reactions).

We can take the following example in [2] to get an idea.

Example Given reactions: $\alpha = A + B \to C$ and $\beta = 2C + B \to 2A + B$. We can obtain a stoichiometry matrix:

$$M = \begin{pmatrix} -1 & 2 \\ -1 & 0 \\ 1 & -2 \end{pmatrix}$$

where the rows are species (A, B, C) and columns are reactions $\alpha = A + B \to C$ and $\beta = 2C + B \to 2A + B$.Hence $M_{\cdot,\alpha} = (-1, -1, 1)^\top$, $M_{\cdot,\beta} = (+2, 0, -2)^\top$. The mass-action ODEs read:

$$\frac{d[A]}{dt} = -k_\alpha[A][B] + 2k_\beta[C]^2[B]$$
$$\frac{d[B]}{dt} = -k_\alpha[A][B]$$
$$\frac{d[C]}{dt} = k_\alpha[A][B] - 2k_\beta[C]^2[B]$$

And if we have set a target species (for example A.) then we have its corresponding steady state function by taking the limit as $t \to \infty$, meaning that we set all the derivatives to 0 solve the resulting system. Though steady states are zeros of polynomial equations, but the dynamic can be unstable.

And we would also call this deducted steady state function *chemically computable* with the following definition [2]:

Definition 2.2.2 (Chemically Computable)**.** A function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is *chemically-computable* if there exists a CRN (Λ, R) and an input encoding $q : \mathbb{R}_{\geq 0} \to \mathbb{R}^U_{\geq 0}$ (for some $U \subset \Lambda$) and an output species S such that for every $z \in \mathbb{R}_{\geq 0}$, the solution $c(t)$ with $c_U(0) = q(z)$ as initial condition of CRN-ODE derived above:

$$\lim_{t\to\infty} c_S(t) = f(z)$$

And this function corresponds with the RFNCs(Rational function with non-nrgative coefficients) that we derive from the biochemical cascades in [7]:

Proposition 2.2.3. *any RFNC can be realized at steady state by a mass-action network/ biochemical cascades.*

2.2.2 From enzymatic reactions to CRN

We follow the motif in [7] that introduces an enzymatic futile cycle as follows:

$$\varnothing \underset{M_r^*}{\overset{M_p^*}{\rightleftharpoons}} S$$

In this cycle, the enzymes M_p^* and M_r^* respectively regulate the production and degradation of messenger S in cell signaling. And $\varnothing$ here denotes the implicit production source or degradation sink often used to abstract production and decay. More explicitly, we can align the above cycle with the canonical enzymatic reactions introduced in [8] under the format of:

$$E + W \underset{k_r}{\overset{k_p}{\rightleftharpoons}} C \xrightarrow{k_{cat}} E + W^*$$

We can decompose the futile cycle by rewriting the reactions with the above enzymatic reactions template.

We have the following proposition:

Proposition 2.2.4. *Given* $E + W \underset{k_r}{\overset{k_p}{\rightleftharpoons}} C \xrightarrow{k_{cat}} E + W^*$, *rename* W^* *as* S *and introduce two enzyme forms* E_p, E_r *to obtain production reaction:* $M_p^* + W \underset{d_1}{\overset{a_1}{\rightleftharpoons}} M_p^*W \xrightarrow{k_1} M_p^* + S$ *and degradation reaction:* $M_r^* + S \underset{d_2}{\overset{a_2}{\rightleftharpoons}} M_r^*S \xrightarrow{k_2} M_r^* + W$.

Here W^* corresponds to the activated messenger S,E_p, E_r are the production/degradation enzymes;C_p, C_r are intermediate complexes. Under mass-action kinetics with consistent rate constants, the resulting $\frac{d[S]}{dt}$ coincides with the net production rate of W^* in the template.

From the above reactions, our goal is to study the value of the product S at steady state.

1. **In production:** we first extract the reaction $M_p^*W \xrightarrow{k_1} M_p^* + S$, and we have:

$$\frac{d[S]}{dt} = k_1[M_p^*W]$$

To compute $[M_p^*W]$ we have:

$$M_p^* + W \xrightarrow{a_1} M_p^*W$$

$$M_p^*W \xrightarrow{d_1} M_p^* + W$$

And at equilibrium, we have the equation:

$$a_1[M_p^*][W] = d_1[M_p^*W]$$
$$[M_p^*W] = \frac{a_1}{d_1}[M_p^*][W]$$

And therefore we get:

$$\frac{d[S]}{dt} = k_1\frac{a_1}{d_1}[M_p^*][W]$$

2. **In degradation:** similarly, we first consider the reactions:

$$M_r^* + S \xrightarrow{a_2} M_r^*S$$

$$M_r^*S \xrightarrow{d_2} M_r^* + S$$

and we have:

$$\frac{d[S]}{dt} = -a_2[M_r^*][S] + d_2[M_r^*S]$$

And from statement given by [8], at steady state we have $k_1[M_p^*W] = k_2[M_r^*S]$, so the above equation can be written as follows:

$$\frac{d[S]}{dt} = -a_2[M_r^*][S] + d_2\frac{k_1a_1}{k_2d_1}[M_p^*][W]$$

3. **Finally:** We can sum up the above computation from production and reduction, and finally we have:

$$\begin{aligned}\frac{d[S]}{dt} &= k_1\frac{a_1}{d_1}[M_p^*][W] - a_2[M_r^*][S] + d_2\frac{k_1a_1}{k_2d_1}[M_p^*][W] \\ &= k_1\frac{a_1}{d_1}(1+\frac{d_2}{k_2})[M_p^*][W] - a_2[M_r^*][S]\end{aligned}$$

To simplify the above equation, we can write the above formula as: $\frac{d[S]}{dt} = k_p[M_p^*][W] + k_r[M_r^*][S]$.

When the production and reduction of the intermediate complex are much faster compared to catalysis, it is possible to ignore the complex, we can then rewrite the enzymatic futile cycle as:

$$M_p^* + \varnothing \xrightarrow{k_p} M_p^* + S$$

$$M_r^* + S \xrightarrow{k_r} M_r^* + \varnothing$$

If we consider the above reactions as a part of chemical reaction network, we then have the species as: $\{M_p^*, \varnothing, S, M_r^*\}$, and the stoichiometry matrix:

$$M = \begin{pmatrix} 0 & 0 \\ -1 & 1 \\ 1 & -1 \\ 0 & 0 \end{pmatrix}$$

Following the idea introduced in the previous section, for the concentration of the target product:

$$\frac{d[S]}{dt} = k_p[M_p^*] - k_r[M_r^*][S]$$

. This is the same formula that we have deducted. Therefore, the value of S at steady state we have:

$$S = \frac{k_p[M_p^*]}{k_r[M_r^*]}$$

2.2.3 CRN construction

Chemical reaction networks (CRNs) have been used to program computations over functions or distributions.Many different methods have already been proposed on how to build CRN with different input functions. Two classic strands are deterministic computation of semilinear functions and CRN synthesis for finite-support discrete distributions. These provide powerful templates for static computation, but offer limited support for online Bayesian inference and learning in sequential decision problems.

[4] proved that CRNs can deterministically compute semilinear functions. The authors demonstrated how to convert affine partial functions into CRNs by encoding output values as the difference between two monotonically increasing molecular species.Then they proposed a construction framework that decomposes semilinear functions into multiple

linear pieces, each described by affine partial functions, then uses semilinear predicates to decide which linear function to apply for given inputs.

Limitation of this method lies in focusing on static deterministic function computation, lacking the capability to handle stochastic processes and dynamic learning. For the probabilistic inference and online learning functions required in the reinforcement learning framework of this research, this method appears too rigid and cannot adapt to environmental changes and real-time parameter update requirements.

[3] further explored how to program CRNs from discrete probability distribution with finite support. The authors proposed decomposed target distributions into combinations of basic operations, with each operation corresponding to specific reaction patterns. Although this method performs well in probability distribution computation, its limitation also lies in handling static distribution computation rather than dynamic probabilistic inference and parameter learning.

Closer to our needs, our research mainly referred to the work of [13] that specifically addresses the parameter learning problem in Hidden Markov Models (HMMs), proposing a reaction network-based implementation of the Baum-Welch algorithm. The main contribution is transforming all steps of the traditional Baum-Welch algorithm (forward algorithm, backward algorithm, E-step, M-step) into chemical reaction network dynamical systems. Each variable is encoded by independent molecular species, and each reaction changes only one molecule of one species. This method particularly adopts a design similar to *futile cycles* in biochemical pathways, enabling forward and reverse changes through different enzyme sets. Theoretical analysis proves that every positive fixed point of the reaction network corresponds to a fixed point of the BW algorithm, ensuring algorithmic equivalence and convergence.

This paper directly addresses the parameter learning problem in dynamic probabilistic models, with reaction network designs that are closer to real biological systems. It provides complete construction methods for all four subnetwork modules of the BW algorithm, establishing the most suitable technical foundation for our research on online POMDP. We use a modular CRN-ODE design for belief filtering and value accumulation for immediate reward and curiosity. We also establish polynomial upper bound for mutual information. Our design of CRN-ODE pipeline finds middle ground between static CRN compilers and HMMCRN.

3. Problem Formulation

In the biological settings of phototaxis, the partial observability is the rule for the agent rather than exception as it possesses partial observability:

- the **hidden state** is not directly accessible for the algae(agent);
- **observations** are corrupted by sensory noise and biophysical geometry such as self-shadowing and occlusion etc.;
- **run-tumble actions** change both position and directing, affecting future sensory opportunities;
- **rewards** balance the exploitation and exploration (curiosity/information gain) for the task goal of approximating the light source.

A POMDP makes each of these components explicit with state, action,transition ,observation, emission, reward and policy, so that we can formulate the problem within one model-based, belief-based modulation.

3.1 Biophysical Model

Although Chlamydomonas has a single eyespot and swims while spinning in 3D, we use a 2D projection model: the incident flux at the eyespot is decomposed into left/right channels relative to the current heading. The 'two eyes' are a modeling surrogate, not an anatomical claim, to encode directionality and occlusion in 2D.

We consider the agent as a circular body of radius $R > 0$, with center $x_t \in \mathbb{R}^2$ and unit heading vector $v_t \in \mathbb{S}^1$ at time t. Two eye-spots are mounted on the rim at symmetric angles $\pm\gamma$; their optical axes are $\pm\alpha$ relative to v_t. Each eye only perceives directions within a half field-of-view FoV,centered on its optical axis. So we have the position of the left eye and the right eye as:

$$x^L_{eye}(t) = x_t + R\mathrm{Rot}_{+\gamma}v_t, \quad x^R_{eye}(t) = x_t + R\mathrm{Rot}_{-\gamma}v_t, \quad \text{where } \mathrm{Rot}_\theta = \begin{bmatrix}\cos\theta & -\sin\theta \\ \sin\theta & \cos\theta\end{bmatrix}.$$

where we have optical rim normal:$n_{L/R}(t) = \mathrm{Rot}_{\pm\gamma}v_t = \frac{x^{L/R}_{eye}(t)-x}{||x^{L/R}_{eye}(t)-x||}$

and the optical axes with offset $\pm\alpha$ indicates where each looks, denoted as u_L, u_R:

$$u_L(t) = \mathrm{Rot}_{+\alpha}v_t, \quad u_R(t) = \mathrm{Rot}_{-\alpha}v_t.$$

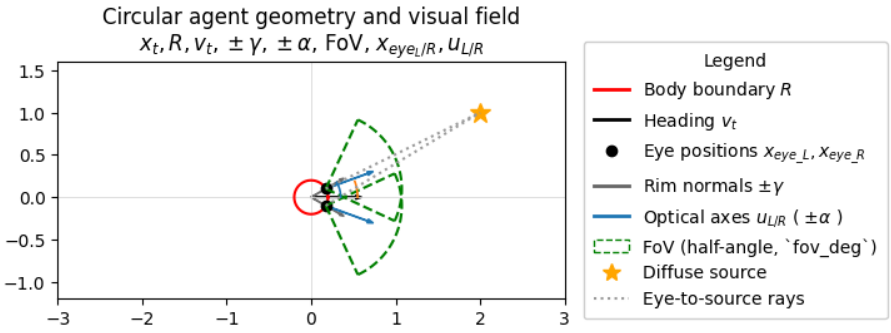


Figure 3.1: Illustration of the bio-physical agent

3.2 POMDP Framework

States At time t, the physical state of the agent is: $(x_t, v_t) \in \mathbb{R}^2 \times \mathbb{S}^1$ where x_t and v_t represent the position and direction of the agent. And the physical state of the agent is used to generate observations.

We also define the hidden state which is a discrete bin $h \in \{0, \ldots, n-1\}$ where n is the number of bins for hidden states. Each h is associated with an angle

$$\theta(h) = \frac{2\pi h}{n}$$

which is not inferred, but is used to construct the emission column $Z[:, h]$.

Actions We have a action set $\mathcal{A} = \{0, 1\}$, where we have: (different noise)

- $a = 0$: **run** — move one step forward with current noise-added direction
- $a = 1$: **tumble** — stay in the same position but randomly choose a direction

Transition We build an action-conditioned transition matrix $T \in \mathbb{R}^{a \times n \times n}$, which is column-stochastic. And we define it as:

$$T_i(h'|h) = \Pr(H_{t+1} = h' | H_t = h, a_t = i)$$

- **run:** The pre-normalization matrix is

  $$\widetilde{T}_{run} = I_n + 0.1\,\mathbf{1}\mathbf{1}^\top.$$

  And after column normalization:

  $$T_0(h'|h) = \frac{\delta_{h'h} + 0.1}{1 + 0.1n}.$$

  The matrix is diagonal dominant, meaning that the agent stays in the same state with high probability but plus small leakage.

- **tumble:** The pre-normalization matrix is

$$\widetilde{T}_{tumble} \;=\; 0.1\,\mathbf{1}\mathbf{1}^{\top}.$$

And after column normalization:

$$T_1(h'|h) = \frac{0.1}{0.1n} = \frac{1}{n}.$$

This matrix shows uniform randomization over all states.

Observations The purpose of the observation design is to map continuous geometry into a **2-channel discrete observation** $(o_1, o_2) \in \{0, \ldots, m-1\}^2$, where o_1, o_2 respectively represents the left and right light perception for the agent, and m bins are quantized **intensity levels** for each side.The raw photic signal is computed by a biophysical model that integrates the *distance attenuation* with diffusion of light, *angular sensitivity*, *visibility gates* (occlusion and field-of-view), then passed through **sensor saturation** and **quantization**.

single-source Recall the physical states defined in the biophysical model, we have agent center x, unit heading vector v, left and right eye spot position (x^L_{eye}, x^R_{eye}), rim normals (n_L, n_R) optical axis (u_L, u_R). Each eye has half field-of-view FoV.

For each eye $c \in \{L, R\}$ and light source center at μ, we define **direction-to-light unit vector**$\hat{r_c}$ and relative distance s_c:

$$\hat{r_c} = \frac{\mu - x^c_{eye}}{||\mu - x^c_{eye}||}, \qquad s_c = ||\mu - x^c_{eye}||.$$

Then we consider the **visibility gates** with **occlusion** and **field-of-view**:

$$g^c_{\text{occ}} = \mathbf{1}\{n_c^{\top}\hat{r_c} \geq 0\}, \qquad g^c_{\text{fov}} = \mathbf{1}\{u_c^{\top}\hat{r_c} \geq \cos FoV\}$$

The **occlusion gate** enforces that the line of sight to the light does not cross the opaque body (half-space test with the rim normal). And the **FoV gate** ensures the light is perceived within the field of vision.

And the light is only perceived when both the occlusion and field-of-view gates are satisfied; the **visibility gate** is:

$$g^c = g^c_{\text{occ}} \cdot g^c_{\text{fov}}.$$

As we intend to simulate the experiment **underwater**, we consider the diffusion of light by modeling a **base Gaussian field** and a **propagation distance attenuation**:

$$I^c_{\text{base}} = I_0 \cdot \exp(-\frac{s_c}{2\sigma^2}), \qquad a(s_c) = (1 + \exp(\beta_d s_c))^{-1}$$

where I_0 is the peak intensity at light source center μ and σ is constant parameters set in the environment configuration. β_d is a tunable parameter that controls how fast attenuation decays with distance.

We also define an **angular gain** that provides information on where the lights come in and hit the eyes using the angle between optical axis and the direction-to-light vector:

$$G_c = \max(0, u_c^\top \hat{r}_c)$$

Then we get the **raw intensity perceived per eye**:

$$I_c = g^c \cdot I^c_{\text{base}} \cdot a \cdot G_c, \qquad c \in \{L, R\}$$

This raw intensity is also interpretable physically as photon flux density incident on the surface of each eye.

To compress the dynamic range of raw intensity as well as to stabilize quantization under photon shot noise and match photoreceptor-like responses, we define a **logistic saturation** term:

$$y_c := S(I_c) = (1 + \exp[-\beta_s(I_c - I_{\text{thr}})])^{-1} \in [0, 1]$$

where I_{thr} is the half-response/semi-saturation: it is the input intensity at which $S(I_{\text{thr}}) = 0.5$. And finally we quantize the saturated sensor output into discrete observation bins:

$$o_c = \begin{cases} \mathbf{1}\{y_c \geq 0.5\}, & m = 2 \\ \text{bin}(y_c) - 1, & m \geq 3 \end{cases}$$

multi-source Set K light sources in the experiment, for $k \in \{1, \ldots, K\}$,for each light source center μ_k, we compute the saturated continuous outputs for left and right eye$y_L^{(k)}, y_R^{(k)} \in [0, 1]$ as before with single-source.

Instead of processing the exact observation from each light, we encode an aggregator to feed the agent with a weighted observation by a distance-weighted combination with distance attenuation defined above $w_k = a(s^k)$, where $s^k = ||\mu_k - x||$. Then, the weighted average:

$$y_c = \frac{\sum_k w_k y_c^k}{\sum_k w_k}, \qquad c \in \{L, R\}.$$

Finally we quantize the observation with bins as single-source case:

$$o_c = \begin{cases} \mathbf{1}\{y_c \geq 0.5\}, & m = 2 \\ \text{bin}(y_c) - 1, & m \geq 3 \end{cases}$$

multi-source (alternative) We propose another dominant-source selector as algorithmic aggregator for multi-source scenario. we score candidate sources (alignment/distance), keep only the top-scoring source's raw channels before binning, thereby avoiding synthetic mixed patterns from weighted averaging and preserving single-source-like emission statistics. This does not assume the sensor can identify source identities; it is a statistical approximation.

Then for each light, we propose 2 modes of **scoring**:

- **forward alignment:**

$$r^k = \frac{\mu_k - x}{||\mu_k - x||}, \qquad \mathrm{S}_k^{\text{alignment}} = \max(0, v^\top r^k)$$

  This score favors closer lights better aligned with current heading.

- **channelmax:**

$$\mathrm{S}_k^{\text{channelmax}} = \max(y_L^{(k)}, y_R^{(k)})$$

  This score favors closer lights that are perceived stronger on one side, encouraging the agent to be more determined to turn to one side.

  For example, when scoring on `"alignment"`, the agent might prefer the light farther in distance but better aligned with its heading, with higher chance of ignoring the less aligned but closer light. While the "left-right bias" score produces a clearer turning signal as it favors stimuli that are strongly unbalanced on one side of sensors.

After scoring with raw observations, we come to dominance competition and suppression, where we keep only dominant raw observations and all others are ignored. The winner is chosen as:

$$k^* = \arg\max_k S_k$$

And in case of multiple lights sharing the same score, we add a hierarchical tie-break so that the winner can be chosen in the order of :

1. $\epsilon-$**max candidate set:** keep all lights whose scores are within error ϵ of the global maximum:
$$\mathcal{C}_0 = \{k : S_k \geq S_{\max} - \epsilon\}$$
2. **Alignment:** Within candidates, retain those with the highest alignment to the heading direction:
$$\mathcal{C}_1 = \arg\max_{k \in \mathcal{C}_0} S_k^{\text{align}}$$
3. **Nearest-distance:** If ties remain, select those with smallest distance:
$$\mathcal{C}_2 = \arg\min_{k \in \mathcal{C}_1} s^k$$
4. **Inertia:** Keep the previous winner if still in the candidate set:
$$\texttt{use\_inertia} \wedge \texttt{prev\_idx} \in \mathcal{C}_2 \rightarrow k^* = \texttt{pred\_idx}$$
5. **Random:** Otherwise choose uniformly at random, else take smallest index $k^* = \min \mathcal{C}_2$ deterministically.

Finally, we quantize the winner's outputs to obtain the discrete two-channel observation as before:

$$o_c = \begin{cases} \mathbf{1}\{y_c^{k^*} \geq 0.5\}, & m = 2 \\ \mathrm{bin}(y_c^{k^*}) - 1, & m \geq 3 \end{cases}$$

and we have $o = (o_L, o_R) \in \{0, \ldots, m-1\}^2$.

By selecting a dominant light source as the winner before binning, the dominant-selector method avoids synthetic patterns from the previous method. It also preserves the single-source statistics like the emission matrix, while the same statistic might not be suitable for the per-channel max/average method.

Emission To build the emission matrix $Z \in \mathbb{R}^{m \times m \times n}$ that encodes the likelihood of observing o at hidden state h: $Z(o|h)$, we first recall that for each hidden state h, we have $\theta(h)$ as a discretized prototype angle:

$$\theta_h := \theta(h) = \frac{2\pi h}{n}, \quad h \in \{0, \ldots, n-1\}$$

and we also use a hard gate so the agent responds only when roughly facing the light:

$$g(\theta) = \mathbf{1}[\cos\theta \geq -0.01]$$

Then the response splits into left and right channels according to the sign of $\sin\theta_h$, and the raw channels we get are:

$$s_1(h) = \big[\sin\theta_h\big]_+ g(\theta_h), \qquad s_2(h) = \big[-\sin\theta_h\big]_+ g(\theta_h), \quad \text{where} \quad [\mathrm{z}]_+ = \max(\mathrm{z}, 0).$$

And we use the same function to bin the channels:

$$o_k(h) = \mathrm{bin}(\tilde{s}_k(h)) - 1, \quad k \in \{1, 2\}.$$

To make each hidden state h emit a dominant observation, we place a **spike** at the dominant bin pair and zero elsewhere:

$$\widetilde{Z}\big((o_1, o_2) \mid h\big) = \begin{cases} 10, & (o_1, o_2) = (o_1(h), o_2(h)), \\ 0, & \text{otherwise.} \end{cases}$$

Adding a uniform floor 0.1 yields

$$\widehat{Z}\big((o_1, o_2) \mid h\big) = \widetilde{Z}\big((o_1, o_2) \mid h\big) + 0.1.$$

Finally, we normalize each column (fixed h) to obtain a column-stochastic emission kernel:

$$Z\big((o_1, o_2) \mid h\big) = \frac{10 \cdot \mathbf{1}[(o_1, o_2) = (o_1(h), o_2(h))] + 0.1}{10 + 0.1\, m^2}.$$

And the emission matrix we have built for the POMDP framework is not action-dependent.

Belief update In our framework, the belief update is **memoryless**. This means reweighting a fixed prior by the instantaneous likelihood at each step, rather than rolling the previous posterior forward

At each step, we apply a single-step likelihood reweighting with a fixed uniform prior b_0, i.e. no prediction step and no rolling the posterior to the next step:

$$b_t^+(h) \;=\; \frac{Z[o_t, h]\; b_0(h)}{\sum_{j=1}^{n} Z[o_t, j]\; b_0(j)}, \quad o_t = (o_1, o_2)_t.$$

3.3 Policy

We adopt a **one-step look ahead policy**: for each action a, we form a predicted belief $b'_a = T_a b_0$ for scoring that includes the immediate expected reward and the mutual information (curiosity).

Immediate expected reward After taking action a, the next-state distribution is $b'_a = T_a b^+$ So the immediate value is the expectation of the state reward under this distribution with a reward vector $r : \mathcal{H} \to \mathbb{R}^{|\mathcal{H}|}$ which is a function on hidden states and here $|\mathcal{H}| = n$:

$$V^{(1)}(a) = \mathbb{E}_{h' \sim b'_a}[r] = \sum_{h'} r(h') b'_a(h') = \sum_{h'} \sum_{h} r(h') T_a(h'|h) b^+(h) = r(h')^\top T_a b^+$$

Curiosity Mutual information quantifies the expected information gain about hidden state H from observation O. We set:

$$p_O(o_1, o_2) = \sum_h Z_{o_1,o_2|h} b'_a(h)$$

$$\begin{aligned} V^{(2)}(a) &= I(H; O|b'_a) \\ &= \sum_{o_1,o_2,h} b'_a(h) D_{KL}(Z(:,:|h)||Z b'_a) \\ &= \sum_{o_1,o_2,h} b'_a(h) Z_{o_1,o_2|h} \log \frac{Z_{o_1,o_2|h}}{p_O(o_1,o_2)} \\ &= \sum_{o_1,o_2,h} b'_a(h) Z_{o_1,o_2|h} (\log Z_{o_1,o_2|h} + \log \frac{1}{p_O(o_1,o_2)}) \end{aligned}$$

Curiosity Upper Bound The curiosity/mutual information term contains logarithms and is not an RFNC. To enable mass-action realization, we replace it by a poly-

nomial RFNC upper bound and build the corresponding CRN-ODE.

$$\begin{aligned}
V^{(2)}(a) &= \sum_{o_1,o_2,h} b'_a(h) Z_{o_1,o_2|h} (\log Z_{o_1,o_2|h} + \log \frac{1}{p_O(o_1,o_2)}) \\
&\leq \sum_{o_1,o_2,h} b'_a(h) Z_{o_1,o_2|h} (Z_{o_1,o_2|h} + \frac{1}{p_O(o_1,o_2)} - 2) \qquad (\log x \leq x-1) \\
&\leq \sum_{o_1,o_2,h} b'_a(h) Z^2_{o_1,o_2|h} + \sum_{o_1,o_2,h} b'_a(h) \frac{Z_{o_1,o_2|h}}{p_O(o_1,o_2)} - 2 \sum_{o_1,o_2,h} b'_a(h) Z_{o_1,o_2|h} \\
&\leq \sum_{o_1,o_2,h} b'_a(h) Z^2_{o_1,o_2|h} + \sum_{o_1,o_2} \frac{\sum_h b'_a(h) Z_{o_1,o_2|h}}{\sum_g b'_a(g) Z_{o_1,o_2|g}} - 2 \sum_{o_1,o_2} \sum_h b'_a(h) Z_{o_1,o_2|h} \\
&\leq \sum_{o_1,o_2|h} b'_a(h) Z^2_{o_1,o_2|h} + \sum_{o_1,o_2} 1 - 2 \sum_{o_1,o_2} p_O(o_1,o_2) \\
&\leq \sum_{o_1,o_2,h} b'_a(h) Z^2_{o_1,o_2|h} + m^2 - 2
\end{aligned}$$

Policy The **total reward** V is defined as:

$$V(a) = V^{(1)}(a) + \lambda V^{(2)}(a).$$

where $\lambda \geq 0$ is a tunable hyperparameter to balance *exploitation and exploration.*

As for the selection of optimal action, we allow 2 choices:

- deterministic: $a^* = \arg\max_a V(a)$
- stochastic: $a^* \sim \pi(a) \propto e^{-V(a)/\gamma}$. Here we use a Boltzman sampling, which follows from entropy-regularized optimization: minimizing expected cost with an entropy (or KL-to-uniform) term yields a closed-form Boltzmann policy. When the $\tau \to 0$, the solution approaches to the deterministic argmax; $\tau \to \infty$ approaches a uniform policy.

4. RL algorithms implemented with CRN

In our research we have implemented a numerical pipeline and a biochemical pipeline with CRN-ODE under the demonstrated POMDP framework. Especially, the biochemical pipeline with chemical reaction networks marks the innovative design of this research.

4.1 Numerical Implementation

We implement a single POMDP model for phototaxis and study two objective variants that differ only in the curiosity term: one with the theoretical term of mutual information and the other with polynomial upper bound which is amenable to CRN implementation introduced in the next section.

Algorithm 1 Numerical POMDP Implementation (Exact Mutual Information)

Require: Action set $\mathcal{A} = \{0, 1\}$ (run/tumble); transition matrix $\{T_a\}_{a\in\mathcal{A}} \in \mathbb{R}^{N\times N}$; emission matrix $Z \in \mathbb{R}^{m\times m\times N}$; reward $r \in \mathbb{R}^N$; curiosity weight $\lambda \geq 0$; total steps T; fixed prior $b_0 \in \Delta^{N-1}$.

1: Initialize physical state (x_1, v_1); obtain first observation $o_1 = (o_{1,1}, o_{1,2}) \in \{0, \ldots, m-1\}^2$.

2: **for** $t = 1, \ldots, T$ **do**

3: **(Belief update)** With normalizer $\eta_t = \big(\sum_{j=1}^N Z[o_t|j]\, b_0(j)\big)^{-1}$, set

$$b_t^+(h) = \eta_t\, Z[o_t|h]\, b_0(h), \quad h = 1, \ldots, N.$$

4: **(Action scoring)** For each $a \in \mathcal{A}$, form the one-step prediction $b'_{t,a} = T_a\, b_t^+$ and compute

$$V_t^{(1)}(a) = r^\top b'_{t,a}, \qquad p_{O|a}(o) = \sum_{h=1}^N Z[o|h]\; b'_{t,a}(h), \quad o \in \{0, \ldots, m-1\}^2,$$

$$V_t^{(2)}(a) = \sum_o \sum_{h=1}^N b'_{t,a}(h)\, Z[o|h]\, \log\frac{Z[o|h]}{p_{O|a}(o)}, \qquad V_t(a) = V_t^{(1)}(a) + \lambda\, V_t^{(2)}(a).$$

5: **(Policy)** $a_t = \arg\max_{a\in\mathcal{A}} V_t(a)$, or $a_t \sim \pi(a) \propto e^{V_t(a)/\tau}$.

6: **(Environment step)** Evolve (x_{t+1}, v_{t+1}) under a_t; acquire o_{t+1}.

7: **end for**

As defined in the previous section of POMDP components, $\mathcal{A} = \{0 = run, 1 = tumble\}$, $\{T_a\}_{a\in\mathcal{A}} \in \mathbb{R}^{N\times N}$ are column-stochastic transitions, $Z[o|h] \in \mathbb{R}^{m\times m\times N}$ is the emission likelihood for $o = (o_1, o_2)$, $r \in \mathbb{R}^N$ is the exploitation reward vector; $\lambda \geq 0$ is the curiosity weight balancing exploration and exploitation, and $b_0 \in \Delta^{N-1}$ is the fixed uniform prior.

We adopt the emission-only correction $b_t^+(h) \propto Z[o_t|h]\, b_0(h)$ for the memoryless belief update and fold the prediction into action scoring via $b'_{t,a} = T_a\, b_t^+$.

Algorithm 2 Numerical POMDP Implementation (Polynomial Curiosity Bound)

Require: Same inputs as Alg. 1.

1: Initialize (x_1, v_1) and o_1.
2: **for** $t = 1, \ldots, T$ **do**
3: **(update)** $b_t^+(h) = \eta_t \, Z[o_t|h] \, b_0(h)$, with $\eta_t^{-1} = \sum_j Z[o_t|j] \, b_0(j)$.
4: **(Action scoring)** For each $a \in \mathcal{A}$, let $b'_{t,a} = T_a \, b_t^+$ and compute

$$V_t^{(1)}(a) = r^\top b'_{t,a}, \qquad \widetilde{V}_t^{(2)}(a) = \sum_o \sum_{h=1}^{N} b'_{t,a}(h) \, Z[o|h]^2 \; + \; (m^2 - 2),$$

$$\widetilde{V}_t(a) = V_t^{(1)}(a) + \lambda \, \widetilde{V}_t^{(2)}(a).$$

5: **(Policy)** $a_t = \arg\max_{a \in \mathcal{A}} \widetilde{V}_t(a)$ or sample $a_t \sim \pi(a) \propto e^{\widetilde{V}_t(a)/\tau}$.
6: **(Environment step)** Update (x_{t+1}, v_{t+1}) under a_t; get o_{t+1}; log (a_t, o_t, b_t^+).
7: **end for**

Alg. 1 uses the exact mutual information while Alg. 2 replaces it with the CRN-amenable polynomial upper bound according to which we built our CRN-ODE and make experimental simulations for comparison.

4.2 Biochemical Implementation

In our research, we introduce a CRN-ODE pipeline to execute the one-step Bayesian belief filtering and action scoring and selecting. The choice of this biochemical pipeline is motivated by rational function with non-negative coefficients (RFNC) view of biochemical cascades, where probabilistic computations can be implemented by mass-action networks at steady state, and also by enzymatic motifs of futile cycle that realize reversible regulation with biochemical plausibility using action-related rate parameter. With our design, in each step, we run all the reactions simultaneously within one system, which uses a shared normalization pool to implement divisions, leaky reservoirs (reactions that lead to null set) for value accumulation.

4.2.1 From RFNC to CRN

In the previous section we have given the explicit formula for belief filtering and rewards in form of linear or polynomial functions. Especially to avoid logarithms in the computation of mutual information we have also provided a polynomial upper bound, from which we can design the corresponding CRN.

We have mentioned several studies on the CRN construction as references that supports our design of CRN. Here we are going to dive into details:

[4] Researchers show that all *semilinear functions are deterministically computable* by CRNs, where semilinear function has following definition:

Definition 4.2.1 (Semilinear function)**.** A function $f : \mathbb{N}^\Lambda \to \mathbb{N}$ for a CRN $\mathcal{N} = (\Lambda, R)$

is semilinear if its graph is a finite union of linear sets. Intuitively, semilinear functions are piecewise-affine over regions definable in Presburger arithmetic.

The construction first decomposes each target semilinear function into affine partial piecewise functions with predicate-based selection. Outputs are encoded via monotone positive/negative species so that no chemical subtraction is required under mass action. The following is a state-of-the-art recap of the CRN computability workflow from [4]. We only refer to part of the following workflow in our CRN design.

1. Decomposition: for any semilinear function f, we denote:
$$f(x) = f_i(x), \quad \text{if} \quad x \in dom(f_i), \text{for any affine partial functions} \quad f_i$$
2. parallel computation for every $f_i(x)$: from lemma 4.2 in [4] we build CRN for every $f_i(x)$ with the method from Lemma 4.2 of [4], that outputs candidate species, denoted as $\hat{Y_{i,j}^P}$, and $\hat{Y_{i,j}^C}$ for partial function f_i. Then introduce the predicate outputs a boolean value that determines whether or not we select f_i ($x \in dom(f_i)$).
$$\psi_i = \begin{cases} T_i & \text{if } x \in dom(f_i) \\ F_i & \text{else} \end{cases}$$
3. **Only activate the chosen f_i and terminate the others**
   - **when $\psi_i = T_i$:**
   $$T_i + \hat{Y_{i,j}^P} \to T_i + Y_{i,j}^P + Y_j$$
   $$T_i + \hat{Y_{i,j}^C} \to T_i + Y_{i,j}^C$$
   where Y_j is the actual output of this algorithm, and what we want is $Y_j = Y_{i,j}^P - Y_{i,j}^C$. If f_i is the chosen function, we make $\hat{Y_{i,j}^P}$ effective for $Y_{i,j}^P$ and since $Y_{i,j}^C$ is something we want to clear, in the later steps we will pair up the $Y_{i,j}^P$ and $Y_{i,j}^C$ in conflict check and we will erase the paired-up $Y_{i,j}^C$ from $Y_{i,j}^P$.
   - **when $\psi_i = F_i$ (meaning that we will erase a falsely activated f_i):**
   $$F_i + Y_{i,j}^P \to F_i + M_{i,j}$$
   $$M_{i,j} + Y_j \to \hat{Y_{i,j}^P}$$
   $$F_i + Y_{i,j}^C \to F_i + \hat{Y_{i,j}^C}$$
   where $M_{i,j}$ is an introduced intermediate as an error marker. In the above reactions, we erase the falsely added $Y_{i,j}^P$ from the container for total output Y_j and we return to the inactivated state of $\hat{Y_{i,j}^P}$ and same for the wrongfully activated $Y_{i,j}^C$ that returns to $\hat{Y_{i,j}^C}$.
   - **conflict checks:**
   $$Y_{i,j}^P + Y_{i,j}^C \to K_j$$
   $$K_j + Y_j \to \varnothing$$

where K_j is the conflict marker that indicates there are more than one f_i selected. And the last reaction that produces $\varnothing$ is to clear the total output container Y_j to assume only one f_i is selected to avoid confusion.

This pattern of **monotone-accumulation** +**selective-activation** supports our design of CRN that realizes division/normalization via a shared pool (additional species) and implements **value accumulations** as leaky reservoirs where the steady states of the target species equal the desired scores.

We also largely refer our idea of CRN design to the work of [13] who translate Hidden Markov Model (HMM) inference into biochemical reaction networks (CRNs), by designing a reaction network scheme that simulates the Baum–Welch algorithm using mass-action kinetics, allowing it to converge to the same parameter estimates as the classical EM algorithm. In the original BW algorithm, it includes 4 modules: forward, backward, E-step and M-step. Here we elaborate on the first module with forward algorithm for belief filtering.

Following [13], we consider a tuple $(H, V, \theta, \psi, \pi)$ as HMM, where H is set of hidden states, V is set of visible states, $\theta : H \to H$ is the transition matrix, $\phi : H \to V$ is the emission matrix and π is an initialized prior. And we denote $\pi_h := P(h_1 = h)$, $\theta_{gh} := P(h_{l+1} = h \mid h_l = g)$, $\psi_{hv} := P(v_l = v \mid h_l = h)$. we compute the joint probability of being at state h at time l while seeing $v_1, \ldots, v_l$:

$$\alpha_{1,h} = P(v_1, h_1 = h \mid \theta, \psi) = P(h_1 = h) \cdot P(v_1 \mid h_1 = h) = \pi_h \psi_{h,v_1}$$

from which we can build equivalence to ensure scale-invariance in forward module:

$$\alpha_{1,h}\pi_{h^*}\psi_{h^*,v_1} = \alpha_{1,h^*}\pi_h\psi_{h,v_1} \quad \text{for any} \quad h^* \in H$$

From this equation we can build 2 reactions that indicates the transition between $\alpha_{1,h} \rightleftharpoons \alpha_{1,h^*}$:

$$\alpha_{1,h} + \pi_{h^*} + \psi_{h^*,v_1} \to \alpha_{1,h^*} + \pi_{h^*} + \psi_{h^*,v_1}$$
$$\alpha_{1,h^*} + \pi_h + \psi_{h,v_1} \to \alpha_{1,h} + \pi_h + \psi_{h,v_1}$$

And we can derive the ODE from the above reactions for the target $\alpha_{1,h}$:

$$\frac{d[\alpha_{1,h}]}{dt} = [\alpha_{1,h^*}] \cdot [\pi_h] \cdot [\psi_{h,v_1}] - [\alpha_{1,h}] \cdot [\pi_{h^*}] \cdot [\psi_{h^*,v_1}]$$

and at equilibrium we can get the exact equation implied above.

Same with $l = 1, \ldots, L-1$, we have

$$\alpha_{l+1,h} = P(v_1, \ldots, v_{l+1}, h_{l+1} = h \mid \theta, \psi) = \sum_g \alpha_{l,g}\theta_{g,h}\psi_{h,v_{l+1}} \quad \text{from which we have:}$$

$$\alpha_{l+1,h} \sum_g \alpha_{l,g}\theta_{g,h^*}\psi_{h^*,v_{l+1}} = \alpha_{l+1,h^*} \sum_g \alpha_{l,g}\theta_{g,h}\psi_{h,v_{l+1}} \quad \text{for any} \quad h^* \in H$$

And similarly we can build reactions from transition between $\alpha_{l+1,h} \rightleftharpoons \alpha_{l+1,h^*}$:

$$\alpha_{l+1,h} + \alpha_{l,g} + \theta_{g,h^*} + \psi_{h^*,v_{l+1}} \to \alpha_{1+1,h^*} + \alpha_{l,g} + \theta_{g,h^*} + \psi_{h^*,v_{l+1}}$$
$$\alpha_{1,h^*} + \pi_h + \psi_{h,v_1} \to \alpha_{1,h} + \pi_h + \psi_{h,v_1}$$

we can also build ODE from the above reactions for the target $\alpha_{l+1,h}$:

$$\frac{d[\alpha_{l+1,h}]}{dt} = \sum_g [\alpha_{l+1,h^*}] \cdot [\alpha_{l,g}] \cdot [\theta_{g,h}] \cdot [\psi_{h,v_l}] - \sum_g [\alpha_{l+1,h}] \cdot [\alpha_{l,g}] \cdot [\theta_{g,h^*}] \cdot [\psi_{h^*,v_l}]$$

same as before we can get the implied equation at equilibrium.

The problem here is that since ψ_{h,v_l} depends on v_l, with different values of v_l, we have to design different corresponding reactions. So to simplify the process, we introduce another species $E_{l,w}$ as another catalyzer of the reactions, initialized with:

$$E_{l,w}(0) = \begin{cases} 1 & w = v_l \quad \text{(reaction activated iff} \quad w = v_l) \\ 0 & \text{else} \end{cases}$$

so that the reaction rate of the target will depend on the concentration of $E_{l,w}$. And this species will also function in the other modules. Therefore, we can construct the subnetworks $\mathcal{R}_1^\alpha$ and $\mathcal{R}_l^\alpha$ as follows:

$$\begin{aligned}
\mathcal{R}_1^\alpha: \quad & \alpha_{1,h} + \pi_{h^*} + \psi_{h^*,v_1} + E_{1,w} \to \alpha_{1,h^*} + \pi_{h^*} + \psi_{h^*,v_1} + E_{1,w} \\
& \alpha_{1,h^*} + \pi_h + \psi_{h,v_1} + E_{1,w} \to \alpha_{1,h} + \pi_h + \psi_{h,v_1} + E_{1,w} \\
\mathcal{R}_l^\alpha: \quad & \alpha_{l+1,h} + \alpha_{l,g} + \theta_{g,h^*} + \psi_{h^*,v_{l+1}} + E_{l+1,w} \to \alpha_{1+1,h^*} + \alpha_{l,g} + \theta_{g,h^*} + \psi_{h^*,v_{l+1}} + E_{l+1,w} \\
& \alpha_{1,h^*} + \pi_h + \psi_{h,v_1} + E_{l+1,w} \to \alpha_{1,h} + \pi_h + \psi_{h,v_1} + E_{l+1,w}
\end{aligned}$$

In the following section, we instantiate a one-step forward, memoryless posterior/belief update with 2 pragmatic simplifications and modifications:

- Instead of building a futile cycle between different hidden states, we replace the explicit normalization by a shared pool
- Since the model is much simplified, we keep only posterior update and action scoring for online control without backward nor parameter update for EM, which greatly preserves homomorphism to our numerical pipeline while minimizing network size and scheduling complexity.

Our design combines the monotone-accumulation/selective principle (from semilinear to CRN) with a forward-style scaffold (from BW), to realize normalization and steady-state readouts—thus implementing one-step posterior + action scoring as modular, parallel CRN–ODE blocks.

4.2.2 CRN-ODE construction

We encode the memoryless Bayesian belief update and the exploitation-exploration objective into a chemical reaction network(CRN), and simulate via ODEs.

We build CRN for each action `run` and `tumble`, the two CRNs share the same structure but with different reaction rates. And in our CRNs, we only consider one-next-step update, so we build the CRN of only two consecutive steps. We can split the CRN with 3 parts: one-step belief update, immediate expected reward, curiosity.

One-step belief update

Original formula Given **hidden state** $h = 1, \ldots, n$, **observation pair** (o_1, o_2) and **mapped observation index** $v = o_1 \cdot m + o_2 \in \{1, \ldots, m^2\}$, flattened emission matrix $Z_{\text{flat}} \in \mathbb{R}^{m^2 \times n}$. With prior $b_0(h)$, the posterior (updated belief) $b^+(h)$ is:

$$b^+(h) \;=\; \frac{Z_{\text{flat}}[v,h]\; b_0(h)}{\sum_{j=1}^{n} Z_{\text{flat}}[v,j]\; b_0(j)}$$

Species Here we list all the species needed to compute the one-step belief update:

- $A_h^{(1)}$: stores the prior $b_0(h)$.
- $A_h^{(2)}$: stores the posterior $b^+(h)$.
- U_h: stores the unnormalized product $Z_{\text{flat}}[v,h]b_0(h)$.
- E_v: one-hot observation catalyst that is activated only with observation index v.
- M: a shared normalization pool

Reactions Since the emission matrix Z is fixed in our POMDP, we decided to put it as reaction rate in our following design.

1. **unnormalized likelihood $\times$ prior $U_h \propto Z_{\text{flat}}[v,h]b_0(h)$:** For all h, v:

$$A_h^{(1)} + E_v \xrightarrow{k_{v,h}=Z_{\text{flat}}[v,h]} A_h^{(1)} + U_h + E_v$$

This is a catalytic reaction with only U_h accumulating while keeping $A_h^{(1)}, E_v$ unchanged.

Only when E_v is activated, we have:

$$\frac{d}{dt}U_h = Z_{\text{flat}}[v,h]A_h^{(1)}E_v$$

2. **Normalization using shared pool M:** For all h:

$$M + U_h \xrightarrow{1} A_h^{(2)} + U_h$$

Here U_h appear only catalytically,which is not consumed and no further production occurs. And the above reaction realizes normalization because:

$$\dot{M} = \frac{d}{dt}M = -M\sum_j U_j = -\mu M \qquad \text{where } \mu = \sum_j U_j$$

$$\frac{d}{dt}A_h^{(2)} = MU_h = M\frac{U_h}{\sum_j U_j}\sum_j U_j = -\frac{U_h}{\mu}\dot{M}$$

If integrate from 0 to ∞ with $A_h^{(2)}(0) = 0, M(0) = 1, M(\infty) = 0$:

$$\int \frac{d}{dt}A_h^{(2)}dt = \int MU_h dt = \int -\frac{U_h}{\mu}\dot{M}dt = \frac{U_h}{\mu}(1-e^{-\mu t}) \xrightarrow{t\to\infty} \frac{U_h}{\mu} = \frac{Z_{\text{flat}}[v,h]b_0(h)}{\sum_j Z_{\text{flat}}[v,j]b_0(j)} = b^+(h)$$

Immediate expected reward

Original Formula We can recall the immediate expected reward is computed as:

$$V^{(1)}(a) = r^\top T_a b^+$$

where r is a constant vector, and T_a is fixed.

Species Since we already have A to store the beliefs, the only species we add here is R that stores the immediate expected reward.

Reactions

$$A_h^{(2)} \xrightarrow{w_a(h)=r^\top T_a} A_h^{(2)} + R, \qquad R \xrightarrow{k_R} \varnothing$$

And we have the ODE for R:

$$\begin{aligned}
\frac{d}{dt}R &= \sum_h w_a(h) A_h^{(2)}(t) - k_R R \\
&= \sum_h w_a(h) \frac{U_h}{\mu}(1 - e^{-\mu t}) - k_R R \\
&= \sum_h w_a(h) A_h^{(2)}(\infty)(1 - e^{-\mu t}) - k_R R \\
&= S(\infty)(1 - e^{-\mu t}) - k_R R \qquad \text{where} \quad S(t) = w_a(h) A_h^{(2)}(t)
\end{aligned}$$

or we can also write it as:

$$\dot{R} + k_R R = S(t) = S(\infty)(1 - e^{-\mu t})$$

To get the closed form on $R(t)$ explicitly, we first have:

$$\begin{aligned}
\frac{d}{dt}(e^{k_R t} R(t)) &= k_R e^{k_R t} R(t) + e^{k_R t} \dot{R}(t) \\
&= e^{k_R t}(k_R R + \dot{R}) \\
&= e^{k_R t} S(t)
\end{aligned}$$

$$\begin{aligned}
e^{k_R t}(R(t) - R(0)) &= \int e^{k_R \tau} S(\tau) d\tau \\
&= S(\infty) \int_0^t e^{k_R \tau}(1 - e^{-\mu \tau}) d\tau \\
&= S(\infty)\left(\frac{1}{k_R}(e^{k_R t} - 1) - \frac{1}{k_R - \mu}(e^{(k_R - \mu)t} - 1)\right)
\end{aligned}$$

$$\begin{aligned}
R(t) &= e^{-k_R t}\left(R(0) + \int e^{k_R \tau} S(\tau) d\tau\right) \\
&= e^{-k_R t}\left(R(0) + \frac{S(\infty)}{k_R}(1 - e^{-k_R t}) - \frac{S(\infty)}{k_R - \mu}(e^{-\mu t} - e^{-k_R t})\right) \\
&\xrightarrow{t \to \infty} \frac{S(\infty)}{k_R} = \frac{\sum_h w_a(h) b^+(h)}{k_R} \qquad \text{where} \qquad R(0) = 0.
\end{aligned}$$

So when the ODE system reaches equilibrium, we can obtain the **immediate expected reward** with $k_R = 1$:

$$R(\infty) = \sum_h w_a(h) b^+(h) = r^\top T_a b^+$$

Curiosity

Original formula Since we cannot compute $f(x) = \log x$ directly with CRN, so we build the CRN for the curiosity part with its polynomial upper bound:

$$V^{(2)}_{\text{bound}}(a) = \sum_{o_1,o_2,h} b'_a(h) Z^2_{o_1,o_2|h} + m^2 - 2 = \sum_{o_1,o_2,h} b'_a(h) Z^2_{o_1,o_2|h} + \text{constant}$$

And we only use CRN to encode the first part:

$$C(a) = \sum_h T_a b^+(h) \sum_v Z_{\text{flat}}[v,h]^2$$

Species Similar to the previous part, we only need to add another species C_h to accumulate curiosity from state h.

Reactions For every h:

$$A^{(2)}_h \xrightarrow{c_h = T_a \sum_v Z[v,h]^2} A^{(2)}_h + C_h, \qquad C_h \xrightarrow{k_C} \varnothing$$

And similarly we have ODE for C_h:

$$\frac{d}{dt} C_h = c_h A^{(2)}_h - k_C C_h$$

When the ODE system reaches equilibrium with $C_h(0) = 0, k_C = 1$we have :

$$\sum_h C_h(\infty) = \sum_h c_h b^+(h)$$

The final curiosity bound is obtained with: $\sum_h C_h(\infty) + m^2 - 2$

CRN-ODE policy

In the CRN-ODE policy, we run separately the CRN-ODE built for two actions and extract respectively the exploitation(immediate expected reward) and exploration (curiosity) rewards.

Same as the numerical policy before, the total reward and the optimal action are obtained by:

$$V_{CRN-ODE}(a) = R(a) + \lambda C(a), \qquad a^* = \arg\max_a V_{CRN-ODE}(a)$$

where λ is the balancing weight for exploitation and exploration terms.

4.3 Experiments and Results

In our experiments, we first evaluate the proposed POMDP framework that we have designed above for algae phototaxis behavior in single- and multi-source scenarios. For all scenarios, we simulated trajectories and computed the statistics on the ratio of tumbles.

In the setup of first two experiments, we simulate a circular agent with two eye-spots in 2D. Unless stated otherwise, we fixed the following configuration:

- **Geometry & FoV.** Radius $R = 0.2$, eye offsets $\pm\gamma = \pm 30^\circ$, optical axes $\pm\alpha = \pm 20^\circ$, per-eye half-FoV FoV $= 60^\circ$.
- **Light source configuration.** center at `[100,100]`,peak intensity at center $I_0 = 1.0$, Gaussian field parameter $\sigma = 100.0$, distance attenuation decay at $\beta_d = 0.01$.
- **Sensor saturation.** $S(z) = (1 + \exp[-\beta_s(z - I_{50})])^{-1}$ with $\beta_s = 5.0$,$I_{\text{thr}} = 0.005$, threshold `thr` $= 0.5$;
- **POMDP components.** $n = 5$, $m = 2$, actions $\{0 = \text{run}, 1 = \text{tumble}\}$, and the exploitation reward vector $r = (10, 8, 0, 0, 8)$.

We put the initial position of the agent at `[0,0]`, with the heading vector of `[1.0,0.0]`. The agent executes a one-step look-ahead policy as derived in Chapter 3. For reproducibility, we explicitly document where **new hyper-parameters** (curiosity weight λ) start to differ from the preliminary runs. And with our first two experiments with simulations, we use the same stride for both actions at `0.125`, and we add same amount of random noise at uniform distribution for both actions.

4.3.1 Simulation on single-source

Numerical POMDP with exact MI

Following the Algo.1 mentioned above, we fix the light source center at `[100.0,100.0]` for single-source scenario. And we randomly choose a hyperparameter of curiosity weight $\lambda = 0.3$:

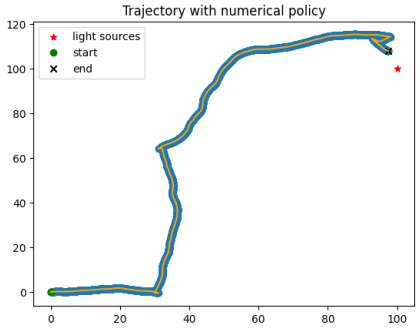


Figure 4.1: Trajectory simulated with numerical POMDP pipeline with exact mutual information, hyperparameter curiosity weight $\lambda = 0.3$

With `4000` steps, the agent tumbles for only `9` steps, where the tumble ratio is at `0.00225`.

We match exploitation and exploration scales using empirical means over a length-T trajectory:

$$\lambda^* = \frac{\hat{\mathbb{E}}[V^{(1)}]}{\hat{\mathbb{E}}[V^{(2)}]}, \quad \hat{\mathbb{E}}[V^{(1)}] = \frac{1}{T}\sum_{t=1}^{T} V_t^{(1)}, \quad \hat{\mathbb{E}}[V^{(2)}] = \frac{1}{T}\sum_{t=2}^{T} V_t^{(2)}$$

We use the $\lambda^* =$ `(6.8319716)` for the following experiments on single-source. And with the tuned hyperparameter,we plot again the simulated trajectory of numerical POMDP pipeline with exact MI:

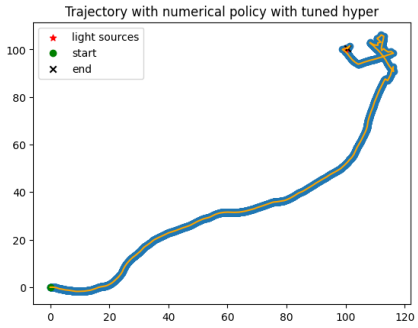


Figure 4.2: Trajectory simulated with numerical POMDP pipeline with exact mutual information, coarsely-tuned hyperparameter curiosity weight $\lambda = 6.83$

After the coarse scaling of λ, the curiosity/ exploration term is weighted more than before, the agent also tumbles slightly more at tumble ratio `0.00725` (29"tumbles" out of 4000 steps.) And by balancing and putting more weight on exploration,though there are more "swirling" but the agent still moves towards the light. We continue the following experiments with this tuned λ^*.

Numerical POMDP with curiosity upper bound

As we have proposed above with Algo.2, we simulate the trajectory under numerical POMDP policy with polynomial upper bound for curiosity:

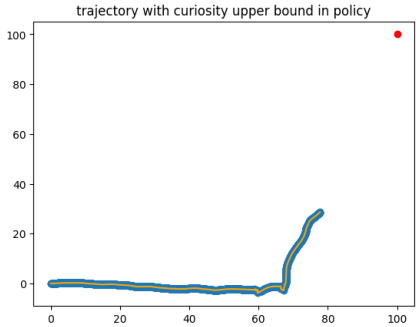


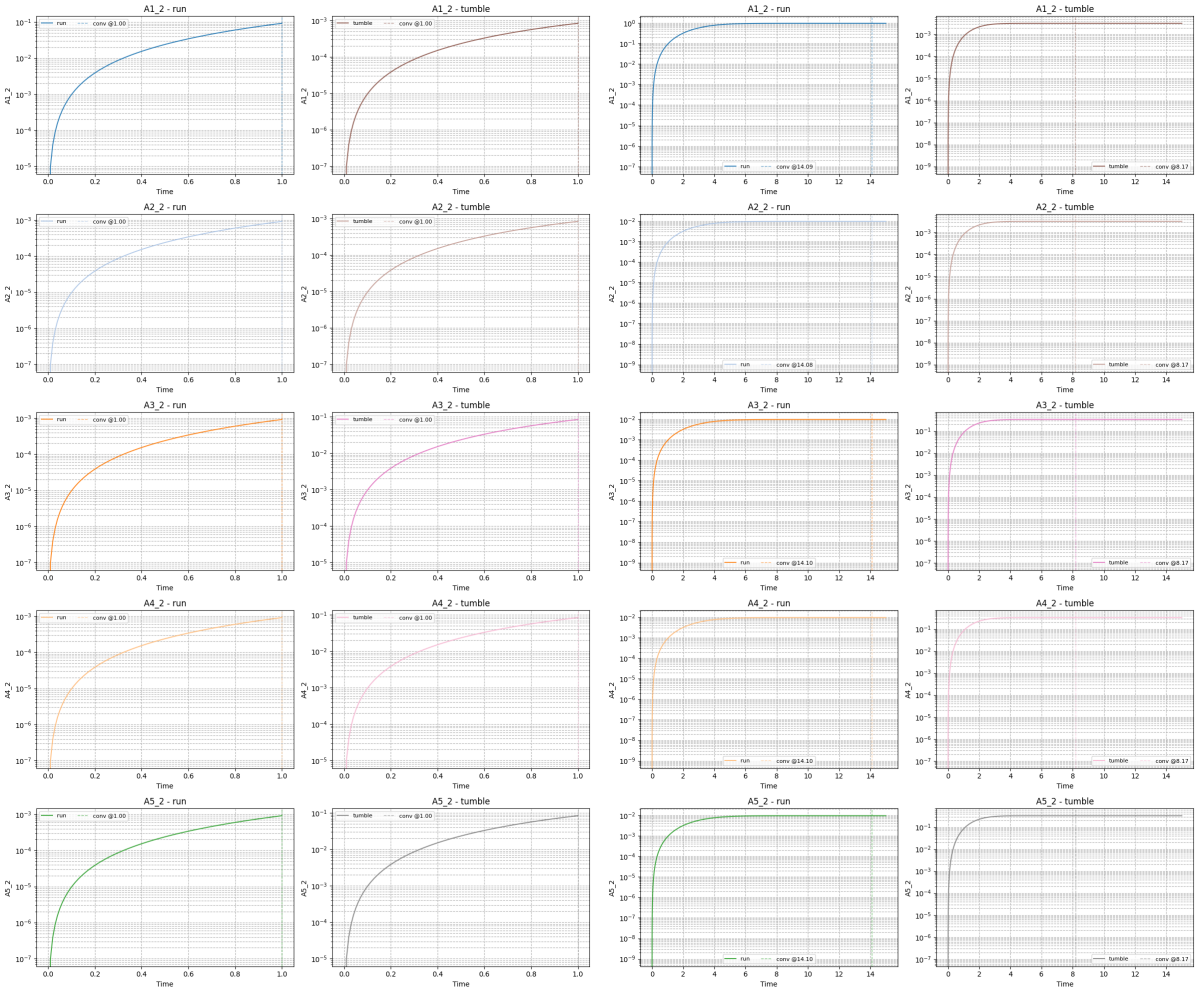
Figure 4.3: Trajectory simulated with numerical POMDP pipeline with curiosity upper bound, coarsely-tuned hyperparameter curiosity weight $\lambda = 6.83$

And in this plot, there are `7` tumbling events out of total steps of `2000`, making tumbling ratio at `0.0035`. Though with less tumbles under less total steps of simulation, we can still see the agent is indeed moving towards the light source.

CRN-ODE simulation

While all above simulations with numerical policy only take 1 second for 2000 steps , for CRN-ODE simulation it takes much longer time: more than 1.5 hour for 2000 steps. This is because, even with the fixed structure of CRN-ODE, during each step we need to make sure the ODE system reaches its steady state, and we need to re-initialize the system(species' concentration) according to the obtained observation, even with memoryless belief update.

Here we plot out the convergence of the target value —concentration of species A_h^2:



(a) Convergence with $t_8 = 1.0\,\mathrm{s}$ (ten panels: $A1\_2$–$A5\_2$ for run/tumble).

(b) Convergence with $t_8 = 15.0s$ (same species/actions as above).

Figure 4.4: Effect of the integration horizon t_8 on CRN–ODE convergence. Log-y scale; dashed vertical lines denote the stopping time in each panel.

In the above figure, each panel shows log-scaled concentrations of representative species A_i under two actions (run/tumble). The vertical dashed line marks the stopping time. Here we note t_8 the integration horizon per step used to approximate steady state. Smaller t_8 speeds up simulation but may truncate before full steady state; larger t_8 improves accuracy at higher cost.

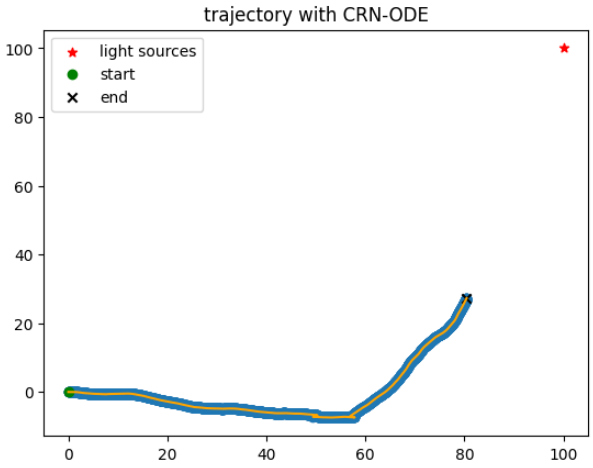


Figure 4.5: Trajectory simulated with CRN-ODE, coarsely-tuned hyperparameter curiosity weight $\lambda = 6.83$

Though with much more computational time, CRN-ODE achieves tumbling ratio at `0.002` with `4` tumbles over 2000 steps, and the plot shows a similar behavioral pattern as the previous numerical simulation with curiosity upper bound.

Having matched patterns, we now calibrate the exploration–exploitation balance quantitatively with curiosity weight λ that also controls tumbling ratio and alignment statistics.

Hyperparameter search on λ

We also do a further fine-tune grid search on hyperparameter of curiosity weight λ that balances off exploitation and exploration term in policy value, following the numerical policy with curiosity upper bound. We put λ on the fixed grid of $[0, 2]$, with 10 evenly spaced points. And because of the noise added in each step of action, the plots demonstrate randomness though following visually-similar pattern. So for each choice of λ, we generate 100 trajectories, and we log:

1. **progress:** Normalized reduction in distance to the target from start to end; higher is better:
$$\text{prog} = (\|x_0 - x_l\| - \|x_T - x_l\|)/(\|x_0 - x_l\| + \varepsilon)$$
2. **tumble ratio:** As the statistics we have compute before, the fraction of taking the tumble action
3. **total reward:** the total objective V_{total}

4. **Balance index:** Ratio of average magnitudes

Selection follows our original rule: first keep $\Lambda_\tau = \{\lambda : \rho(\lambda) \in [1/\tau, \tau]\}$ and within it maximize the mean V_{total}; if $\Lambda_\tau = \varnothing$, choose the λ minimizing $|\log \rho(\lambda)|$, breaking ties by larger mean V_{total}.

All the trials and the whole search took 2.5 hours to run:

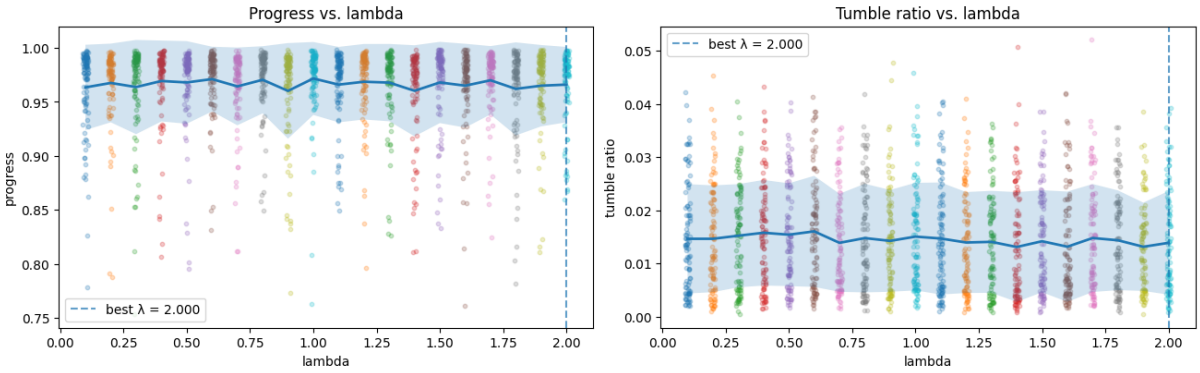


Figure 4.6: Fixed-grid fine-tuning of hyperparameter: curiosity weightλ over 20 candidates and 100 trials each

The resulting $\lambda^\star = 2.0$ is then used in the following multi-source experiments.

And we also plot a trajectory with numerical policy with curiosity upper bound

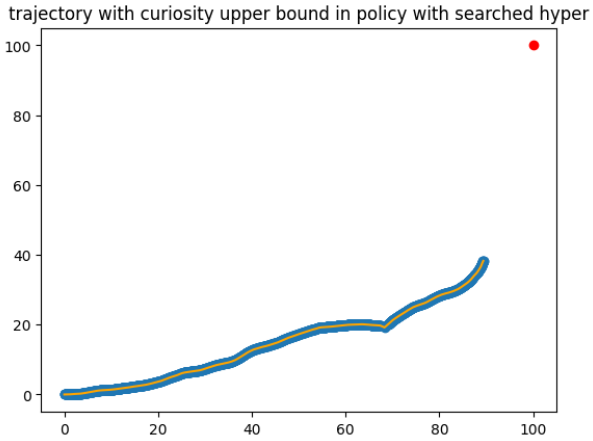
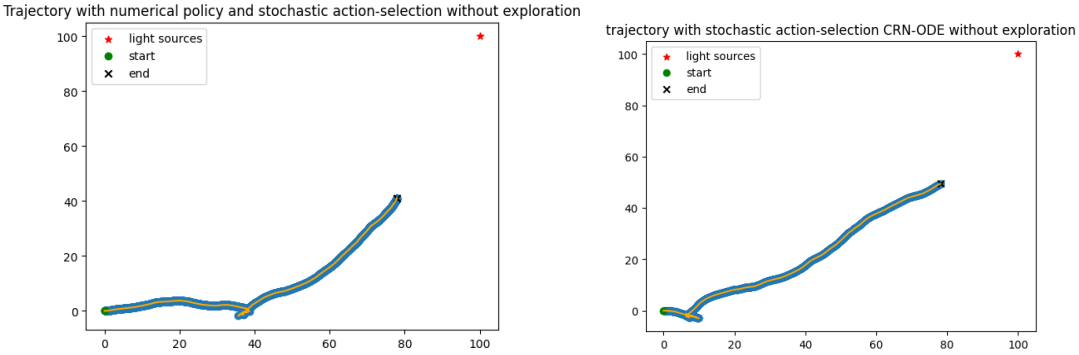


Figure 4.7: Trajectory simulated with numerical POMDP pipeline with curiosity upper bound, fine-tuned hyperparameter curiosity weight $\lambda = 2.0$,at tumble ratio of `0.001`

Simulation with stochastic action-selection without exploration

The above simulations are done based on deterministic action-selection, however, To match the soft policy used in the next chapter, we replace deterministic arg max by a stochastic soft policy while turning off exploration. Here we have simulations on single-source scenario for both numerical and biochemical POMDP implementation with stochastic action-selection ($a_t \sim \pi(a) \propto e^{V_t(a)/\tau}$) and without exploration ($\lambda = 0$).



(a) Numerical implementation, soft policy, $\tau = 0.1$

(b) CRN-ODE implementatio, soft policy, $\tau = 0.1$

Figure 4.8: Single-source with stochastic action-selection and no exploration $\lambda = 0$.

On the single-source scenario, both numerical and CRN-ODE trajectories progress toward the light center; compared to deterministic control, stochastic selection does not change the global direction, with tumble ratio at `0.0035` (7 tumbles out of 2000 steps) for numerical implementation (same as deterministic action-selection) and `0.005` (10 tumbles out of 2000 steps) for CRN-ODE implementation. But we observe that stochastic selection adds local fluctuation where tumble events clusters whereas the deterministic policy shows more evenly spread tumbles across trajectory.

4.3.2 Simulation on multi-source

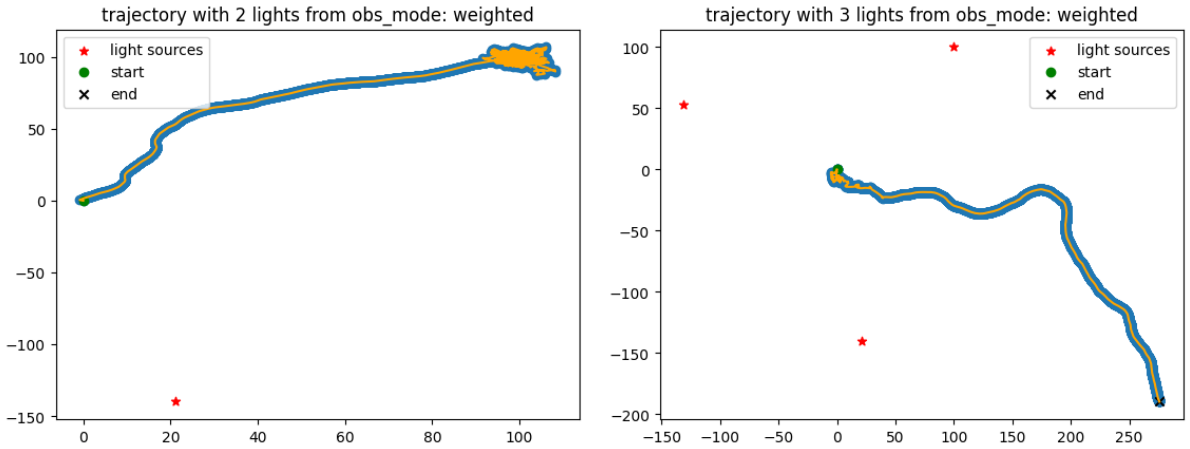
In the multi-source scenarios, we randomly add other lights on a circle with center at `(0,0)` and radius at $100\sqrt{2}$, so that the new lights are on the same circle as the previously-fixed light on `(100,100)`. Here we return to the deterministic action-selection.

And as CRN-ODE policy is too time-consuming, we continue the multi-source experiment on numerical policy with curiosity upper bound. And set more steps for the agent to navigate with total of `8000` steps. And it takes 8 seconds for each of the following simulation. And in each method, we discuss the difference between 2-lights and 3-lights cases.

Weighted-average

Here we primarily report weighted-observation results, as they are grounded biologically. The experiment results for alternative multi-source observation aggregator is presented in the appendix B.

We recall that with weighted-average method, we aggregate post-saturation channels $y_c = \frac{\sum_k w_k y_c^k}{\sum_k w_k}, \quad c \in \{L, R\}$ via distance-attenuated weights $w_k \propto a(s^k)$.



(a) 2 lights - weighted average

(b) 3 lights - weighted average

Figure 4.9: Multi-source with weighted average observation

When there are 2 lights, we can observe that the weighted-average method first balances the lights' perception, then leads the agent to drift toward the nearer/stronger light. While when the agent is situated in the middle of 3 lights, the agent with weighted-average observation follows the resultant synthetic vector and moves toward a weighted centroid, not biasing one particular source.

5. Policy learning

In the previous simulations, we have experimented with a fixed unified POMDP and with fixed design of the policy, where the reward vector r in exploitation is a given constant.

The above simulations establish feasibility for single- & multi- source scenarios , but biological validity requires policy learning from real trajectories in single-source experiments. Our goals is to train the policy $\pi(a|o)$ and combine with inverse reinforcement learning (IRL) to infer rewards and obtain optimal policy to match empirical real-data statistics.

5.1 Data-preprocessing

The experiment where done using the microalgae Chlamydomonas Reinhardtii under a microscope. We used an optical fiber with blue light source to trigger phototactic behavior. The imaging light had a red filter to avoid interference with the light triggering phototaxis. The algae were swimming in a 1mm depth cavity made of laser cut plastic (PMMA). We have raw data of 30 trajectories and the corresponding time stamps captured by cameras.

Mapping DTI to frame-level time intervals Let the movie frames span integers $(f \in f_{\min}, \ldots, f_{\max})$. The DTI file provides **inter-frame intervals**(Δt_k) for $(k = 0, \ldots, L-1)$, where

$$L = f_{\max} - f_{\min}, \qquad \Delta t_k \text{ corresponds to } (f_{\min} + k) \to (f_{\min} + k + 1).$$

We build a **global absolute time axis** by cumulative sum:

$$T(0) = 0, \qquad T(k) = \sum_{j=0}^{k-1} \Delta t_j \quad (k \geq 1).$$

For any frame f, the absolute time is looked up by

$$t_{\text{abs}}(f) = T\big(f - f_{\min}\big).$$

This automatically handles **missing frames**: if a trajectory jumps from f_0 to $f_I > f_0 + 1$, then $(t_{\text{abs}}(f_I) - t_{\text{abs}}(f_0) = \sum_{k=f_0}^{f_I-1} \Delta t_k)$.

5.1.1 Per-trajectory preprocessing(time, velocity and heading)

For a trajectory c, let its ordered frames be $\{f_i^c\}_{i=0}^{n_c-1}$ with positions (x_i^c, y_i^c).

Relative time(per trajectory) :

$$t_i^c = t_{\text{abs}}(f_i^c); -; t_{\text{abs}}(f_0^c).$$

Time step :

$$\Delta t_i^c = \begin{cases} 0, & i = 0, \\ t_{\text{abs}}(f_i^c) - t_{\text{abs}}(f_{i-1}^c), & i \geq 1. \end{cases}$$

Displacements :

$$\Delta x_i^c = x_i^c - x_{i-1}^c, \qquad \Delta y_i^c = y_i^c - y_{i-1}^c \quad (i \geq 1).$$

Velocities :

$$v_{x,i}^c = \frac{\Delta x_i^c}{\Delta t_i^c}, \qquad v_{y,i}^c = \frac{\Delta y_i^c}{\Delta t_i^c}, \qquad |v_i^c| = \sqrt{(v_{x,i}^c)^2 + (v_{y,i}^c)^2}.$$

Heading angle & wrapped angular speed :

$$\theta_i^c = \text{atan2}(\Delta y_i^c, \Delta x_i^c), \qquad \omega_i^c = \frac{\text{wrap}(\theta_i^c - \theta_{i-1}^c)}{\Delta t_i^c} \quad (i \geq 2),$$

where $\text{wrap}(\alpha) = (\alpha + \pi) \bmod 2\pi - \pi$ enforces $(-\pi, \pi]$.

5.1.2 Forward-aligned interval features

For each trajectory, row i represents a frame f_i, while action applies on the **interval**

$$\mathcal{I}_i = [t_i, , t_{i+1}) \quad (i = 0, \ldots, n_c - 2).$$

We therefore **forward-align** the interval features to the next row:

$$v_i^{\text{fwd}} \equiv |v_{i+1}|, \qquad \omega_i^{\text{fwd}} \equiv \omega_{i+1}.$$

5.1.3 Run/Tumble labeling rule

We assign a discrete action $a_i \in \{0 = \text{run}, 1 = \text{tumble}\}$ to each interval $\mathcal{I}_i$ using two thresholds:

1. **Low-speed threshold (global robust estimate from all intervals):**

$$\varepsilon_v = \text{Quantile}_q\left(v_i^{\text{fwd}}\right), \quad q \in [0, 1] \text{ (e.g. } q = 0.05).$$

2. **High-turn threshold from a 2-means clustering on $|\omega^{\text{fwd}}|$:** Fit KMeans with $K = 2$ to $\{|\omega_i^{\text{fwd}}|\}$, obtain centers $\mu_{\text{lo}} \leq \mu_{\text{hi}}$, and set

$$\tau_\omega = \frac{\mu_{\text{lo}} + \mu_{\text{hi}}}{2}.$$

Decision rule (per interval $\mathcal{I}_i$):

$$a_i = \begin{cases} 1 & \text{if } v_i^{\text{fwd}} \leq \varepsilon_v \wedge |\omega_i^{\text{fwd}}| > \tau_\omega \quad \text{(near-stationary \& large turn)}, \\ 0 & \text{otherwise (run)}. \end{cases}$$

If ω_i^{fwd} is invalid (`NaN/inf`), we fall back to the speed criterion:

$$a_i = \mathbb{1}\{v_i^{\text{fwd}} \leq \varepsilon_v\}.$$

5.1.4 Mapping to uniform time interval

Since the timestamps are not captured uniformly, we resample the irregular trajectories onto a uniform step Δt via linear interpolation to correspond with our POMDP framework in order to train the policy accordingly.

Let $\bar{\tau}_{\text{tum}}$ be the mean tumble dwell on raw data. Choose `steps per tum` $s = 3$:

$$\Delta t_{\text{policy}} = k\Delta t_{\text{frame}}, \quad \text{with } k = \max(1, \text{round}(\frac{\bar{\tau}_{\text{tum}}}{s\Delta t_{\text{frame}}}))$$

And interpolate each trajectory at $t^* = t_0 + m\Delta t_{\text{policy}}$:

$$\mathbf{x}(t^*) = (1-\alpha)\mathbf{x}(t_0) + \alpha\mathbf{x}(t_1), \quad \alpha = \frac{t^* - t_0}{t_1 - t_0}.$$

After projecting the trajectories to uniform grid, we compute overlaps between uniform grid window $[T_0, T_1)$ (at length `dt_policy`) and the original interval $[t_i, t_{i+1})$ with:

$$\text{overlap}_i = \max(0, \min(T_1, t_{i+1}) - \max(T_0, t_i)).$$

And we map the action labels to uniform grid by using **tumble-time fraction** ϕ:

$$\phi = \frac{\sum_i \text{overlap}_i \mathbf{1}[a_i = 1]}{T_1 - T_0}, \qquad a_{\text{new}} = \mathbf{1}[\phi \geq 0.5]$$

This projection method can effectively avoid interpolation-induced drift. And each sample becomes:

$$(o_t.\Delta t, x_0, v_0, x_1, v_1, o_{t+1}, a_{\text{new}})$$

5.2 Learning POMDP policy

We designed 2 phases for the policy-learning:

1. joint learning of observation policy and fitting kinematics and POMDP kernels
2. Inverse Reinforcement Learning (IRL) for exploitation reward

And we compare the simulation results from the learned POMDP policy with 2 versions of benchmark SSA algorithms and statistics of the real data. At time t, we input $(o_t.\Delta t, x_0, v_0, x_1, v_1, o_{t+1}, a_{\text{new}})$, and the action-label a_t is only used for evaluation (tumbling ratio) but not for supervision.

5.2.1 Phase 1: Policy learning

Hazard Policy: A 2-layer MLP maps one-hot observation $O \in \{0,1\}^{m^2}$ (see footnote [1])to a hazard rate $h_\theta(o)$:

$$h_\theta(o) = \text{softplus}(W_2 \tanh(W_1 O + b_1) + b_2) \geq 0$$

[1]biophysical intensities (y_L, y_R) are saturated and thresholded into m bins per eye, yielding $m \times m$ discrete bins encoded as one-hot vectors. In our experiments m=2 ⇒ d=4. Here we initialize linear layers with Xavier-uniform and zero biases, and apply weight decay (10^{-4}) to improve stability on the small one-hot input. The implemented 2-layer MLP on one-hot input behaves like a per-bin scalar lookup. It can be replaced with a tabular parameterization with fewer parameters.

And the one-step tumble probability on window Δt (policy to be learned):

$$p_\theta(a = 1 \mid o, \Delta t) = 1 - \exp(-h_\theta(o)\Delta t).$$

This hazard policy design is initially for cases when windows have different Δt. Since we have build the samples on uniform grid, it works the same as policy at Bernoulli distribution. The MLP for hazard policy has following parameters: the input dimension $d = m^2$, hidden width H. $W_1 \in \mathbb{R}^{H\times d}$, $W_2 \in \mathbb{R}^{1\times H}$, $b_1 \in \mathbb{R}^H$, $b_2 \in \mathbb{R}$. Total parameter count is $H(d + 2) + 1$.

Kinematics In the original algorithm, the choice of stride for different actions matters for simulation. when the agent `runs`, it keeps its current heading $\hat{u}_t = \dfrac{v_t}{|v_t| + \varepsilon}$ and moves with speed c_{run}:

$$\hat{x}^{\text{run}}_{t+1} = x_t + \Delta t_t \, c_{\text{run}} \, \hat{u}_t, \qquad \hat{v}^{\text{run}}_{t+1} = c_{\text{run}} \, \hat{u}_t.$$

when it `tumbles`, its directional heading is (approximately) isotropic and the speed scale is c_{tum} (typically $c_{\text{tum}} \le c_{\text{run}}$):

$$\hat{v}^{\text{tum}}_{t+1} \sim c_{\text{tum}}(\cos\theta, \sin\theta), \qquad \theta \sim \text{Unif}[0, 2\pi).$$

So we have the kinematic reconstruction loss as part of training target:

$$\ell_{\text{run}} = \alpha_x \|\hat{x}^{\text{tum}}_{t+1} - x_{t+1}\|^2 + \alpha_v \|\hat{v}^{\text{tum}}_{t+1} - v_{t+1}\|^2,$$

$$\ell_{\text{tum}} = \alpha_x \mathbb{E}\|\hat{x}^{\text{tum}}_{t+1} - x_{t+1}\|^2 + \alpha_v \mathbb{E}\|\hat{v}^{\text{tum}}_{t+1} - v_{t+1}\|^2,$$

where for `tumble`, we sample K candidates using Monte-Carlo approximation. And the total reconstruction loss is:

$$L_{\text{kin}} = \mathbb{E}_t(\Delta t \cdot \{(1 - p_\theta)\ell_{\text{run}} + p_\theta \ell_{\text{tum}}\})$$

This reconstruction loss mixes action kinematic prediction errors with policy's action probabilities, thus training the policy and kinematic scales jointly.

Observation likelihood As we have given in the design of POMDP, the memoryless belief update follows:

$$b_t(h) \propto Z(o \mid h), \qquad q(o_{t+1} \mid o_t) = \sum_t Z(o_{t+1} \mid h) b_t(h)$$

$$L_{\text{obs}} = -\frac{1}{B} \sum \log q(o_{t+1} \mid o_t), \qquad B = \texttt{batch size}$$

Learnable POMDP kernel In our original design of POMDP, the emission kernel Z is fixed. Here we learn it from data so that observation-statistics are aligned using memoryless update.

To ensure valid probabilities and stable gradients, we parameterize in logit space with softmax normalization:

$$Z(\cdot \mid h) = \text{softmax}(\phi_h) \in \Delta^{m^2-1}, \quad \phi \in \mathbb{R}^{n \times m^2}$$

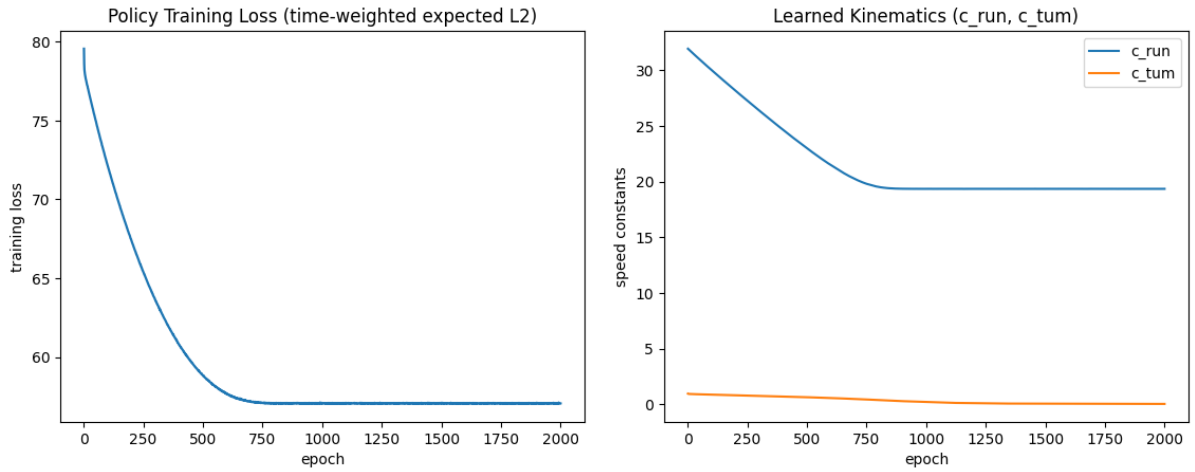
We optimize ϕ and obtain valid Z via softmax in the forward pass. We add a regularization term to the training target:

$$L_{\text{reg}} = \alpha_Z \text{KL}(Z||Z_0)$$

where Z_0 is prior from original design of POMDP.

The phase-1 objective is:

$$L_{\text{Phase1}} = L_{\text{kin}} + \beta_{\text{obs}} L_{\text{obs}} + \alpha_Z \, \text{KL}(Z \| Z_0).$$



(a) Phase 1 - Policy training loss

(b) Phase 1 - learnable kinematics

Figure 5.1: Phase 1 Policy training

We train the policy with a 2-layer MLP for policy (hidden =32) . We use Adam optimizers with learning rates: $\text{lr}_\pi = 3 \times 10^{-3} \text{lr}_{\text{kin}} = 10^{-3}$, $\text{lr}_Z = 10^{-3}$, batch size 1024, 2000 epochs, $\alpha_x = \alpha_v = 1.0, \beta_{\text{obs}} = 0.2, \alpha_Z = 10^{-3}, K_{\text{tum}} = 32$. The training monotonically decreased and converged to around 57. c_{run} converged to ~ 19.3 and c_{tum} to a small value , consistent with run » tumble speeds. These learned parameters are then frozen and used by inverse reinforcement learning in the following section to recover interpretable reward.

5.2.2 Phase 2: Inverse Reinforcement Learning (IRL)

In phase 1, we trained the responsive policy that maps observation to p_{tum}, jointly with physically-grounded kinematics and emission kernels. However it does not tell why the agent prefers certain observations—i.e., the latent objective. Here, IRL recovers the reward vector r to reproduce real-data (expert) behavioral pattern with fixed learned parameters.

Recall the state-action value in our POMDP policy with exploitation and exploration term:

$$Q_r(o,a) = r^\top (T_a b^+) + \lambda_{\text{cur}} MI, \quad b^+ = \frac{Z(o\,|:)}{Z(o\,|:)b_0}$$

Here we set $\lambda_{\text{cur}} = 0.0$ as we do not consider the exploration's influence here. And in IRL, we recover a soft policy with softmax over Q at temperature hyperparameter τ:

$$\pi_r(a \mid o) = \frac{\exp(Q_r(o,a)/\tau)}{\sum_a \exp(Q_r(o,a)/\tau)}.$$

We also compute the expert distribution from real-data statistics with laplacian smoothing:

$$P_{\text{exp}}(a=1 \mid o) = \frac{N_{\text{tum}}(o)+1}{N_{\text{run}}(o)+N_{\text{tum}}(o)+2}, \qquad N_{\text{action}}(o) = \sum_t \mathbf{1}\{o_t = o\}\mathbf{1}\{a_t = \text{action}\}.$$

The IRL objective is:

$$\min_r \mathbb{E}[-\sum_a P_{\text{exp}}(a \mid o) \log \pi_r(a \mid o)] + \lambda_r ||r||_2^2.$$

In IRL, we use $\tau = 1.0, \lambda_r = 10^{-3}$, Adam optimizer with learning rate at 5×10^{-2} at 1000 epochs, and initial reward vector at `[10,8,0,0,8]`. The learned reward vector we get is $r^\star =$ `[ 0.07092752, -0.28374606, 0.07092735, 0.07092735, 0.07092707]` . After recovering a policy from the learned $r^\star$, we simulate one trajectory:

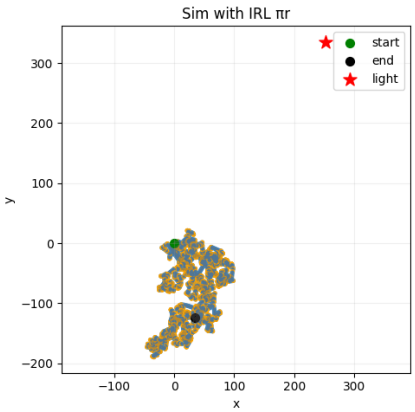


Figure 5.2: Trajectory simulated with policy recovered from reward vector $r^\star$ after IRL

In the simulated trajectory, there are `1940` tumbles out of total of `4000` steps. So we plot out and compare the tumble ratio from real-data statistics and the simulated trajectory:

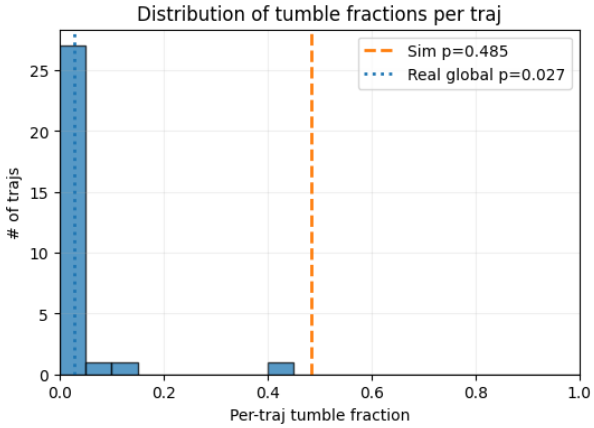


Figure 5.3: Comparison of tumble ratio between POMDP policy retrieved from learned reward of IRL and real data, with histogram of tumble ratio computed from real data.

We can see that there is a large gap of tumble ratio between the retrieved POMDP policy (tumble ratio at `0.485`) and the global mean of the real data (tumble ratio at `0.027`). Therefore, tumble ratio cannot be the only standard to evaluate the effectiveness of the POMDP policy learning.

We implement another standard with **alignment to light source center**. For each step define the alignment cosine:

$$\cos\theta_{\text{align}} = \frac{v_t(\mu - x_t)}{||v_t||||\mu - x_t||}$$

where μ is the light source center position. And first we plot out the overlaid histogram for the distribution of alignment:

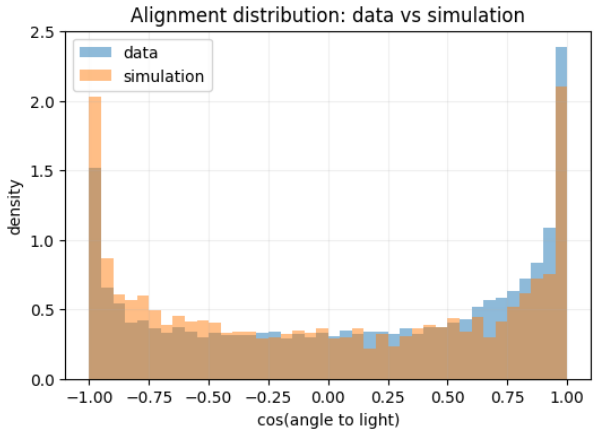
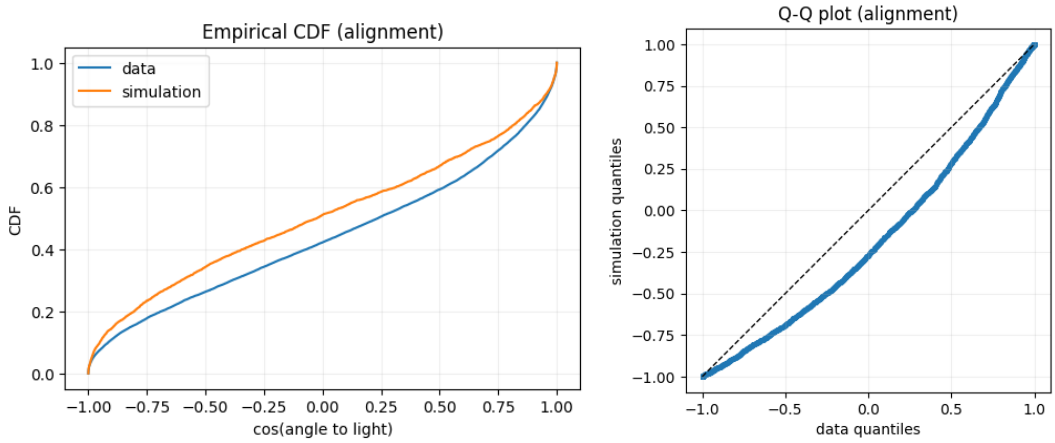


Figure 5.4: Overlaid histogram alignment densities between real data and simulation from learned POMDP

Despite an overestimated tumble frequency at current temperature τ, the alignment distribution shapes between real data and simulation are reasonably close. For more

explicit analysis, we also visualize the empirical cumulative distribution functions (CDFs) and quantile-quantile (Q-Q plot).



(a) empirical CDFs (b) Q-Q plot

Figure 5.5: Alignment analysis between real data and simulation from learned POMDP policy

From the above figures, the orange (simulation) CDF lies above the blue (data) curve in the middle range meaning that the simulated $\cos\theta$ values are stochastically smaller —i.e.,the agent is less aligned with the light than in the data. And from the Q-Q plot, the the simulation's quantiles are smaller than the data's at the same probability level—again indicating slightly weaker alignment in the simulation.

However, the above results are based on only one simlated trajectory from learned POMDP, which might be greatly affected by the behavioral randomness. In the following section, we generate 30 trajectories from the learned POMDP and compare with real data as well as 2 versions of the benchmark SSA algorithm.

5.2.3 Comparison with benchmark: Gillespie algorithm

Gillespie/SSA algorithm is a standard algorithm that models stochastic biological navigation behavior. In our experiment, we implement 2 versions of Gillespie algorithm as benchmarks.

In the first version with standard Gillespie, when the agent tumbles, it only reorients randomly without moving a step forward, and it samples event time from exponential distribution to switch between actions. This seems to be more aligned with the true biological behavior while it does not correspond with the POMDP we have established. Therefore we have designed a modified version of Gillespie algorithm where the agent moves forward while tumbling and the agent's behavior also follows the learned kinematics from previous training.We include two SSA variants to bracket the dynamics: standard SSA for instantaneous reorientation, and a run-while-tumble CTMC that explicitly models reorientation while moving, enabling path-shape and distributional comparisons.

The two versions of benchmark algorithms are presented in details as follows:

Algorithm 3 Standard SSA (instantaneous reorientation)

Require: $x_0 \in \mathbb{R}^2$, initial heading $\hat{u}_0$, light μ, run speed c_{run}, grid step Δt_{grid}, horizon $T_{\max}$, parameters $(\lambda_{\min}, \lambda_{\max}, \gamma)$, turn-noise σ.

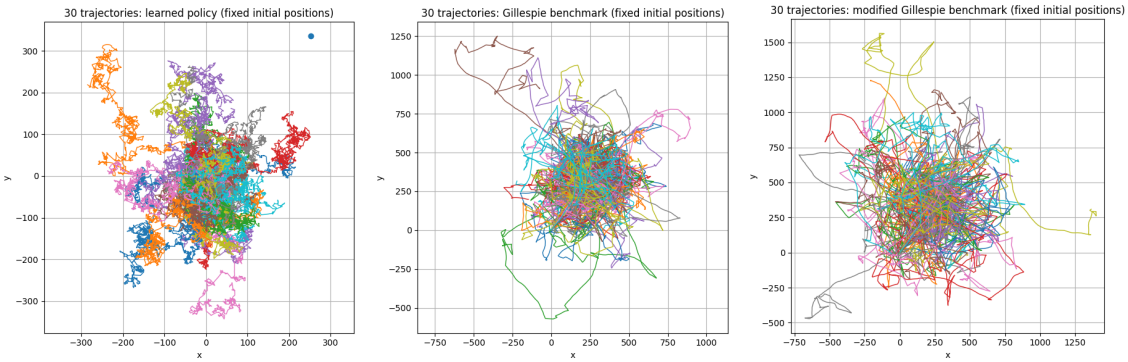
$t \leftarrow 0$; $t_{\text{out}} \leftarrow \Delta t_{\text{grid}}$; draw $\Delta t \sim \text{Exp}(\lambda_{\text{R}\to\text{T}}(x_0, \hat{u}_0))$; $t_{\text{evt}} \leftarrow t + \Delta t$.
while $t < T_{\max}$ **do**
  $t_{\text{end}} \leftarrow \min\{t_{\text{out}}, t_{\text{evt}}, T_{\text{tot}}\}$; $h \leftarrow t_{\text{end}} - t$.
  (Run advance) sample noise $\sim \mathcal{N}(0, \sigma^2)$; $\hat{u} \leftarrow R(\text{noise})\hat{u}$; $x \leftarrow x + h\, c_{\text{run}} \hat{u}$; $t \leftarrow t_{\text{end}}$.
  if $t = t_{\text{evt}}$ **then** ▷ instantaneous reorientation
    sample new heading $\hat{u} \sim \text{Unif}(\mathbb{S}^1)$;
    draw $\Delta t \sim \text{Exp}(\lambda_{\text{R}\to\text{T}}(x, \hat{u}))$; $t_{\text{evt}} \leftarrow t + \Delta t$.
  end if
  if $t = t_{\text{out}}$ or $(t = T_{\text{tot}} \wedge t < t_{\text{out}})$ **then**
    record $\{t, \Delta t_{\text{out}} = h, x, \text{align}\}$; $t_{\text{out}} \leftarrow t_{\text{out}} + \Delta t_{\text{grid}}$.
  end if
end while

Algorithm 4 Modified SSA (explicit run/tumble CTMC)

Require: x_0, initial velocity v_0 (or heading), light μ, speeds $(c_{\text{run}}, c_{\text{tum}})$, grid step Δt_{grid}, horizon $T_{\max}$, $(\lambda_{\min}, \lambda_{\max}, \gamma)$, constant $\lambda_{\text{T}\to\text{R}}$, turn-noise σ.

$t \leftarrow 0$; $t_{\text{out}} \leftarrow \Delta t_{\text{grid}}$; state $\leftarrow$ run; set $\hat{u} \leftarrow \frac{v_0}{\|v_0\|}$; $\hat{u}_{\text{tum}} \leftarrow \varnothing$.
draw $\Delta t \sim \text{Exp}(\lambda_{\text{R}\to\text{T}}(x, \hat{u}))$; $t_{\text{evt}} \leftarrow t + \Delta t$.
while $t < T_{\max}$ **do**
  $t_{\text{end}} \leftarrow \min\{t_{\text{out}}, t_{\text{evt}}, T_{\text{tot}}\}$; $h \leftarrow t_{\text{end}} - t$.
  if state = run **then**
    sample noise $\sim \mathcal{N}(0, \sigma^2)$; $\hat{u} \leftarrow R(\text{noise})\hat{u}$; $x \leftarrow x + h\, c_{\text{run}} \hat{u}$.
  else ▷ tumble
    if $\hat{u}_{\text{tum}} = \varnothing$ **then** sample $\hat{u}_{\text{tum}} \sim \text{Unif}(\mathbb{S}^1)$
    end if
    $x \leftarrow x + h\, c_{\text{tum}}\, \hat{u}_{\text{tum}}$.
  end if
  $t \leftarrow t_{\text{end}}$.
  if $t = t_{\text{evt}}$ **then** ▷ state jump
    if state = run **then**
      state $\leftarrow$ tumble; $\hat{u}_{\text{tum}} \leftarrow \varnothing$; draw $\Delta t \sim \text{Exp}(\lambda_{\text{T}\to\text{R}})$.
    else
      state $\leftarrow$ run; draw $\Delta t \sim \text{Exp}(\lambda_{\text{R}\to\text{T}}(x, \hat{u}))$.
    end if
    $t_{\text{evt}} \leftarrow t + \Delta t$.
  end if
  if $t = t_{\text{out}}$ or $(t = T_{\text{tot}} \wedge t < t_{\text{out}})$ **then**
    record $\{t, \Delta t_{\text{out}} = h, x, \text{action} = \mathbb{1}[\text{state=tumble}], \text{align}\}$; $t_{\text{out}} \leftarrow t_{\text{out}} + \Delta t_{\text{grid}}$.
  end if
end while

We simulate 30 trajectories from the previously learned policy as well as both benchmark algorithms:



(a) learned POMDP (b) standard Gillespie (c) modified Gillespie

Figure 5.6: 30 trajectories simulated from: a) learned POMDP policy; b) standard Gillespie algorithm; c) modified Gillespie algorithm. All with same fixed initial position at `[0.0,0.0]` and fixed light source center.

From the above figures of 30 trajectory simulations, we can observe that except for the first one with learned POMDP, all other trajectories show that the agent move around the light source (covered in the middle of trajectories), while in the first figure, the agent moves around somewhere far from the light source. But we need further analysis to evaluate the effectiveness of POMDP modeling.

For the tumbling ratio, we have:

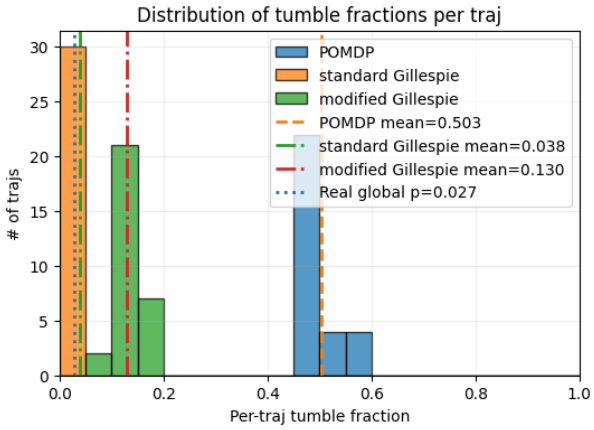
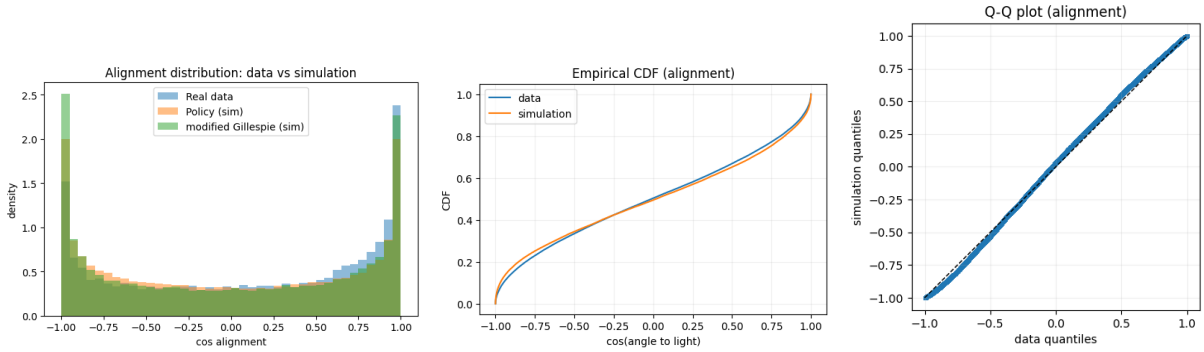


Figure 5.7: Tumbling ratio of learned POMDP, standard Gillespie, modified Gillespie and real data with 30 trajectories.

As expected, the tumbling ratio of standard Gillespie algorithm at `0.038` is closest to the real-data statistics at `0.027`, as it is the closest simulation of the true biological behaviors. And the modified Gillespie with tumbling ratio at `0.13` is relatively closer to

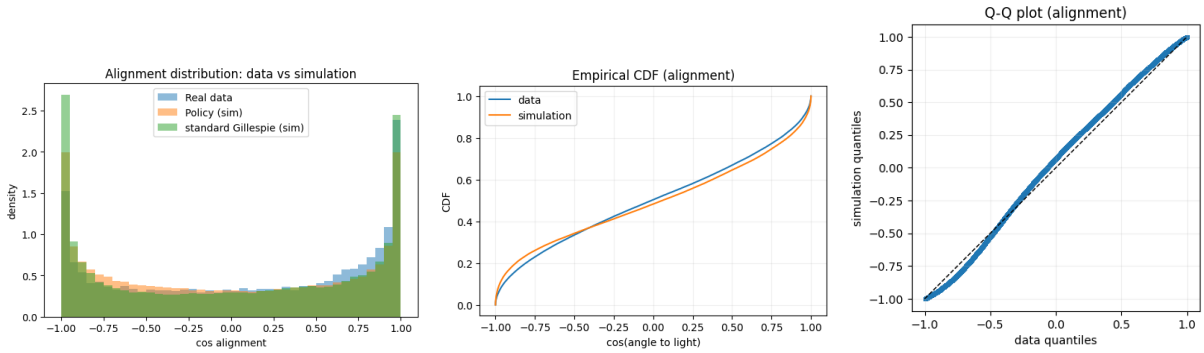
the agent's behavior under learned POMDP policy with tumbling ratio at `0.503`.

We continue to analyze the alignment distribution, first comparing the learned POMDP policy with modified Gillespie, then with standard Gillespie:



(a) overlaid histogram for alignment densities (b) empirical CDFs (c) Q-Q plot

Figure 5.8: alignment distribution comparison: a) histogram of real data, learned POMDP and modified Gillespie; b) empirical CDFs between learned POMDP and modified Gillespie; c) Q-Q plot between learned POMDP and modified Gillespie



(a) overlaid histogram for alignment densities (b) empirical CDFs (c) Q-Q plot

Figure 5.9: alignment distribution comparison: a) histogram of real data, learned POMDP and standard Gillespie; b) empirical CDFs between learned POMDP and standard Gillespie; c) Q-Q plot between learned POMDP and standard Gillespie

From the alignment perspective, we can conclude from the above figures that the learned POMDP policy aligns perfectly well with both benchmark algorithms. Even with huge gaps on the tumbling ratio, we can see that based on the above visualizations, the learned reward and retrieved policy successfully captures directional preferences the same way as the benchmark algorithms.

However, we still need to address the problem with the overestimated tumbling ratio under learned POMDP policy, for which we implement the following hyperparameter search on the temperature τ.

5.2.4 Temperature τ calibration

After several trials on the artificial modification to the temperature τ which contributes to the retrieval of π_r after IRL, we have observed that this hyperparameter has much influence on the tumbling ratio. Therefore we conduct a hyperparameter search (from coarse scale to fine scale) to match the global tumble ratio in real data:

$$\hat{p}_{\text{tum}}(\tau) = \frac{1}{T}\sum_{t}^{T} \pi_r(a = 1 \mid o_t), \qquad \tau^\star = \arg\min_{\tau} |\hat{p}_{\text{tum}}(\tau) - p_{\text{tum}}^{\text{data}}|.$$

We redo the simulation of 30 trajectories again with the new $\tau^\star = 0.010114$:

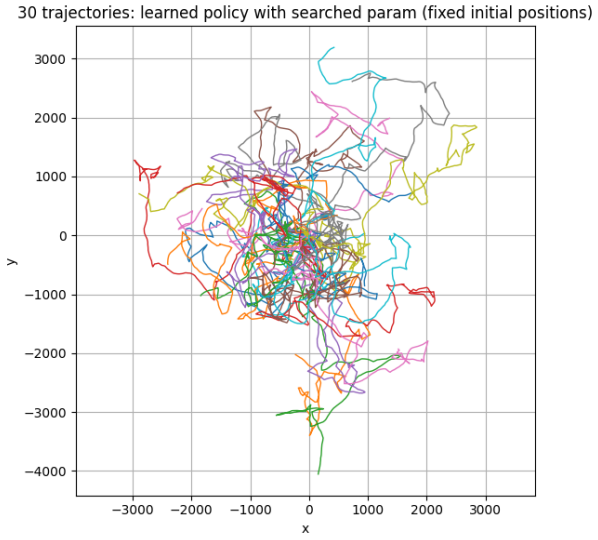


Figure 5.10: 30 trajectories simulated with learned POMDP policy and searched temperature hyperparameter $\tau^\star$

With the same initial position and light source center, we can see that with calibrated temperature, the agent moves approaches and moves around the light source and with less tumbling observed than previously.

For the tumbling ratio, we have:

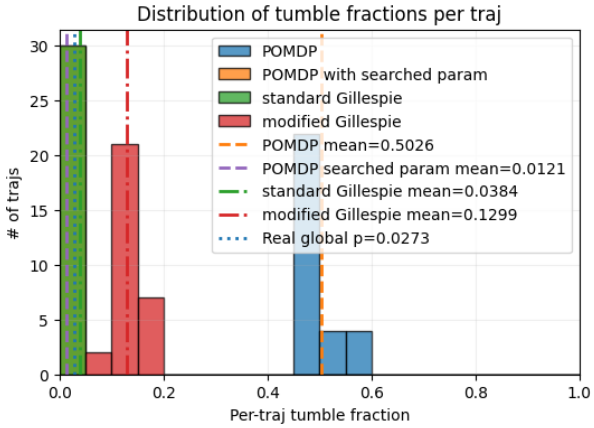
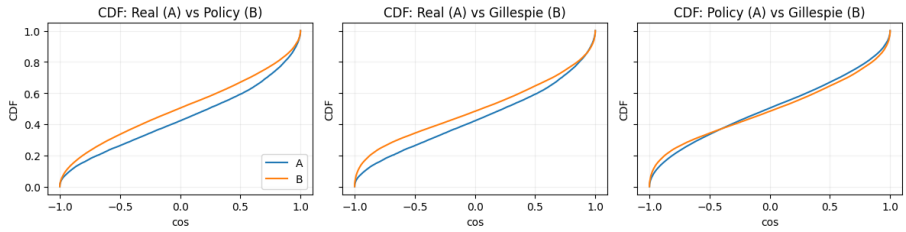


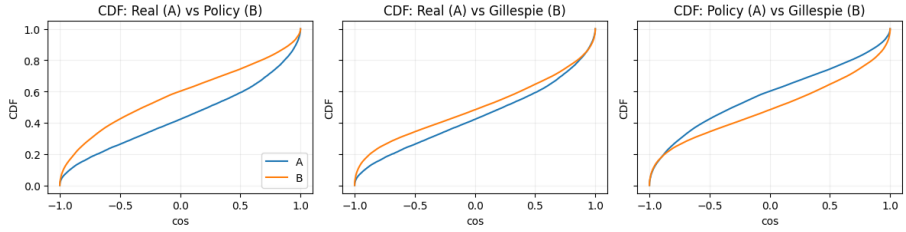
Figure 5.11: Tumbling ratio of learned POMDP , learned POMDP with calibrated hyper, standard Gillespie, modified Gillespie and real data with 30 trajectories.

Correspond with the visualization of the above trajectories' plot, the POMDP with searched hyperparameter presents with much smaller tumbling ratio at `0.0121`, closer to the real data than the previous version.

Finally we compare the alignment distribution for two versions of learned POMDP policy with real data and standard Gillespie at the same time:

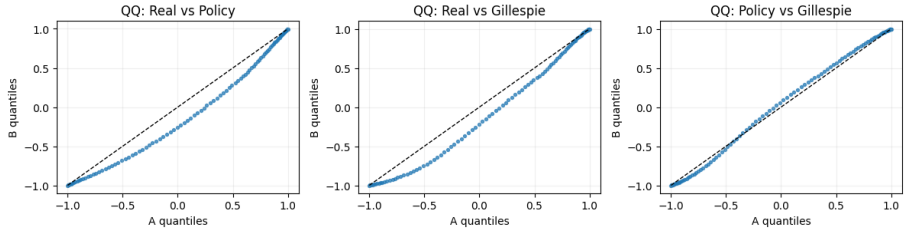


(a) Comparing with learned policy with original τ

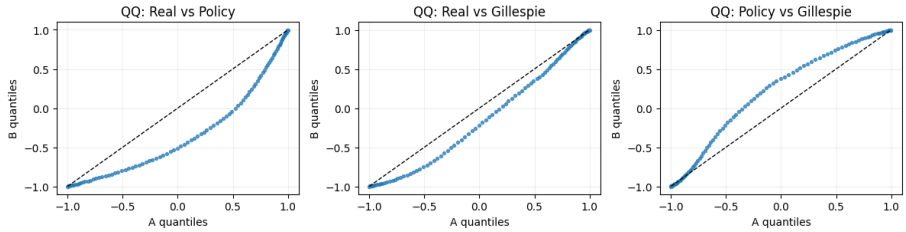


(b) Comparing with learned policy with calibrated $\tau^{\star}$

Figure 5.12: Empirical CDFs summaries for real data, standard Gillespie and learned POMDP policy with a) original temperature, b) calibrated temperature



(a) Comparing with learned policy with original τ



(b) Comparing with learned policy with calibrated $\tau^\star$

Figure 5.13: Quantile-Quantile plots summaries for real data, standard Gillespie and learned POMDP policy with a) original temperature, b) calibrated temperature

From the above comparison, we can see the trade-offs between tumble ratio and alignment distribution: with original τ, the policy achieves better alignment at the cost of over-tumbling, while calibrated hyperparameter matches the frequency but sacrifices alignment. This trade-off is physically meaningful: the temperature τ adjusts how often we turn, but its modification also changes the directional structure simultaneously.

6. Discussions

Several studies have been made to demonstrate different ways of obtaining the "optimal reward/ strategy". Though we have experimented with inverse reinforcement learning to learn the reward vector of exploitation, there are still much more variations of this method to explore.

6.1 Optimal sensing

[10] provide a normative framework of optimal sensing and control: cast chemotaxis as a POMDP, solve an HJB on the belief and obtain a **belief-only optimal policy**——an exponential law for the tumble rate, modeled as function of belief. The objective maximizes **discounted up-gradient drift** minus a **pathwise KL cost** while a continuous-time filter ties sensory noise and responsive adaptation into the closed loop. This establishes that the posterior belief is sufficient and that the policy depends only on belief. No demonstrations are required; the result is an offline policy that is provably optimal given the model,which is separate from the online update of belief.

The pros for this method is that it ensures the global optimality under assumed dynamics/noise; interpretable (filter + control); matches known biochemistry. No reward guessing or demos. However, it requires accurate generative model; solving HJB may be heavy; utility/cost must be chosen a priori; model mismatch degrades performance.

In our framework, we have adopted the same idea of belief-centric view for phototaxis experiments.The key differences between the study of [10] and ours are:

- **Time & control:** their model is continuous-time with rate control while in ours we have discrete run/tumble actions with discrete-time which suis quantized observations and multi-source scenes.
- **Policy derivation:** they obtain the closed-form exponential law via HJB with KL regularization, while we use one-step-look-ahead exploitation and exploration state-action value accompanied with `argmax` to choose action, and we implement it mechanistically with biochemical pipeline.
- **Task design:** they focus on single-gradient climbing and noise–cost trade-offs; we extend to multi-source phototaxis and curiosity-driven exploration, comparing different observation aggregation and their statistics/behavior under multiple phototaxis scenarios.
- **Biochemical mapping:** they align belief/prediction with receptor activity/methylation; we translate posterior filtering and value terms into CRN-ODE, yielding experimentally executable dynamics.

The research from [10] emphasizes "optimal control under uncertainty" —net up-gradient motion with KL cost, captures the biological trade-off “accuracy vs effort”. The

belief-optimal control perspective provides valuable idea for cell-scale navigation tasks, and might be feasible to be applied to phototaxis tasks by bridging the discrete framework with their continuous rate control. It might provide a principled baseline: an offline optimal policy against which learned policy can be compared.

6.2 Inverse Reinforcement learning

The objective of inverse reinforcement learning (IRL) is to get the reward from the policy instead of the traditional thinking of the other way around.

In our experiment, we use the idea of inverse reinforcement learning to learn the optimal reward under expert's guidance and retrieve a policy from the learned reward with soft value. According to our research, many studies have already demonstrated different methods of IRL under POMDP framework and a linear reward class, where the IRL objective seeks to learn the reward vector and a policy that explain the expert data while respecting POMDP dynamics. Here we present studies on 3 different algorithms: Max-Margin between values(MMV), Max-Margin between Feature Expectation(MMFE) by [6], and Maximum Causal Entropy(MCE) by [1].

MMV The key idea of MMV is to learn the reward vector α by maximizing the sum of value margins between expert policy π_E and a growing set of candidate policies Π over unique beliefs B^{π_E} visited in demonstrations.

Algorithm 5 IRL for POMDP$\backslash R$ from sampled trajectories using the MMV method

Require: POMDP$\backslash R$ $\langle S, A, Z, T, O, b_0, \gamma \rangle$, basis functions ϕ, M expert trajectories

Choose a set B^{π_E} of all the unique beliefs in the trajectories.
Choose a random initial policy π_1 and set $\Pi \leftarrow \{\pi_1\}$.
for $k = 1$ to *MaxIter* **do**
    Find $\hat{\alpha}$ by solving the linear program:
    $$\max_{\hat{\alpha}} \sum_{\pi \in \Pi} \sum_{b \in B^{\pi_E}} p\Big(\hat{V}^{\pi_E}(b) - V^{\pi}(b)\Big) \; - \; \lambda \, \|\hat{\alpha}^\top \phi\|_1$$
    subject to $|\hat{\alpha}_i| \leq 1, \quad i = 1, 2, \ldots, d.$
    Compute an optimal policy π_{k+1} for the POMDP with $\hat{R} = \hat{\alpha}_k^\top \phi$.
    if $\left[\hat{V}^{\pi_E}(b) - V^{\pi_{k+1}}(b)\right] \leq \varepsilon, \;\; \forall b \in B^{\pi_E}$ **then**
        return $\hat{R} = \hat{\alpha}_k^\top \phi$
    else
        $\Pi \leftarrow \Pi \cup \{\pi_{k+1}\}$
    end if
end for
$K \leftarrow \arg\min_{\kappa:\, \pi_\kappa \in \Pi} \; \max_{b \in B^{\pi_E}} \left[\hat{V}^{\pi_E}(b) - V^{\pi_\kappa}(b)\right]$
return $\hat{R} = \hat{\alpha}_K^\top \phi$

This margin-based algorithm can be computed directly and it uses multiple beliefs that contribute to a tighter feasible reward set. The optimization problem is linear program

(LP) which is simple and L1 norm can be added to encourage sparse and interpretable rewards. Meanwhile, it requires belief reconstruction and repeated value solves; sensitive if beliefs are noisy or features misspecified.

MMFE In this algorithm, it first re-expresses the value at initial belief as $V^{\pi}(b_0) = \alpha^{\top}\mu(\pi)$, where $\mu(\pi)$ is the discounted feature expectation computed from occupancy measures. And it learns the reward vector α by maximizing a margin in feature space between expert μ_E and candidate $\mu(\pi)$.

Algorithm 6 IRL for POMDP$\backslash R$ from sampled trajectories using the MMFE method

Require: POMDP$\backslash R$ $\langle S, A, Z, T, O, b_0, \gamma\rangle$, basis functions ϕ, M expert trajectories

Choose a random initial weight vector α_0.
$\Pi \leftarrow \emptyset$, $\Omega \leftarrow \emptyset$, $t \leftarrow \infty$.
for $k = 1$ to *MaxIter* **do**
  Compute an optimal policy π_{k-1} for the POMDP with $R = \alpha_{k-1}^{\top}\phi$.
  $\Pi \leftarrow \Pi \cup \{\pi_{k-1}\}$, $\Omega \leftarrow \Omega \cup \{\alpha_{k-1}\}$.
  if $t \leq \varepsilon$ **then**
    break
  end if
  Solve the following optimization problem:

$$\max_{t,\,\alpha_k} \; t$$
$$\text{subject to} \quad \alpha_k^{\top}\mu_E \;\geq\; \alpha_k^{\top}\mu(\pi) + t, \quad \forall\, \pi \in \Pi,$$
$$\|\alpha_k\|_2 \leq 1.$$

end for
$K \leftarrow \arg\min_{\kappa:\,\pi_\kappa \in \Pi} \; \| \mu_E - \mu(\pi_\kappa) \|_2$
return $R = \alpha_K^{\top}\phi$

This algorithm works directly in feature space; more robust to slightly suboptimal experts. And it avoids multi-belief value constraints by using only value at initial belief via $\mu(\pi)$. However, it solves a QCP which is heavier than LP in MMV. It also requires accurate occupancies or an FSC approximation. And just like MMV, it still depends on feature quality.

In the experiments by [6], they compare the efficiency of MMV and MMFE as IRL for POMDP with different POMDP problems and different sets of basis functions. MMFE outpreforms MMV in most of the tasks and more stable than MMV. Both methods spend most of time solving POMDP.

MCE In the study of [1], with only observations ω, the idea of this algorithm is to maximize causal entropy while matching feature expectations via EM: E-step: compute completed demo features using posteriors over latent trajectories; M-step: solve the convex MaxEnt/MCE dual to update reward vector λ.

Algorithm 7 uMaxEnt(EM) for Noisy/Partial Observations

Require: Observations $\Omega = \{\omega\}$, posterior model $P(\omega|X)$, feature $\phi(X)$.
Ensure: $r_{\text{uME}}(\tau) = \lambda^\top \phi(\tau)$ with $P_\lambda(X) \propto \exp(\lambda^\top \phi(X))$.

1: Initialize λ'
2: **while** not converged **do**
3: **E-step:** posterior moment (uncertain empirical feature)

$$\hat{\phi} = \sum_{\omega\in\Omega} \tilde{P}(\omega)\, \mathbb{E}_{P_{\lambda'}(X|\omega)}\big[\phi(X)\big], \qquad P_{\lambda'}(X \mid \omega) = \frac{P(\omega|X)\, e^{\lambda'^\top \phi(X)}}{\sum_{X'} P(\omega|X')\, e^{\lambda'^\top \phi(X')}}.$$

4: **M-step:** update λ by matching model moments

$$\lambda \leftarrow \lambda + \eta\Big(\hat{\phi} - \mathbb{E}_{P_\lambda}[\phi(X)]\Big), \quad \text{where} \quad P_\lambda(X) \propto e^{\lambda^\top \phi(X)}.$$

5: $\lambda' \leftarrow \lambda$
6: **end while**

This algorithm handles partial/noisy observations yields stochastic policies consistent with biological variability; monotonic likelihood ascent via EM, and there is no need to reconstruct exact belief trajectories. However, it is computationally heaviest among three IRL methods, and EM may find only a local optimum. It also still requires good feature functions with temperature and regularization hyperparameters that need to be tuned.

To summarize, MMV maximizes value margins over beliefs (lightweight but belief-dependent); MMFE maximizes feature-expectation margins via occupancies (stable offline); MCE maximizes observed-data likelihood under causal-entropy with EM (robust to partial/noisy observations). If we explore the further extension to our phototaxis POMDP, we can prioritize MMFE when belief replay is reliable and MCE when only sensor observations are available; both of them can then be evaluated against the offline optimal chemotaxis baseline.

6.3 Limitations and future applications

6.3.1 Limitations and improvements of current study

Memoryless belief update In our numerical pipeline of POMDP, we use a memoryless belief update with $b_t^+(h) \propto Z[o_t \mid h]b_0(h)$ and in the policy the action scoring relies on a single prediction step with $b' = T_a b^+$. This simplifies inference but forfeits temporal memory. A more realistic POMDP should propagate beliefs $b_{t+1} \propto Z[o_{t+1} \mid h]T_a b_t(h)$ to roll with memories. And in our current version of POMDP with numerical pipeline and corresponding CRN-ODE pipeline, we try to simplify the CRN design with only one-step forward algorithm. However, the policy might work better to exploit multi-step information instead of only one-step ahead prediction. Plus, we could also see how curiosity affects agent's behavior with memoryful belief update. Though the current design of CRN-ODE does not support belief update, the idea of designing new CRN-ODE to align with memoryful belief update is also worth considering.

Observation design and sensor configuration Our two-channel discretized observation and WTA aggregator are convenient but assume angular selectivity and strong quantization; comments note that physical photosensors may lack such angular resolution, and the binning can conflate "front and back" situations for $m = 2$. Improving the observation model with more bins or richer angular response, or learned encoder($o = \psi(y_L, y_R)$ with small MLP) would address this. And by continuing with the learnable-kernel and memoryful-belief update idea, we could also learn the transition kernel T in the policy-training section of the experiment by implementing KL regularization on T. This could address identifiability and stabilize training.

6.3.2 Broader applications

swarm coordination Even though in this study, we mainly consider the independent behavior of single-agent. However in the real scenario of biological experiments, many researchers study on the collective phototaxis. And the belief-centric rewards in our POMDP framework might be idea to be extended to collective behavioral pattern (with consensus on belief b_t, collision-avoidance considered in exploitation-exploration rewards etc.), opening to distributed POMDPs.

Micro-swimmers & bio-robotics Back to micro-scale research, the POMDP+CRN-ODE split (belief & action scoring as chemical modules) suggests a route to controllers for phototactic/chemotactic microrobots, where reaction networks implement on-chip sensing-decision loops under uncertainty. This might bring new ideas to solve similar or even more complex navigation tasks.

Autonomous driving & field robotics To a broader view, robots act with occlusions and latent world states. This study on belief-centric trainable policies with learnable kernels can be a drop-in surrogate for modern planning in robotics. The implementation of IRL in our experiment becomes a cost map or shaping signal for model-based control or RL fine-tuning in robotic tasks. And our inclusion of curiosity/ mutual information term in policy values is a principled **next-best-view** bonus in action-sensing perspective, substituting depth/semantic sensors yields the same acquisition rule for occluded scenes.

Neuroscience & Systems Biology We can also further extend the idea of this study to other broader field of biology. From IRL, we recover reward vector that explains action frequencies conditioned on sensory bins. This is a normative hypothesis about what the organism optimizes—complementary to mechanistic models. In our hazard policy design, the hazard function maps to firing or transition rates in spiking/Markov descriptions of neural control; the discrete-time probability matches standard point-process links. And from the biochemical perspective, the CRN-ODE we have designed can be part of the biochemical cascade, whose structure could be optimized and be mapped to futile cycles and mass-action motifs, returning back to where we started. Also, the tumble ratio and alignment distribution provide compact behavioral targets; matching these while varying sensory/biochemical parameters enables phenotype-to-mechanism inference.

Acknowledgments

We thank Pierre Bessieère and Jacques Droulez for advice on the observation design and evaluation protocol, and for discussions that improved our evaluation setup and presentation. And we thank supports from Sorbonne University and SONY CSL for this research.

This internship was funded by AMIES – PEPS project « Generative models of chemotaxis and phototaxis ».

References

[1] Kenneth Bogert, Yikang Gui, and Prashant Doshi. *IRL with Partial Observations using the Principle of Uncertain Maximum Entropy*. 2022. arXiv: 2208.06988 [cs.LG]. URL: https://arxiv.org/abs/2208.06988.

[2] Robert Brijder. *Computing with Chemical Reaction Networks: A Tutorial*. 2018. arXiv: 1811.10361 [cs.ET]. URL: https://arxiv.org/abs/1811.10361.

[3] Luca Cardelli, Marta Kwiatkowska, and Luca Laurenti. *Programming Discrete Distributions with Chemical Reaction Networks*. 2018. arXiv: 1601.02578 [cs.DC]. URL: https://arxiv.org/abs/1601.02578.

[4] Ho-Lin Chen, David Doty, and David Soloveichik. *Deterministic Function Computation with Chemical Reaction Networks*. 2013. arXiv: 1204.4176 [cs.CC]. URL: https://arxiv.org/abs/1204.4176.

[5] Xiuli Chen, Kieran Mohr, and Joseph M. Galea. "Predicting explorative motor learning using decision-making and motor noise". In: *PLOS Computational Biology* 13.4 (Apr. 2017), pp. 1–33. DOI: 10.1371/journal.pcbi.1005503. URL: https://doi.org/10.1371/journal.pcbi.1005503.

[6] Jaedeug Choi and Kee-Eung Kim. "Inverse Reinforcement Learning in Partially Observable Environments". In: *J. Mach. Learn. Res.* 12.null (July 2011), pp. 691–730. ISSN: 1532-4435.

[7] David Colliaux, Pierre Bessière, and Jacques Droulez. "Cell signaling as a probabilistic computer". In: *International Journal of Approximate Reasoning* 83 (2017), pp. 385–399. ISSN: 0888-613X. DOI: https://doi.org/10.1016/j.ijar.2016.10.012. URL: https://www.sciencedirect.com/science/article/pii/S0888613X16302110.

[8] A Goldbeter and D E Koshland. "An amplified sensitivity arising from covalent modification in biological systems." In: *Proceedings of the National Academy of Sciences* 78.11 (1981), pp. 6840–6844. DOI: 10.1073/pnas.78.11.6840. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.78.11.6840. URL: https://www.pnas.org/doi/abs/10.1073/pnas.78.11.6840.

[9] Justus A. Kromer et al. "Decision making improves sperm chemotaxis in the presence of noise". In: *PLOS Computational Biology* 14.4 (Apr. 2018), pp. 1–15. DOI: 10.1371/journal.pcbi.1006109. URL: https://doi.org/10.1371/journal.pcbi.1006109.

[10] Kento Nakamura and Tetsuya J. Kobayashi. "Optimal sensing and control of run-and-tumble chemotaxis". In: *Phys. Rev. Res.* 4 (1 Feb. 2022), p. 013120. DOI: 10.1103/PhysRevResearch.4.013120. URL: https://link.aps.org/doi/10.1103/PhysRevResearch.4.013120.

[11] Sam Nicol and Iadine Chadès. "Which States Matter? An Application of an Intelligent Discretization Method to Solve a Continuous POMDP in Conservation Biology". In: *PLOS ONE* 7.2 (Feb. 2012), pp. 1–8. DOI: 10.1371/journal.pone.0028993. URL: https://doi.org/10.1371/journal.pone.0028993.

[12] Tristan Ursell et al. "Motility Enhancement through Surface Modification Is Sufficient for Cyanobacterial Community Organization during Phototaxis". In: *PLOS Computational Biology* 9.9 (Sept. 2013), pp. 1–14. DOI: 10.1371/journal.pcbi.1003205. URL: https://doi.org/10.1371/journal.pcbi.1003205.

[13] Carsten Wiuf et al. "A reaction network scheme for hidden Markov model parameter learning". In: *Journal of The Royal Society Interface* 20.203 (2023), p. 20220877. DOI: 10.1098/rsif.2022.0877. eprint: https://royalsocietypublishing.org/doi/pdf/10.1098/rsif.2022.0877. URL: https://royalsocietypublishing.org/doi/abs/10.1098/rsif.2022.0877.

[14] Ruohan Zhang et al. "Modeling sensory-motor decisions in natural behavior". In: *PLOS Computational Biology* 14.10 (Oct. 2018), pp. 1–22. DOI: 10.1371/journal.pcbi.1006518. URL: https://doi.org/10.1371/journal.pcbi.1006518.

A. Supplementary Information on CRN construction

A.1 Build CRNs from semilinear functions

In the work of [4], the researchers mainly proposed that all semilinear functions can be deterministically computable by CRNs.

And here we tried to look into their proposed method to build deterministic CRNs from affine partial functions and semilinear functions.

A.1.1 Deterministic CRN built from affine partial function with lemma 4.2 in [4]

We first look into the case of affine partial functions, which are defined only on a subset of the input space where they take form of an affine expression:

$$f(\vec{x}) = A\vec{x} + \vec{b} = Y \in \mathbb{N}^l \quad \text{for } \vec{x} \in \text{dom}(f) \subseteq \mathbb{N}^k$$

For $j \in \{1, \ldots, l\}$, we can decompose the target Y as:

$$Y_j = Y_j^P - Y_j^C \quad \text{where} \quad Y_j^P, Y_j^C \geq 0$$

Y_j is the j-th component fo the output vector $Y \in \mathbb{N}^l$, Y_j^P and Y_j^P respectively represent the positive (positively produced output species) and negative part (cancellation indicators that are not consumed directly)to avoid negative molecular counts in chemical reactions. And the decomposition $Y_j = Y_j^P - Y_j^C$ is only a logical computation that can be realized through chemical reactions for conflict detection and path selection (introduced in the next part with semilinear functions).

More explicitly, we have:

$$Y_j^P = b_j + \frac{1}{d_j} \sum_{i=1}^{k} \max(0, n_{i,j})(x_i - c_i)$$

$$Y_j^C = -\frac{1}{d_j} \sum_{i=1}^{k} \min(0, n_{i,j})(x_i - c_i)$$

We can decompose in this way because:

$$
\begin{aligned}
Y_j &= b_j + \frac{1}{d_j}\sum_{i=1}^{k}\max(0, n_{i,j})(x_i - c_i) + \frac{1}{d_j}\sum_{i=1}^{k}\min(0, n_{i,j})(x_i - c_i) \\
&= b_j + \frac{1}{d_j}(\sum_{i:n_{ij}\geq 0} n_{ij}(x_i - c_i) + \sum_{i:n_{ij}\leq 0} n_{ij}(x_i - c_i)) \\
&= b_j + \frac{1}{d_j}(\sum_{i=1}^{k} n_{ij}(x_i - c_i)) \\
&= \sum_{i=1}^{k}\frac{n_{ij}}{d_j}x_i - \sum_{i=1}^{k} n_{ij}c_i + b_j \quad \text{where} \quad b_j, c_i.d_j \in \mathbb{N} \\
&= A_j \cdot \vec{x} + \tilde{b}_j \quad \text{where } A_j \text{ is the} j\text{-th line of } A, \text{ and } \tilde{b}_j \text{ the } j\text{-th component of } \vec{b}
\end{aligned}
$$

Then we can define a chemical reaction network or a chemical reaction circuit(CRC) that receives input of $X_1, \ldots X_k$ and outputs $(Y_1^p, \ldots, Y_l^P, Y_1^C, \ldots, Y_l^C)$ with the following steps:

1. **Build constant offset b_j:** for $j \in \{1, \ldots, l\}$, initialize with $b_j \times Y_j^P$
2. **Build offset checks $(x_i - c_i)$:** we want only $(x_i - c_i)\times$ species X_i effective for the computation, the idea is that we first skip/consume first $c_i \times X_i$, then build productive reactions from the $(c_i{+}1)$-th X_i.

Algorithm 8 Offset Check for $x_i - c_i$

for $i = 1$ to k **do**
  Input: x_i molecules of species X_i
  Goal: Only use $(x_i - c_i)$ molecules for downstream production
  Initialize offset counter: C_i^0
  for $m = 0$ to $c_i - 1$ **do**
    $C_i^m + X_i \to C_i^{m+1}$ ▷ Offset counting (skip)
  end for
  $C_i^{c_i} + X_i \to C_i^{c_i} + X_i'$ ▷ Start producing X_i'
end for
Output: $\max(0, x_i - c_i) \times X_i$

3. **Decompose X_i' into $\sum_j^l X_{ij}$:** we copy X_i' for l times and each copy aligns with a certain y_j, where $j \in \{1, \ldots, l\}$. To demonstrate the process, we have reaction:

$$X_i^j \to X_{i,1} + \cdots + X_{i,l}$$

4. **Build $\sum_{i=1}^k n_{ij}(x_i - c_i)$** We introduce the intermediate products Z_j^P and Z_j^C whose production is determined by the sign of n_{ij}. More explicitly, we can break down $\sum_{i=1}^k n_{ij}(x_i - c_i)$ as:

$$\sum_{i=1}^{k} n_{ij}(x_i - c_i) = \sum_{i,n_{ij}>=0}^{k} n_{ij}(x_i - c_i) - \sum_{i,n_{ij}<0}^{k} -n_{ij}(x_i - c_i)$$

For the two components on the right side of the equation we have two corresponding reactions as follows:

$$\begin{aligned} \text{for all} \quad & i \in \{1,\dots,l\}, \quad j \in \{1,\dots,k\}: \\ \text{if} \quad & n_{ij} \geq 0: \quad X_{ij} \to n_{ij} Z_j^P \\ & \text{else:} \quad X_{ij} \to (-n_{ij}) Z_j^C \end{aligned}$$

From the above reactions, the count for Z_j^P we obtained is $\sum_{i,n_{ij}>=0}^{k} n_{ij}(x_i - c_i) = d_j(y_j^P - b_j)$, and the count for Z_j^C is $\sum_{i,n_{ij}<0}^{k} n_{ij}(x_i - c_i) = d_j y_j^C$ Therefore, we have the next step to obtain Y_j^P and Y_j^C.

5. **Build $\frac{1}{d_j}$:** Here for every $d_j \times Z_j^P$ we want to get a Y_j^P and for every $d_j \times Z_J^C$ we want to get a Y_j^C. The idea is very similar with step 2, but here we consume the first $(d_j - 1) \times Z_j$ while adding to the counts and start producing Y_j with the d_j-th Z_j.

Algorithm 9 Divide d_j

for $j = 1$ to l **do**
  Input: $\sum_{i,n_{ij}>=0}^{k} n_{ij}(x_i - c_i)$ molecules of species Z_j^P and $\sum_{i,n_{ij}<0}^{k} n_{ij}(x_i - c_i)$ molecules of species Z_j^C
  Goal: Obtain $\frac{1}{d_j}$ number of the species
  Initialize counters: $D_j^{0,P}$,$D_j^{0,C}$
  for $m = 0$ to $d_j - 1$ **do**
    if $m < d_j - 1$ **then**
      $D_j^{m,P} + Z_j^P \to D_j^{m+1,P}$
      $D_j^{m,C} + Z_j^C \to D_j^{m+1,C}$ ▷ Consume $(d_j - 1) \times Z_j$
    else$m = d_j - 1$
      $D_j^{m,P} + Z_j^P \to D_j^{0,P} + Y_j^P$
      $D_j^{m,C} + Z_j^C \to D_j^{0,C} + Y_j^C$ ▷ clear the counter and produce a Y_j
    end if
  end for
end for
Output: Y_j^P, Y_j^C

6. **Obtain target result:** $Y_j = Y_j^P - Y_j^C$. However, this is only a mathematical representation since chemically the computation of *minus* or *reduction* cannot be done. In the algorithm, our final output is actually the set of (Y_j^P, Y_j^C).

Example: We can take a simple example of $f(x_1, x_2) = 2x_1 - x_2 + 1$ and to align it with the previous decomposition, we have the corresponding parameters:$b_1 = 1, c_1 = c_2 = 0, d_1 = 1, n_{1,1} = 2, n_{2,1} = -1, x_i = 1 \quad \text{for} i = 1, 2$. And we can construct a CRN with the following steps:

1. initialize with 1 $\times Y_1^P$.
2. $$C_1^0 + X_1 \to C_1^0 + X_1'$$ $$C_2^0 + X_2 \to C_2^0 + X_2'$$
3. $$X_1' \to X_{1,1}$$ $$X_2' \to X_{2,1}$$
4. $$D_1^{0,P} + Z_1^P \to D_1^{0,P} + Y_1^P$$ $$D_1^{0,C} + Z_1^C \to D_1^{0,C} + Y_1^C$$

Now we turn to the method that builds CRNs from semilinear functions.

A.1.2 Deterministic CRN built from semilinear functions with lemma 4.4 in [4]

Semilinear functions is a function defined in pieces, where each piece is an affine partial function, valid only on a specific region of the domain described by linear constraints. So we are going to use the results of the previous method in the following steps:

1. **Piecewise expression:** for any semilinear function f, we denote:
$$f(x) = f_i(x), \quad \text{if} \quad x \in dom(f_i), \text{for any affine partial functions} \quad f_i$$
2. **parallel computation for every $f_i(x)$:** from lemma 4.2 in [4] we build CRN for every $f_i(x)$ with the method from previous section, that outputs candidates, denoted as $\hat{Y}_{i,j}^P$, and $\hat{Y}_{i,j}^C$.
3. **Introduce a separate chemical reaction decider (CRD) to compute predicate ψ_i:** the predicate outputs a boolean value that determines whether or not we select f_i ($x \in dom(f_i)$).
$$\psi_i = \begin{cases} T_i & \text{if } x \in dom(f_i) \\ F_i & \text{else} \end{cases}$$
4. **Only activate the chosen f_i and terminate the others**
   - **when $\psi_i = T_i$:**
   $$T_i + \hat{Y}_{i,j}^P \to T_i + Y_{i,j}^P + Y_j$$ $$T_i + \hat{Y}_{i,j}^C \to T_i + Y_{i,j}^C$$

where Y_j is the actual output of this algorithm, and what we want is $Y_j = Y^P_{i,j} - Y^C_{i,j}$. If f_i is the chosen function, we make $\hat{Y^P_{i,j}}$ effective for $Y^P_{i,j}$ and since $Y^C_{i,j}$ is something we want to clear, in the later steps we will pair up the $Y^P_{i,j}$ and $Y^C_{i,j}$ in conflict check and we will erase the paired-up $Y^C_{i,j}$ from $Y^P_{i,j}$.

- **when $\psi_i = F_i$ (meaning that we will erase a falsely activated f_i):**

$$\begin{aligned} F_i + Y^P_{i,j} &\to F_i + M_{i,j} \\ M_{i,j} + Y_j &\to \hat{Y^P_{i,j}} \\ F_i + Y^C_{i,j} &\to F_i + \hat{Y^C_{i,j}} \end{aligned}$$

where $M_{i,j}$ is an introduced intermediate as an error marker. In the above reactions, we erase the falsely added $Y^P_{i,j}$ from the container for total output Y_j and we return to the inactivated state of $\hat{Y^P_{i,j}}$ and same for the wrongfully activated $Y^C_{i,j}$ that returns to $\hat{Y^C_{i,j}}$.

- **conflict checks:**

$$\begin{aligned} Y^P_{i,j} + Y^C_{i,j} &\to K_j \\ K_j + Y_j &\to \varnothing \end{aligned}$$

where K_j is the conflict marker that indicates there are more than one f_i selected. And the last reaction that produces $\varnothing$ is to clear the total output container Y_j to assume only one f_i is selected to avoid confusion.

Example: we consider the following function:

$$\begin{aligned} f(x_1, x_2) &= min(x_1, x_2) \\ &= \begin{cases} x_1, & x_1 \leq x_2 \\ x_2, & x_2 < x_1 \end{cases} \end{aligned}$$

t And we note $y_1 = x_1$, $y_2 = x_2$. With lemma 4.2 we can take $b_j = c_i = 0$, and we can also ignore C^m, D^m_j to directly construct the reactions:

$$\begin{aligned} X_1 &\to \hat{Y^P_1} \\ X_2 &\to \hat{Y^P_2} \end{aligned}$$

And we can construct 2 predicates for $x_1 \leq x_2$ and $x_2 < x_1$

$$\phi_1 = \begin{cases} T_1 & x_1 \leq x_2 \\ F_1 & else \end{cases}$$

$$\phi_2 = \begin{cases} T_2 & x_2 < x_1 \\ F_2 & else \end{cases}$$

for $i \in \{1, 2\}$:

$$T_i + \hat{Y_i^P} \to F_i + M_i$$
$$F_i + Y_i^P \to F_i + M_i$$
$$M_i + Y \to \hat{Y_i^P}$$

Since there are no $\hat{Y_i^C}$ to start with, there is no need for conflict checks.

A.1.3 Conclusion:

This proposed method can only be used in discrete setting and CRNs are constructed deterministically by a certain kind of functions: semilinear functions. This cannot be directly applied to the phototaxis case since our research goal focuses on the target concentration by computing the ODE system constructed from the CRN, which is not achievable with this method.

A.2 Build CRNs from pmf

In this paper [3] the authors looked into how to build CRNs or a CRS(System) from probability mass function with finite support. This is also a discrete case and the framework is as follows:

1. **Define a probability mass function (pmf) with finite support:** Given a pmf $f : \mathbb{N} \to [0, 1]$ with finite support $J = \{z_1, \ldots, z_{|J|}\}$ such that $\sum_i^{|J|} f(z_i) = 1$.We can denote the function f as:
$$f(y) = \begin{cases} p_1 & y = z_1 \\ \cdots & \\ p_n & y = z_n \end{cases}$$

2. **Construct CRS (or CRN):** The goal is to design a CRN that at its steady state, the concentration of the network output follows the distribution of $f(y)$.

   Define a CRS $C = (\Lambda, R, x_0)$ as follows:

   - $x_0(\lambda)$: initial number of particles in species λ
   - Species set Λ:
     - λ_i: carries the target output z_i with $x_0(\lambda_i) = z_i$.
     - λ_z: ensures only one path is activated as a global trigger with $x_0(\lambda_z) = 1$
     - $\lambda_{i,i}$: ensures that only path i is activated with $x_0(\lambda_{i,i}) = 0$.
     - λ_{out}: records the distribution results as the network output with $x_0(\lambda_{out}) = 0$
   - Reactions set R:

     (a) **Choice of path:** only activate one pathway from the initial state:
$$\Gamma_{i,1} : \lambda_z \xrightarrow{f(z_i)} \lambda_{i,i}$$

We can consider this reaction as pushing the "trigger" button as we choose only one pathway with the rate of $f(z_i)$, that results in choosing the pathway i.

(b) **Determine the output of the activated path**

$$\Gamma_{i,2} : \lambda_{i,i} + \lambda_i \to \lambda_{i,i} + \lambda_{out}$$

where $\lambda_{i,i}$ is the catalyzer in the reaction that only allows the entry of λ_i in the reaction.

Example: Given a pmf f:

$$f(y) = \begin{cases} \frac{1}{6} & y = 2 \\ \frac{1}{3} & y = 5 \\ \frac{1}{2} & y = 10 \\ 0 & \text{else} \end{cases}$$

We can build a CRS as follows:

$$\Lambda = \{\lambda_1, \lambda_2, \lambda_3, \lambda_{1,1}, \lambda_{2,2}, \lambda_{3,3}, \lambda_z, \lambda_{out}\}.$$
$$x_0(\lambda_z) = 1, x_0(\lambda_1) = 2, x_0(\lambda_2) = 5, x_0(\lambda_3) = 10,$$
$$x_0(\lambda_{1,1}) = x_0(\lambda_{2,2}) = x_0(\lambda_{3,3}) = 0,$$
$$\lambda_z \xrightarrow{\frac{1}{6}} \lambda_{1,1}, \lambda_z \xrightarrow{\frac{1}{3}} \lambda_{i,i}, \lambda_z \xrightarrow{\frac{1}{2}} \lambda_{3,3}$$
$$\lambda_{1,1} + \lambda_1 \to \lambda_{1,1} + \lambda_{out}, \lambda_{2,2} + \lambda_2 \to \lambda_{2,2} + \lambda_{out}, \lambda_{3,3} + \lambda_3 \to \lambda_{3,3} + \lambda_{out}$$

In the paper the author also mentioned some examples of simplified CRNs that computes Poisson, Binomial and Uniform distributions, but without any details on the how these CRNs are constructed. We directly demonstrate the CRNs here:

- Poisson: $\pi(\lambda) = \frac{\mu^\lambda e^{-\mu}}{\lambda!}$:

$$\varnothing \xrightarrow{k_1} \lambda$$
$$\lambda \xrightarrow{k_2} \varnothing$$

- Binomial: $\pi_\lambda(y) = \binom{k}{y} c^y (1-c)^{k-y}$:

$$\lambda_1 \xrightarrow{k_1} \lambda_2$$
$$\lambda_2 \xrightarrow{k_2} \lambda_1$$

  where $x_0(\lambda_1) + x_0(\lambda_2) = k$

- Uniform: $\pi_{\lambda_1}(y) = \pi_{\lambda_2}(y) = \begin{cases} \frac{1}{k+1} & y \in [0,k] \\ 0 & \text{else} \end{cases}$:

$$\lambda_1 \xrightarrow{k_1} \lambda_2$$
$$\lambda_2 \xrightarrow{k_2} \lambda_1$$
$$\lambda_1 + \lambda_2 \xrightarrow{k_1} \lambda_1 + \lambda_1$$
$$\lambda_2 + \lambda_1 \xrightarrow{k_2} \lambda_2 + \lambda_2$$

  where $x_0(\lambda_1) + x_0(\lambda_2) = k$

Conclusion: This is also not a very good idea to apply to our case with phototaxis. If we try to associate the above method with the state transition of the macromolecule. We can design a pmf like:

$$f(y) = \begin{cases} 0 & y = 0 \\ \cdots & \\ 1 & y = i \quad \text{as the chosen activated state} \\ \cdots & \\ 0 & y = 7 \end{cases}$$

So there is only one designated pathway with the chosen activated state i, and we can only construct a CRN that samples from the distribution of this exact pmf which is much far too simple to align with the real situation.In this way, we cannot tell how the state transits in the cubic transitional space.

A.3 Build CRNs from HMM & EM

Hidden Markov Models (HMMs) are widely used for modeling sequential data, but their parameter learning traditionally relies on discrete, centralized algorithms such as the Baum–Welch (BW) algorithm. [13]

The authors aim to translate HMM inference and learning into biochemical reaction networks (CRNs), by designing a reaction network scheme that simulates the Baum–Welch algorithm using mass-action kinetics, allowing it to converge to the same parameter estimates as the classical EM algorithm.

A.3.1 HMM

In the following, we consider a tuple $(H, V, \theta, \psi, \pi)$ a HMM, where H is set of hidden states, V is set of visible states, $\theta : H \to H$ is the transition matrix, $\phi : H \to V$ is the emission matrix and π is an initialized prior. And we denote $\pi_h := P(h_1 = h)$, $\theta_{gh} := P(h_{l+1} = h \mid h_l = g)$, $\psi_{hv} := P(v_l = v \mid h_l = h)$.

A.3.2 Baum-Welch algorithm

BW algorithm is the EM algorithm applied to HMMs for parameter update. It consists of 4 modules: forward, backward, E-step and M-step, which I will demonstrate in detail:

1. **Forward algorithm:** we compute the joint probability of being at state h at time l while seeing $v_1, \ldots, v_l$.

   - $l = 1$:

$$\begin{aligned}\alpha_{1h} &= P(v_1, h_1 = h \mid \theta, \psi) \\ &= P(h_1 = h) \cdot P(v_1 \mid h_1 = h) \\ &= \pi_h \cdot \psi_{hv_1}\end{aligned}$$

   - $l = 2$ to L:

$$\begin{aligned}\alpha_{lh} &= P(v_1, \ldots, v_l, h_l = h \mid \theta, \psi) \\ &= \sum_g P(v_1, \ldots, v_{l-1}, h_{l-1} = g \mid \theta, \psi) \cdot P(h_l \mid h_{l-1} = g) \cdot P(v_l \mid h_l = h) \\ &= \sum_{g \in H} \alpha_{l-1g} \cdot \theta_{gh} \cdot \psi_{hv_l}\end{aligned}$$

2. **Backward algorithm:** here we compute the probability of observing the future sequence $(v_{l+1}, \ldots, v_L)$ from state h at time l.

   - $l = L$: $\beta_{Lh} = 1$
   - $l = L - 1$ to 1:

$$\begin{aligned}\beta_{lh} &= P(v_{l+1}, \ldots, v_L \mid h_l = h, \theta, \psi) \\ &= \sum_g P(v_{l+2}, \ldots, v_L \mid h_{l+1} = g, \theta, \psi) \cdot P(v_{l+1} \mid h_{l+1} = g) \cdot P(h_{l+1} = g \mid h_l = h) \\ &= \sum_g \beta_{l+1g} \cdot \psi_{gv_{l+1}} \cdot \theta_{hg}\end{aligned}$$

3. **E-step:** Same as the EM algorithm, our goal is to maximize the log-likelihood ($\max \log P(v_{1:L} \mid \theta, \psi)$). More explicitly what we want to compute is:

$$\begin{aligned}&\mathbb{E}_{P(h_{1:L} \mid v_{1:L})} \log P(h_{1:L}, v_{1:L} \mid \theta, \psi) \\ &= \mathbb{E}_{P(h_{1:L} \mid v_{1:L}, \hat{\theta}, \hat{\psi})} \log P(h_{1:L}, v_{1:L} \mid \theta, \psi) \\ &= \sum_{h_{1:L}} \log P(h_{1:L}, v_{1:L} \mid \theta, \psi) \cdot P(h_{1:L} \mid v_{1:L}, \hat{\theta}, \hat{\psi})\end{aligned}$$

   where we have:

$$\begin{aligned}\log P(h_{1:L} \mid v_{1:L}, \theta, \psi) &= \log(\pi_{h_1} \psi_{h_1 v_1} \prod_{l=2}^{L} \theta_{h_{l-1} h_l} \psi_{h_l v_l}) \\ &= \log \pi_{h_1} + \sum_{l=2}^{L} \log \theta_{h_{l-1} h_l} + \sum_{l=1}^{L} \log \psi_{h_l v_l}\end{aligned}$$

   so our original target computation becomes:

$$\mathbb{E}_{P(h_{1:L}|v_{1:L})} \log P(h_{1:L}, v_{1:L} \mid \theta, \psi)$$
$$= \sum_{h_{1:L}} (\log \pi_{h_1} + \sum_{l=2}^{L} \log \theta_{h_{l-1}h_l} + \sum_{l=1}^{L} \log \psi_{h_l v_l}) P(h_{1:L} \mid v_{1:L}, \hat{\theta}, \hat{\psi})$$

We can decompose the above formula as follows:

(a) ignore π_{h_1} since it is fixed

(b)

$$\sum_{h_{1:L}} \sum_{l=2}^{L} \log \theta_{h_{l-1}h_l} P(h_{1:l} \mid v_{1:L}, \hat{\theta}, \hat{\psi})$$
$$= \sum_{g,h} \sum_{l=1}^{L} \log \theta_{gh} P(h_l = g, h_{l+1} = h \mid v_{1:L}, \hat{\theta}, \hat{\psi})$$

where we note:

$$\begin{aligned}
\xi_{lgh} &:= P(h_l = g, h_{l+1} = h \mid v_{1:L}, \hat{\theta}, \hat{\psi}) \\
&= \frac{P(h_l = g, h_{l+1} = h, v_{1:L})}{P(v_{1:L})} \\
&= \frac{\alpha_{lg} \theta_{gh} \psi_{h v_{l+1}} \beta_{l+1h}}{\sum_f P(v_{1:l}, h_l = f) P(v_{l+1:L} \mid h_l = f)} \\
&= \frac{\alpha_{lg} \theta_{gh} \psi_{h v_{l+1}} \beta_{l+1h}}{\sum_f \alpha_{lf} \beta_{lf}}
\end{aligned}$$

(c)

$$\sum_{h_{1:L}} \sum_{l=1}^{L} \log \psi_{h_l v_l} P(h_{1:L} \mid v_{1:L}, \hat{\theta}, \hat{\psi})$$
$$= \sum_{h_{1:L}} \sum_{l=1}^{L} \log \psi_{h_l v_l} P(h_l = h \mid v_{1:L}, \hat{\theta}, \hat{\psi})$$

where we note:

$$\begin{aligned}
\gamma_{lh} &:= P(h_l = h \mid v_{1:L}) \\
&= \frac{P(v_{1:L}, h_l = h)}{P(v_{1:L})} \\
&= \frac{\alpha_{lh} \beta_{lh}}{\sum_f \alpha_{lf} \beta_{lf}}
\end{aligned}$$

4. **M-step:** After above definition of ξ_{lgh} and γ_{lh}, our original target can now be written as:

$$\max_{\theta,\psi} \sum_{l=1}^{L} \sum_{g,h} \xi_{lgh} \log \theta_{gh} + \sum_{l=1}^{L} \sum_{h} \gamma_{lh} \log \psi_{h v_l}$$

(a) θ: $\max_\theta \sum_{l=1}^{L} \sum_{g,h} \xi_{lgh} \log \theta_{gh}$ with $\sum_h \theta_{gh} = 1, \theta_{gh} \geq 0$: for a fixed g, we have Lagrange:

$$\mathcal{L}(\theta, \lambda_g) = \sum_h \sum_l^{L-1} \xi_{lgh} \log \theta_{gh} - \lambda_g (1 - \sum_h \theta_{gh})$$

by taking the derivative on θ, we have:

$$\begin{aligned} \frac{d\mathcal{L}}{d\theta_{gh}} &= \frac{\sum_l^{L-1} \xi_{lgh}}{\theta_{gh}} - \lambda_g = 0 \\ \theta_{gh} &= \frac{\sum_l^{L-1} \xi_{lgh}}{\lambda_g} \\ \text{since} \quad \sum_f \theta_{gf} &= \sum_f \frac{\sum_l \xi_{lgf}}{\lambda_g} = 1, \\ \lambda_g &= \sum_f \sum_l^{L-1} \xi_{lgf} \\ \text{so} \quad \theta_{gh} &= \frac{\sum_l^{L-1} \xi_{lgh}}{\sum_f \sum_l^{L-1} \xi_{lgf}} \end{aligned}$$

(b) ψ: $\max_\psi \sum_h \sum_{l=1}^{L} \gamma_{lh} \log \psi_{hv_l}$ with $\sum_w \psi_{hw} = 1$ and $\psi_{hw} \geq 0$: similarly for a fixed h, we have a Lagrange:

$$\mathcal{L}(\psi, \lambda_h) = \sum_{l=1}^{L} \gamma_{lh} \sum_{w \in V} \delta_{w,v_l} \log \psi_{hw} + \lambda_h (1 - \sum_w \psi_{hw})$$

by taking the derivative on ψ, we have:

$$\begin{aligned} \frac{d\mathcal{L}}{d\psi_{hw}} &= \frac{\sum_l^{L} \gamma_{lh} \delta_{w,v_l}}{\psi_{hw}} - \lambda_h = 0 \\ \psi_{hw} &= \frac{\sum_{l=1}^{L} \gamma_{lh} \delta_{w,v_l}}{\lambda_h} \\ \text{since} \quad \sum_w \psi_{hw} &= \sum_w \frac{\sum_l^{L} \gamma_{lh} \delta_{w,v_l}}{\lambda_h} = 1, \\ \lambda_h &= \sum_w \sum_{l=1}^{L} \gamma_{lh} \delta_{w,v_l} = \sum_l^{L} \gamma_{lh} \\ \text{so} \quad \psi_{hw} &= \frac{\sum_l^{L} \gamma_{lh} \delta_{w,v_l}}{\sum_l^{L} \gamma_{lh}} \end{aligned}$$

A.3.3 BW reaction network

We consider the above variables computed in BW algorithm are species to model the above computations into subnetworks of reactions.

1. **Forward:**

$$\alpha_{1,h} = \pi_h \psi_{h,v_1} \quad \text{implies that} \quad \alpha_{1,h}\pi_{h^*}\psi_{h^*,v_1} = \alpha_{1,h^*}\pi_h\psi_{h,v_1} \quad \text{for any} \quad h^* \in H$$

From this equation we can build 2 reactions that indicates the transition between $\alpha_{1,h} \rightleftharpoons \alpha_{1,h^*}$:

$$\alpha_{1,h} + \pi_{h^*} + \psi_{h^*,v_1} \to \alpha_{1,h^*} + \pi_{h^*} + \psi_{h^*,v_1}$$
$$\alpha_{1,h^*} + \pi_h + \psi_{h,v_1} \to \alpha_{1,h} + \pi_h + \psi_{h,v_1}$$

And we can derive the ODE from the above reactions for the target $\alpha_{1,h}$:

$$\frac{d[\alpha_{1,h}]}{dt} = [\alpha_{1,h^*}] \cdot [\pi_h] \cdot [\psi_{h,v_1}] - [\alpha_{1,h}] \cdot [\pi_{h^*}] \cdot [\psi_{h^*,v_1}]$$

and at equilibrium we can get the exact equation implied above.

Same with $l = 1, \ldots, L-1$, we have

$$\alpha_{l+1,h} = \sum_g \alpha_{l,g}\theta_{g,h}\psi_{h,v_{l+1}} \quad \text{implies that}$$

$$\alpha_{l+1,h} \sum_g \alpha_{l,g}\theta_{g,h^*}\psi_{h^*,v_{l+1}} = \alpha_{l+1,h^*} \sum_g \alpha_{l,g}\theta_{g,h}\psi_{h,v_{l+1}} \quad \text{for any} \quad h^* \in H$$

And similarly we can build reactions from transition between $\alpha_{l+1,h} \rightleftharpoons \alpha_{l+1,h^*}$:

$$\alpha_{l+1,h} + \alpha_{l,g} + \theta_{g,h^*} + \psi_{h^*,v_{l+1}} \to \alpha_{1+1,h^*} + \alpha_{l,g} + \theta_{g,h^*} + \psi_{h^*,v_{l+1}}$$
$$\alpha_{1,h^*} + \pi_h + \psi_{h,v_1} \to \alpha_{1,h} + \pi_h + \psi_{h,v_1}$$

we can also build ODE from the above reactions for the target $\alpha_{l+1,h}$:

$$\frac{d[\alpha_{l+1,h}]}{dt} = \sum_g [\alpha_{l+1,h^*}] \cdot [\alpha_{l,g}] \cdot [\theta_{g,h}] \cdot [\psi_{h,v_l}] - \sum_g [\alpha_{l+1,h}] \cdot [\alpha_{l,g}] \cdot [\theta_{g,h^*}] \cdot [\psi_{h^*,v_l}]$$

same as before we can get the implied equation at equilibrium.

The problem here is that since ψ_{h,v_l} depends on v_l, with different values of v_l, we have to design different corresponding reactions. So to simplify the process, we introduce another species $E_{l,w}$ as another catalyzer of the reactions, initialized with:

$$E_{l,w}(0) = \begin{cases} 1 & w = v_l \quad \text{(reaction activated iff} \quad w = v_l) \\ 0 & \text{else} \end{cases}$$

so that the reaction rate of the target will depend on the concentration of $E_{l,w}$. And this species will also function in the other modules. Therefore, we can construct the subnetworks $\mathcal{R}_1^\alpha$ and $\mathcal{R}_l^\alpha$ as follows:

$$\mathcal{R}_1^\alpha: \quad \alpha_{1,h} + \pi_{h^*} + \psi_{h^*,v_1} + E_{1,w} \to \alpha_{1,h^*} + \pi_{h^*} + \psi_{h^*,v_1} + E_{1,w}$$
$$\alpha_{1,h^*} + \pi_h + \psi_{h,v_1} + E_{1,w} \to \alpha_{1,h} + \pi_h + \psi_{h,v_1} + E_{1,w}$$
$$\mathcal{R}_l^\alpha: \quad \alpha_{l+1,h} + \alpha_{l,g} + \theta_{g,h^*} + \psi_{h^*,v_{l+1}} + E_{l+1,w} \to \alpha_{1+1,h^*} + \alpha_{l,g} + \theta_{g,h^*} + \psi_{h^*,v_{l+1}} + E_{l+1,w}$$
$$\alpha_{1,h^*} + \pi_h + \psi_{h,v_1} + E_{l+1,w} \to \alpha_{1,h} + \pi_h + \psi_{h,v_1} + E_{l+1,w}$$

2. **Backward:**

Since $\beta_{L,h} = 1$ is constant, so there is no need to consider it as a species for reactions. For $l = L-1, \ldots, 1$:,

$$\beta_{l,h} = \sum_{g \in H} \theta_{h,g} \psi_{g,v_{l+1}} \beta_{l+1,g}$$

$$\text{implies that } \beta_{l,h} \sum_{g \in H} \theta_{h^*,g} \psi_{g,v_{l+1}} \beta_{l+1,g} = \beta_{l,h^*} \sum_{g \in H} \theta_{h,g} \psi_{g,v_{l+1}} \beta_{l+1,g}$$

With the same idea we can build a subnetwork $\mathcal{R}_l^\beta$:

$$\begin{aligned} \mathcal{R}_l^\beta : \quad & \beta_{l,h} + \theta_{h^*,g} + \psi_{g,w} + \beta_{l+1,g} + E_{l+1,w} \to \beta_{l,h^*} + \theta_{h^*,g} + \psi_{g,w} + \beta_{l+1,g} + E_{l+1,w} \\ & \beta_{l,h^*} + \theta_{h,g} + \psi_{g,w} + \beta_{l+1,g} + E_{l+1,w} \to \beta_{l,h} + \theta_{h,g} + \psi_{g,w} + \beta_{l+1,g} + E_{l+1,w} \end{aligned}$$

And in the same way we can get the ODE system like before:

$$\begin{cases} \frac{d[\beta_{L,h}]}{dt} = 0 \\ \frac{d[\beta_{l,h}]}{dt} = [\beta_{l,h^*}] \cdot \sum_g [\beta_{l+1,g}] \cdot [\theta_{h,g}] \cdot [\psi_{g,v_l}] - [\beta_{l,h}] \cdot \sum_g [\beta_{l+1,g}] \cdot [\theta_{h^*,g}] \cdot \psi_{g,v_l} \end{cases}$$

3. **E-step:**

For $l = 1, \ldots, L$, we have:

$$\gamma_{l,h} = \frac{\alpha_{l,h}\beta_{l,h}}{\sum_f \alpha_{l,f}\beta_{l,f}} \quad \text{implies that} \quad \gamma_{l,h}\alpha_{l,h^*}\beta_{l,h^*} = \gamma_{l,h^*}\alpha_{l,h}\beta_{l,h}$$

and we can directly get subnetwork $\mathcal{R}_l^\gamma$:

$$\begin{aligned} \mathcal{R}_l^\gamma : \quad & \gamma_{l,h} + \alpha_{l,h^*} + \beta_{l,h^*} \to \gamma_{l,h^*} + \alpha_{l,h^*} + \beta_{l,h^*} \\ & \gamma_{l,h^*} + \alpha_{l,h} + \beta_{l,h} \to \gamma_{l,h} + \alpha_{l,h} + \beta_{l,h} \end{aligned}$$

As for the corresponding ODE, we have:

$$\frac{d[\gamma_{l,h}]}{dt} = [\gamma_{l,h^*}] \cdot [\alpha_{l,h}] \cdot [\beta_{l,h}] - [\gamma_{l,h}] \cdot [\alpha_{l,h^*}] \cdot [\beta_{l,h^*}]$$

And for $l = 1, \ldots, L-1$, we have:

$$\xi_{l,g,h} = \frac{\alpha_{l,g}\theta_{g,h}\psi_{h,v_{l+1}}\beta_{l+1,h}}{\sum_f \alpha_{l,f}\beta_{l,f}}$$

$$\text{implies that} \quad \xi_{l,g,h}\alpha_{l,h^*}\theta_{h^*,h^*}\psi_{h^*,v_{l+1}}\beta_{l+1,h^*} = \xi_{l,h^*,h^*}\alpha_{l,g}\theta_{g,h}\psi_{h,v_{l+1}}\beta_{l+1,h}$$

from which we have subnetwork $\mathcal{R}_l^\xi$:

$$\begin{aligned} \mathcal{R}_l^\xi : \quad & \xi_{lgh} + \alpha_{l,h^*} + \theta_{h^*h^*} + \psi_{h^*w} + \beta_{l+1,h^*} + E_{l+1,w} \to \xi_{lh^*h^*} + \alpha_{l,h^*} + \theta_{h^*h^*} + \psi_{h^*w} + \beta_{l+1,h^*} + E_{l+1} \\ & \xi_{l,h^*,h^*} + \alpha_{l,g} + \theta_{g,h} + \psi_{h,w} + \beta_{l+1,h} + E_{l+1,w} \to \xi_{l,g,h} + \alpha_{l,g} + \theta_{g,h} + \psi_{h,w} + \beta_{l+1,h} + E_{l+1,w} \end{aligned}$$

And we obtain the ODE the same way as before:

$$\frac{d[\xi_{l,g,h}]}{dt} = [\xi_{l,h^*,h^*}] \cdot [\alpha_{l,g}] \cdot [\theta_{g,h}] \cdot [\psi_{h,v_{l+1}}] \cdot [\beta_{l+1,h}] - [\xi_{l,g,h}] \cdot [\alpha_{l,h^*}] \cdot [\theta_{h^*,h^*}] \cdot [\psi_{h^*,v_{l+1}}] \cdot [\beta_{l+1,h^*}]$$

4. **M-step:**

$$\theta'_{g,h} = \frac{\sum_l^{L-1} \xi_{l,g,h}}{\sum_l^{L-1}\sum_f \xi_{lgf}} \quad \text{implies that} \quad \theta'_{g,h}\sum_l^{L-1}\xi_{l,g,h^*} = \theta'_{g,h^*}\sum_l^{L-1}\xi_{l,g,h}$$

and we have the subnetwork $\mathcal{R}^\theta$:

$$\begin{aligned}\mathcal{R}^\theta: \quad & \theta'_{g,h} + \xi_{l,g,h^*} \to \theta'_{g,h} + \xi_{l,g,h^*} \\ & \theta'_{g,h^*} + \xi_{l,g,h} \to \theta'_{g,h} + \xi_{l,g,h}\end{aligned}$$

$$\text{ODE}: \quad \frac{d\theta'_{g,h}}{dt} = [\theta'_{g,h^*}]\cdot\sum_l^{L-1}[\xi_{l,g,h}] - [\theta'_{g,h}]\cdot\sum_l^{L-1}[\xi_{l,g,h^*}]$$

And finally we have:

$$\psi_{h,w} = \frac{\sum_l^L \gamma_{l,h}\delta_{w,v_l}}{\sum_l^L \gamma_{l,h}} \quad \text{implies that} \quad \psi_{h,w}\sum_l \gamma_{l,h}\delta_{v^*,v_l} = \psi_{h,v^*}\sum_l \gamma_{l,h}\delta_{w,v_l}$$

from which we have the final subnetwork $\mathcal{R}^\psi$:

$$\begin{aligned}\mathcal{R}^\psi: \quad & \psi_{h,w} + \gamma_{l,h} + E_{l,v^*} \to \psi_{h,v^*} + \gamma_{l,h} + E_{l,v^*} \\ & \psi_{h,v^*} + \gamma_{l,h} + E_{l,w} \to \psi_{h,w} + \gamma_{l,h} + E_{l,w}\end{aligned}$$

$$\text{ODE:} \quad \frac{d\psi_{h,w}}{dt} = [\psi_{h,v^*}]\cdot\sum_l^L[\gamma_{l,h}]\cdot[E_{l,w}] - [\psi_{h,w}]\cdot\sum_l^L[\gamma_{l,h}]\cdot[E_{l,v^*}]$$

A.3.4 Alternative BW

In the paper of [13], the authors proposed a reformulated computation of the BW algorithm that exposes the relationship to the dynamics of BW reaction network built above.

We can take the first subnetwork $\mathcal{R}_1^\alpha$ as an instance:

$$\begin{aligned}\text{ODE:} \quad \frac{d[\alpha_{1,h}]}{dt} &= [\alpha_{1,h^*}]\cdot[\pi_h]\cdot[\psi_{h,v_1}] - [\alpha_{1,h}]\cdot[\pi_{h^*}]\cdot[\psi_{h^*,v_1}] \\ &= -\frac{d[\alpha_{1,h^*}]}{dt} \quad \text{for any fixed } h^* \neq h \in H \text{ by symmetry}\end{aligned}$$

so we have:

$$\begin{aligned}\dot{\alpha}_{1,h^*} &= -\sum_{h\neq h^*}\dot{\alpha}_{1,h} \\ \sum_h \dot{\alpha}_{1,h} &= \sum_{h\neq h^*}\dot{\alpha}_{1,h} + \dot{\alpha}_{1,h} = 0 \\ \sum_h \alpha_{1,h} &= A_1 \quad \text{where } A_1 \text{ is a constant}\end{aligned}$$

$$\text{And similarly we have:} \quad \sum_h \alpha_{l,h} = A_l \quad \text{for } l = 2,\cdots,L$$

And when the ODE is at equilibrium, we can get the following equation:

$$\begin{aligned}
\sum_{h^*\in H}\alpha_{l,h^*}\sum_g \alpha_{l-1.g}\theta_{g,h}\psi_{h,v_l} &= \sum_{h^*}\alpha_{l,h}\sum_g \alpha_{l-1.g}\theta_{g,h^*}\psi_{h^*,v_l} \\
\alpha_{l,h} &= \frac{A_l\sum_g \alpha_{l-1,g}\theta_{g,h}\psi_{h,v_l}}{\sum_{h^*}\sum_g \alpha_{l-1.g}\theta_{g,h^*}\psi_{h^*,v_l}} \\
&= \frac{A_l\sum_g \alpha_{l-1,g}\theta_{g,h}\psi_{h,v_l}}{\sum_f\sum_g \alpha_{l-1.f}\theta_{f,g}\psi_{g,v_l}} \\
&= \frac{A_l\alpha_{l,h}}{\sum_g \alpha_{l,g}} \\
&:= \hat{\alpha}_{l,h} \quad \text{for } l = 2,\cdots,L
\end{aligned}$$

and similarly we have: $\hat{\alpha}_{1,h} = \dfrac{A_1\alpha_{1,h}}{\sum_g \alpha_{1,g}} = \dfrac{A_1\pi_h\psi_{h,v_1}}{\sum_g \pi_g\psi_{g,v_1}}$

We can do the same reformulation for the backward, E-step and M-step modules to get the full reformulated version of BW algorithm as follows:

1. **Forward algorithm:**
   - **Initialisation:**
   $$\hat{\alpha}_{1h} = \frac{A_1\pi_h\psi_{hv_1}}{\sum_{g\in H}\pi_g\psi_{gv_1}}, \quad \text{for all } h\in H$$
   - **Recursion:**
   $$\hat{\alpha}_{\ell h} = \frac{A_\ell\sum_{g\in H}\hat{\alpha}_{\ell-1,g}\theta_{gh}\psi_{hv_\ell}}{\sum_{f\in H}\sum_{g\in H}\hat{\alpha}_{\ell-1,f}\theta_{fg}\psi_{gv_\ell}}, \quad \text{for all } h\in H,\ \ell = 2,\ldots,L$$
2. **Backward algorithm:**
   - **Initialisation:**
   $$\hat{\beta}_{Lh} = \frac{B_L}{\#H}, \quad \text{for all } h\in H$$
   - **Recursion:**
   $$\hat{\beta}_{\ell h} = \frac{B_\ell\sum_{g\in H}\theta_{hg}\psi_{gv_{\ell+1}}\hat{\beta}_{\ell+1,g}}{\sum_{f\in H}\sum_{g\in H}\theta_{fg}\psi_{gv_{\ell+1}}\hat{\beta}_{\ell+1,g}}, \quad \text{for all } h\in H,\ \ell = 1,\ldots,L-1$$
3. **E-step :**

$$\hat{\xi}_{\ell gh} = \frac{\Xi_\ell\hat{\alpha}_{\ell g}\theta_{gh}\psi_{hv_{\ell+1}}\hat{\beta}_{\ell+1,h}}{\sum_{g\in H}\sum_{h\in H}\hat{\alpha}_{\ell g}\theta_{gh}\psi_{hv_{\ell+1}}\hat{\beta}_{\ell+1,h}} = \frac{\Xi_\ell\hat{\alpha}_{\ell g}\theta_{gh}\psi_{hv_{\ell+1}}\hat{\beta}_{\ell+1,h}}{\sum_{f\in H}\hat{\alpha}_{\ell f}\hat{\beta}_{\ell f}}$$

$$\hat{\gamma}_{\ell h} = \frac{\Gamma_\ell\hat{\alpha}_{\ell h}\hat{\beta}_{\ell h}}{\sum_{f\in H}\hat{\alpha}_{\ell f}\hat{\beta}_{\ell f}}$$

4. **M-step :**

$$\theta'_{gh} = \frac{\sum_{\ell=1}^{L-1} \frac{1}{\Xi_\ell} \hat{\xi}_{\ell gh}}{\sum_{\ell=1}^{L-1} \sum_{f \in H} \frac{1}{\Xi_\ell} \hat{\xi}_{\ell gf}} = \frac{\sum_{\ell=1}^{L-1} \hat{\alpha}_{\ell g} \theta_{gh} \psi_{h v_{\ell+1}} \hat{\beta}_{\ell+1,h}}{\sum_{\ell=1}^{L} \hat{\alpha}_{\ell g} \hat{\beta}_{\ell g}}$$

$$\psi'_{hw} = \frac{\sum_{\ell=1}^{L} \frac{1}{\Gamma_\ell} \hat{\gamma}_{\ell h} \delta_{w,v_\ell}}{\sum_{\ell=1}^{L} \frac{1}{\Gamma_\ell} \hat{\gamma}_{\ell h}} = \frac{\sum_{\ell=1}^{L} \hat{\alpha}_{\ell h} \hat{\beta}_{\ell h} \delta_{w,v_\ell}}{\sum_{\ell=1}^{L} \hat{\alpha}_{\ell h} \hat{\beta}_{\ell h}}$$

In the above reformulated computation, we have the following normalization:

$$\sum_{h \in H} \hat{\alpha}_{\ell h} = A_\ell, \quad \sum_{h \in H} \hat{\beta}_{\ell h} = B_\ell, \quad \sum_{g,h \in H} \hat{\xi}_{\ell gh} = \Xi_\ell = 1, \quad \sum_{h \in H} \hat{\gamma}_{\ell h} = \Gamma_\ell = 1,$$

$$\sum_{h \in H} \theta'_{gh} = 1, \quad \sum_{w \in V} \psi'_{hw} = 1$$

A.3.5 Dynamics of BW reaction network:

There are also three exection models for the reaction network introduced in the paper. The first scheduler BW1 execute all the subnetworks sequentially in the order of $\mathcal{R}_1^\alpha, \cdots, \mathcal{R}_L^\alpha, \mathcal{R}_{L-1}^\beta, \cdots, \mathcal{R}_1^\beta, \mathcal{R}_1^\gamma, \cdots, \mathcal{R}_L^\gamma, \mathcal{R}_1^\xi, \cdots, \mathcal{R}_{L-1}^\xi, \mathcal{R}^\theta, \mathcal{R}^\psi$, so in each iteration there is $4L\times$ subnetworks that reaches the equilibrium and the parameters θ' and ψ' are only updated once after each iteration.

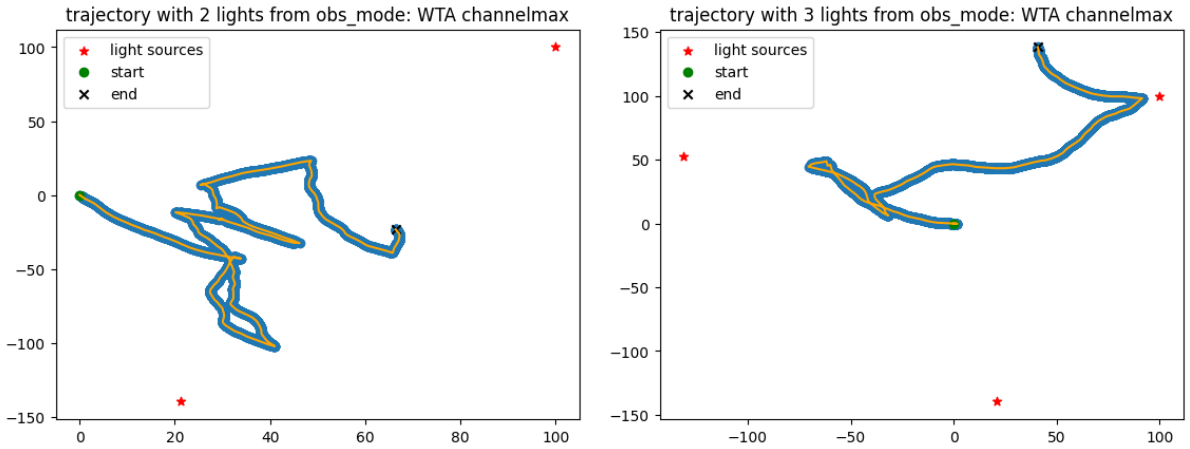
As for scheduler BW2, two modules: inference module (forward + backward + E-step) and learning module (M-step) are executed sequentially in each iteration.

Our research follows the BW3 as the experiments done in the paper. where all subnetworks are executed altogether and we solve the complete ODE system with all four sections of algorithms.

B. Supplementary multi-source experiments

B.1 Channelmax

We recall that channelmax method favors lateral stimulation, encouraging turning to one side over the other.



(a) 2 lights - WTA channelmax

(b) 3 lights - WTA channelmax

Figure B.1: Multi-source with weighted average observation

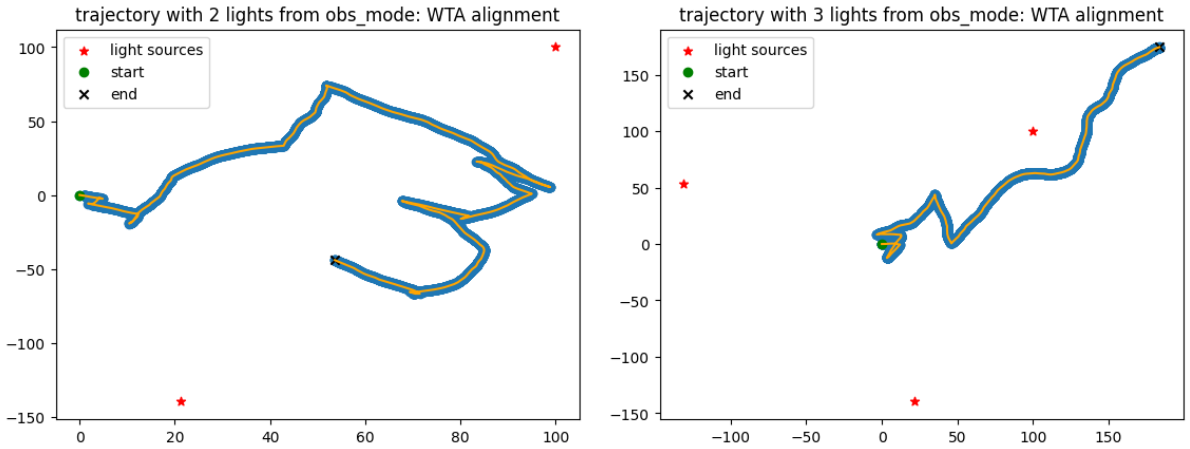
When there are two lights, we can see a frequently changing preference of a single winner that WTA channelmax rapidly commits, And with 3 lights, the agent hovers around the center at first, with trouble determining the winner, then almost deterministically moves towards the decided winner.

B.2 Alignment

We recall that this mode of WTA favors more the alignment between the heading of the agent and direction to one of the lights as winner.

We can observe from the above figure that WTA-alignment oscillates between two alignment peaks as the winner alters frequently than the previous method. When there are 3 lights, the WTA alignment presents similar pattern as the agent still hovers around the center at first, then choose the winner once it decides the best alignment with one of the light source and moves forward.

To conclude:WTA preserves single-source statistics but may oscillate around the center with multiple lights; Weighted observation aggregates cues and follows the resultant vector, yielding more stable drift with less oscilliation.



(a) 2 lights - WTA alignment (b) 3 lights - WTA alignment

Figure B.2: Multi-source with WTA alignment

B.3 Flickering light

Flickering lights model: 2-State Continuous Time Markov Chain (CTMC) Since in the real biological scenarios, the lights are flickering in the phototaxis experiments.

Here we model each light k as a **2-state CTMC** with states $Z_k(t) \in \{0, 1\}$, where $Z_k(t) = 1$ means **ON** and $Z_k(t) = 0$ means **OFF** . And we define $p_k(t) := \Pr\{\mathrm{Z_k(t) = 1}\}$

We set that while light is **ON**, the waiting time to leave the state follows an exponential distribution:

$$T_{\text{on}} \sim \text{Exp}(\alpha_{c,k}), \qquad \mathbb{E}[T_{\text{on}}] = \tau_{\text{on},k}, \qquad \alpha_{c,k} = \frac{1}{\tau_{\text{on},k}}.$$

and similarly for **OFF**:

$$T_{\text{off}} \sim \text{Exp}(\beta_{c,k}), \qquad \mathbb{E}[T_{\text{off}}] = \tau_{\text{off},k}, \qquad \beta_{c,k} = \frac{1}{\tau_{\text{off},k}}.$$

And we can build generator matrix encoding the transition rates between two states:

$$Q_k = \begin{bmatrix} -\alpha_{c,k} & \alpha_{c,k} \\ \beta_{c,k} & -\beta_{c,k} \end{bmatrix}.$$

For an exact Δt-step, the transition is defined according to Kolmogorov equations that:

$$P_k(\Delta t) = \exp(Q_k \Delta t) = \begin{bmatrix} \pi_{\text{on,k}} + \pi_{\text{off,k}} e^{-\lambda_k \Delta t} & \pi_{\text{off,k}}(1 - e^{-\lambda_k \Delta t}) \\ \pi_{\text{on,k}}(1 - e^{-\lambda_k \Delta t}) & \pi_{\text{off,k}} + \pi_{\text{on,k}} e^{-\lambda_k \Delta t} \end{bmatrix}.$$

where $\lambda_k = \alpha_{c,k} + \beta_{c,k}$, $\pi_{\text{on,k}} = \frac{\beta_{c,k}}{\lambda_k}$, $\pi_{\text{off,k}} = 1 - \pi_{\text{on,k}}$

And when Δt is small enough, we can consider leaving one state for another as Bernoulli events with flip rates α_k, β_k:

$$\alpha_k = P(T_{\text{on}} \leq \Delta t) = 1 - e^{-\Delta t/\tau_{\text{on,k}}}, \qquad \beta_k = P(T_{\text{off}} \leq \Delta t) = 1 - e^{-\Delta t/\tau_{\text{off,k}}}.$$

And as we assume that the observation of the agent does not affect the state of light in return, we propose prior update for light is **ON** as:

$$p_k(t+\Delta t) = p_k(t)(1 - \alpha_k) + \big(1 - p_k(t)\big)\beta_k.$$

In the flickering lights scenario, we also implement the WTA to obtain the observation. And here we weight the score by prior **ON** probability $p_k(t)$

$$D_k^{\text{align}} = \max\big(0,\, \hat{v}^\top r^k\big).$$

$$S_k(t) = p_k(t) \cdot a(s_k) \cdot D_k,$$

With the flickering lights scenario we provide an example with 3 flickering lights under WTA-alignment mode.

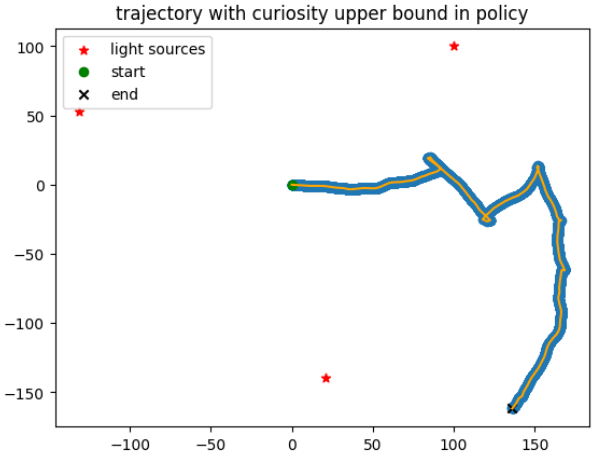


Figure B.3: flickering lights trajectory

Under WTA-alignment with CTMC flickers, the tumbling ratio rises (0.06→0.19). Since S_k depends only on alignment and distance, the winner can be OFF; the resulting near-zero observation increases posterior uncertainty and boosts the curiosity term, leading to more tumbles. ON/OFF intermittency prevents sustained evidence toward a single source, hence no persistent bias—qualitatively similar to the unbiased resultant-vector behavior of weighted-obs.