Experiments 3 - Lake 3

Lake problem describes a situation where inhabitants of a town intend to increase their economic benefits through developing industry and agriculture. These activities will also emit pollution into a lake nearby. Although the lake pollution decreases continuously due to natural removal, once it exceeds a threshold X_{crit}, irreversible lake eutrophication would occur and cause huge losses. This problem requires decision-makers to decide the annual pollution emissions, with the aim of maximizing the economic benefit while avoiding eutrophication. The lake pollution transition function is given by:
\begin{equation}
X_{(t+1)}=X_t+a_t+\frac{(X_t^q)}{(1+X_t^q)}- bX_t+\epsilon_t
\end{equation}
where X_t is the pollution at time t, and $X_0 = 0$; a_t is the rate of anthropogenic pollution at time t; b is the lake's natural removal rate; q is the lake's natural recycling rate; ϵ_t is the rate of natural pollution at time t. The natural pollution ϵ_t is modeled as a log normal distribution with mean μ and variance σ^2. The threshold X_{crit} is the pollution level at which the lake's natural recycling equals its natural removal rate. The total benefit from the pollution is given by
\begin{equation}
f_{economic} = \sum\limits_{t \in {T}}\alpha a_t \delta^t
\end{equation}
where α is the benefit-to-pollution ratio, and δ is the discount rate.

The lake problem is characterized by both stochastic uncertainty and deep uncertainty. The stochastic uncertainty arises from the natural inflow. Deep uncertainty is presented by uncertainty about the mean μ and standard deviation $sigma$ of the lognormal distribution characterizing the natural inflow, the natural removal rate of the lake β, the natural recycling rate of the lake q, and the discount rate δ.

Please formalize this problem description as a model. Please also specify key components, such as state variables, stochastic variables, constant parameters, uncertainties and transition functions.

Excellent — you're describing a **stochastic dynamic decision problem** (often formulated as a Markov Decision Process, or MDP, under uncertainty) with both **stochastic** and **deep uncertainties**. Below is a rigorous formalization of the *Lake Problem* model consistent with decision-analytic and dynamic optimization frameworks.

Formal Model: The Lake Problem

1. Time Horizon

- $t \in \{0, 1, 2, \ldots, T\}$: discrete time periods.

---

2. State Variable

- $X_t \in \mathbb{R}_{\geq 0}$: Pollution level in the lake at time t.
  - Initial condition: $X_0 = 0$.

---

3. Decision (Control) Variable

- $a_t \in [0, a_{\max}]$: Anthropogenic (human-caused) pollution emissions at time t.
  - This is the policy variable the decision-maker chooses to maximize long-term welfare.

---

4. Stochastic Variable

- ϵ_t: Natural (exogenous) pollution inflow at time t.
  - Modeled as a **lognormal** random variable:

$$\epsilon_t \sim \mathrm{LogNormal}(\mu, \sigma^2)$$

  - Represents **stochastic uncertainty** due to random natural inflows.

---

5. System Dynamics (Transition Function)

The lake's pollution evolves according to:

$$X_{t+1} = X_t + a_t + \frac{X_t^q}{1 + X_t^q} - bX_t + \epsilon_t$$

- $\frac{X_t^q}{1+X_t^q}$: nonlinear feedback from nutrient recycling within the lake (self-reinforcing eutrophication dynamics).
- $-bX_t$: natural removal (self-cleaning) process.
- a_t: anthropogenic emissions.
- ϵ_t: stochastic natural inflow.

---

6. Threshold Condition (Irreversible Damage)

- X_{crit}: critical pollution level beyond which the lake becomes irreversibly eutrophic.
- Defined implicitly by:

$$\frac{X_{\text{crit}}^q}{1 + X_{\text{crit}}^q} = bX_{\text{crit}}$$

- Constraint (soft or hard depending on formulation):

$$X_t < X_{\text{crit}} \quad \forall t$$

  Violating this leads to catastrophic losses.

7. Objective Function

Maximize discounted economic benefits from pollution emissions:

$$\max_{\{a_t\}} \; f_{\text{economic}} = \sum_{t=0}^{T} \alpha a_t \delta^t$$

subject to the pollution dynamics and the eutrophication constraint.

8. Parameters

Symbol	Description	Nature
α	Benefit per unit of pollution (benefit-to-pollution ratio)	constant
b	Natural removal rate of the lake	uncertain constant
q	Natural recycling exponent (nonlinear feedback strength)	uncertain constant
μ	Mean of lognormal natural inflow	uncertain constant
σ	Standard deviation of lognormal inflow	uncertain constant
δ	Discount rate	uncertain constant
X_{crit}	Critical pollution threshold	derived constant

9. Uncertainty Structure

9.1 Stochastic Uncertainty (Aleatory)

- Random year-to-year variability:
  - $\epsilon_t \sim \text{LogNormal}(\mu, \sigma^2)$
  - Captures *short-term natural variability*.

9.2 Deep Uncertainty (Epistemic)

- Uncertainty in model parameters themselves:

$$\Theta = \{\mu, \sigma, b, q, \delta\}$$

- Their true values are *unknown* or *controversial*.
- Represented by:
  - Multiple plausible scenarios (robust decision-making), or
  - Probability distributions (Bayesian approach), or
  - Unknown ranges (info-gap or exploratory modeling).

10. Formal Optimization Problem

$$
\begin{aligned}
\max_{\{a_t\}} \quad & \mathbb{E}\left[\sum_{t=0}^{T} \alpha a_t \delta^t\right] \\
\text{s.t.} \quad & X_{t+1} = X_t + a_t + \frac{X_t^q}{1+X_t^q} - bX_t + \epsilon_t, \quad \forall t \\
& X_0 = 0, \\
& X_t < X_{\text{crit}} \quad \forall t, \\
& \epsilon_t \sim \text{LogNormal}(\mu, \sigma^2), \\
& a_t \geq 0.
\end{aligned}
$$

Depending on modeling philosophy:

- **Stochastic Dynamic Programming (SDP)** formulation → expectation over ϵ_t.
- **Robust Optimization or Exploratory Modeling** → maximin or multi-objective over uncertain parameters Θ.

---

11. Model Classification

Aspect	Description
Type	Stochastic Dynamic Decision Model
Uncertainty	Mixed (stochastic + deep uncertainty)
Objective	Economic benefit maximization
Dynamics	Nonlinear pollution accumulation
Risk	Threshold-based irreversible loss (catastrophic regime shift)

---

12. Possible Decision Frameworks

Depending on how uncertainty is treated:

- **Expected Value Optimization** (classic SDP)
- **Robust Decision-Making (RDM)**: maximize robustness under uncertain Θ
- **Dynamic Adaptive Policy Pathways (DAPP)**: define adaptive strategies as uncertainty resolves
- **Info-Gap Decision Theory (IGDT)**: maximize resilience to deep uncertainty in Θ

Would you like me to **express this model explicitly as a dynamic programming formulation** (i.e., Bellman equation form)? That's the next formal step in implementing or solving it computationally.

X_{crit} is not a constant because it depends on other deeply uncertain parameters. Please refine your model specification.

Excellent point — you're absolutely right.
The **critical pollution threshold** X_{crit} is *not fixed*; it *emerges endogenously* from the interaction between the **lake's natural recycling rate** q and the **natural removal rate** b, both of which are deeply uncertain. Let's refine the model accordingly.

🧭 Refined Formalization of the *Lake Problem* (with endogenous X_{crit})

---

1. Time Horizon

$$t \in \{0, 1, 2, \ldots, T\}$$

---

2. State Variable

$$X_t \in \mathbb{R}_{\geq 0} : \text{pollution stock in the lake at time } t$$

- Initial condition: $X_0 = 0$.

---

3. Decision (Control) Variable

$$a_t \in [0, a_{\max}] : \text{anthropogenic pollution emissions at time } t$$

Chosen by decision-makers to balance economic benefit and environmental risk.

---

4. Stochastic Variable

$$\epsilon_t \sim \text{LogNormal}(\mu, \sigma^2)$$

- Represents random natural pollution inflows.
- **Stochastic (aleatory) uncertainty** — year-to-year variability.

---

5. System Dynamics (Transition Function)

$$X_{t+1} = X_t + a_t + \frac{X_t^q}{1 + X_t^q} - bX_t + \epsilon_t$$

where:

- $\frac{X_t^q}{1+X_t^q}$: endogenous nutrient recycling feedback,
- $-bX_t$: natural removal,
- a_t: controllable anthropogenic inflow,
- ϵ_t: stochastic natural inflow.

---

6. Critical Threshold as an Endogenous Quantity

The critical pollution threshold X_{crit} is determined implicitly by the equilibrium between recycling and removal:

$$\frac{X_{\text{crit}}^q}{1 + X_{\text{crit}}^q} = bX_{\text{crit}}$$

That is:

$$X_{\text{crit}} = X_{\text{crit}}(b, q)$$

- X_{crit} **depends on deeply uncertain parameters** b and q.
- Different plausible combinations of b and q yield different lake regimes (e.g., more fragile or more resilient systems).
- Crossing $X_{\text{crit}}(b, q)$ implies a *regime shift* to irreversible eutrophication.

Thus, for each realization of the deep uncertainty vector Θ, the effective constraint becomes:

$$X_t < X_{\text{crit}}(b, q)$$

---

7. Objective Function

Maximize the discounted economic benefit from emissions:

$$\max_{\{a_t\}} \; f_{\text{economic}} = \sum_{t=0}^{T} \alpha a_t \, \delta^t$$

where:

- α: benefit-to-pollution ratio,
- δ: (uncertain) discount factor, reflecting time preference.

---

8. Parameter and Uncertainty Structure

Symbol	Description	Type of Uncertainty	Nature
μ	Mean of lognormal natural inflow	Deep	Epistemic
σ	Standard deviation of lognormal inflow	Deep	Epistemic
b	Natural removal rate of the lake	Deep	Epistemic
q	Natural recycling rate exponent	Deep	Epistemic
δ	Discount rate	Deep	Epistemic
α	Benefit-to-pollution ratio	Assumed constant	Parameter
ϵ_t	Natural stochastic inflow	Stochastic	Aleatory

Define the **deep uncertainty vector**:

$$\Theta = \{\mu, \sigma, b, q, \delta\}$$

Each Θ defines a **possible world** (or model configuration) with its own system behavior and critical threshold $X_{\text{crit}}(\Theta)$.

9. Revised System of Equations

$$\begin{cases} X_{t+1} = X_t + a_t + \dfrac{X_t^q}{1 + X_t^q} - bX_t + \epsilon_t, & \text{(pollution dynamics)} \\ \epsilon_t \sim \text{LogNormal}(\mu, \sigma^2), & \text{(stochastic inflow)} \\ X_{\text{crit}} = \text{solution of } \dfrac{X_{\text{crit}}^q}{1 + X_{\text{crit}}^q} - bX_{\text{crit}} = 0, & \text{(critical threshold)} \\ a_t \geq 0, \; X_t < X_{\text{crit}}(b, q). & \text{(constraints)} \end{cases}$$

10. Optimization Problem Under Uncertainty

$$\begin{aligned} \max_{\{a_t\}} \quad & \mathbb{E}_{\epsilon_t}\left[\sum_{t=0}^{T} \alpha a_t \delta^t\right] \\ \text{s.t.} \quad & X_{t+1} = X_t + a_t + \frac{X_t^q}{1 + X_t^q} - bX_t + \epsilon_t, \\ & X_t < X_{\text{crit}}(b, q), \\ & X_0 = 0, \\ & a_t \geq 0, \\ & \Theta = \{\mu, \sigma, b, q, \delta\} \text{ (deeply uncertain)}. \end{aligned}$$

11. Decision Frameworks for Deep Uncertainty

Because Θ is not known probabilistically, several modeling approaches are possible:

Framework	Formulation	Description
Robust Optimization	$\max_{a_t} \min_{\Theta \in \mathcal{U}} f_{\text{economic}}(\Theta)$	Optimize against worst-case parameters within uncertainty set $\mathcal{U}$.
Exploratory Modeling / RDM	Explore f_{economic} over many sampled Θ	Identify strategies that perform satisfactorily across plausible futures.
Info-Gap Theory	$\max_{a_t} \; \hat{f}_{\text{economic}}(a_t)$ s.t. robustness margin	Maximize robustness to model misspecification.
Bayesian Decision-Making	Introduce $p(\Theta)$ and maximize expected utility	Requires belief distributions over deep uncertainties.

12. Model Summary

Category	Symbol / Function	Description
State variable	X_t	Lake pollution stock
Control variable	a_t	Anthropogenic emissions
Stochastic variable	ϵ_t	Natural inflow (lognormal)
Transition function	$X_{t+1} = X_t + a_t + \frac{X_t^q}{1+X_t^q} - bX_t + \epsilon_t$	Pollution dynamics
Objective	$\sum_t \alpha a_t \delta^t$	Discounted economic benefit
Constraint	$X_t < X_{\text{crit}}(b, q)$	Avoid eutrophication
Critical threshold	$\frac{X_{\text{crit}}^q}{1+X_{\text{crit}}^q} = bX_{\text{crit}}$	Depends on b, q
Uncertainty	ϵ_t: stochastic, $\Theta = \{\mu, \sigma, b, q, \delta\}$: deep	Mixed uncertainty

---

Would you like me to **extend this refinement into a Bellman equation form** (i.e., the dynamic programming formulation for decision optimization under stochastic and deep uncertainty)?
That's the next natural step for computational or policy analysis.

Various stakeholders approach this problem from different perspectives. Please refine the model specification from the perspectives of: (1) the local community; and (2) the environmental regulator. These two perspectives should share a common environment, with each stakeholder independently controlling its own control variables and indirectly influencing each other through the shared environment. In the specification, specify the common environment, including global parameters and variables. For each perspective, also specify the corresponding model and its key components, such as control variables, transition functions, and objective functions. If there are multiple objectives, provide multiple objective functions instead of a weighted objective function.

Excellent — now we're moving from a **single-agent dynamic decision problem** to a **multi-agent coupled dynamic system** (a shared-environment game).

This extension represents a more realistic situation:

- The **local community** wants to maximize economic growth and livelihood benefits.
- The **environmental regulator** seeks to protect the lake and ensure sustainability.
  Both agents operate **within a shared environment**, represented by the lake's pollution dynamics, and their actions indirectly affect each other.

Below is a refined formal specification that clearly distinguishes the **shared environment** and each **stakeholder's model**, while keeping the dynamic system coherent.

🌍 1. Common Environment (Shared System)

1.1 Global State Variable

$$X_t \in \mathbb{R}_{\geq 0}$$

- Pollution level in the shared lake at time t.

1.2 Global Exogenous Stochastic Variable

$$\epsilon_t \sim \mathrm{LogNormal}(\mu, \sigma^2)$$

- Represents random natural inflow of pollution (stochastic uncertainty).

1.3 Global Transition Function (Environmental Dynamics)

$$X_{t+1} = X_t + a_t^{(L)} + a_t^{(R)} + \frac{X_t^q}{1 + X_t^q} - bX_t + \epsilon_t$$

where:

- $a_t^{(L)}$: anthropogenic emissions by **the local community**,
- $a_t^{(R)}$: regulatory intervention by **the regulator**, which may represent *mitigation*, *remediation*, or *pollution permits* (can be positive or negative),
- $\frac{X_t^q}{1+X_t^q}$: endogenous nutrient recycling,
- $-bX_t$: natural decay.

The two stakeholders **interact through** X_t — the lake's pollution state.

1.4 Global Uncertain Parameters (Deep Uncertainty)

$$\Theta = \{\mu, \sigma, b, q, \delta_L, \delta_R\}$$

- Each parameter may be uncertain and differently perceived by the two actors:
  - b: natural removal rate,
  - q: recycling rate exponent,
  - μ, σ: parameters of stochastic inflow,
  - δ_L, δ_R: discount rates for the local community and regulator respectively.

1.5 Endogenous Critical Threshold (Shared Environmental Limit)

$$\frac{X_{\text{crit}}^q}{1 + X_{\text{crit}}^q} = bX_{\text{crit}} \quad \Rightarrow \quad X_{\text{crit}} = X_{\text{crit}}(b, q)$$

Crossing this threshold leads to irreversible eutrophication — a shared catastrophic outcome affecting both agents.

👥 2. Stakeholder (1): Local Community

2.1 Perspective

- Seeks to **maximize economic benefit** from industrial and agricultural activity.

- Considers environmental quality only indirectly (e.g., via regulation or penalties).

2.2 Control Variable

$$a_t^{(L)} \in [0, a_{\max}^{(L)}]$$

- Anthropogenic emissions (industrial/agricultural pollution rate).
- Directly increases local income but also increases X_t.

2.3 Perceived State Variable

$$X_t \text{ (shared lake pollution level)}$$

The local community responds to the observed pollution state (and possibly regulatory policy).

2.4 Transition Function (from community perspective)

The community's emissions affect pollution according to:

$$X_{t+1} = X_t + a_t^{(L)} + a_t^{(R)} + \frac{X_t^q}{1 + X_t^q} - bX_t + \epsilon_t$$

(same as the shared transition function).

2.5 Objective Functions

The community has potentially **two objectives**:

1. **Economic Benefit Maximization**

$$f_{\text{econ}}^{(L)} = \sum_{t=0}^{T} \alpha_L \, a_t^{(L)} \, (\delta_L)^t$$

   - α_L: economic benefit per unit pollution.

2. **Avoidance of Environmental Penalties (optional)**
   If regulation imposes penalties or reduced welfare when pollution is high:

$$f_{\text{env}}^{(L)} = -\sum_{t=0}^{T} \pi(X_t, X_{\text{crit}}) \, (\delta_L)^t$$

   where $\pi(\cdot)$ is a penalty function (e.g., increasing rapidly as $X_t \to X_{\text{crit}}$).

Hence, the local community's multi-objective problem is:

$$\max_{a_t^{(L)}} \left\{ f_{\text{econ}}^{(L)}, \; f_{\text{env}}^{(L)} \right\}$$

🏛️ 3. Stakeholder (2): Environmental Regulator

3.1 Perspective

- Seeks to **protect the lake** and maintain pollution below the critical threshold.
- May also value **economic stability** or **social welfare**.

3.2 Control Variable

$$a_t^{(R)} \in [a_{\min}^{(R)}, a_{\max}^{(R)}]$$

- Represents **regulatory actions** such as:
  - Pollution permits (positive $a_t^{(R)}$ increases allowed emissions),
  - Remediation or cleanup (negative $a_t^{(R)}$),
  - Enforced reduction targets or taxes.

3.3 Perceived State Variable

$$X_t \text{ (shared pollution stock)}$$

The regulator observes X_t and possibly $a_t^{(L)}$ when setting policy.

3.4 Transition Function

Same as the shared environmental dynamic:

$$X_{t+1} = X_t + a_t^{(L)} + a_t^{(R)} + \frac{X_t^q}{1 + X_t^q} - bX_t + \epsilon_t$$

3.5 Objective Functions

The regulator's objectives are typically **environmental protection** and **social-economic welfare**:

1. **Environmental Protection Objective**

$$f_{\text{env}}^{(R)} = -\sum_{t=0}^{T} (X_t - X_{\text{crit}}(b, q))_+^2 \ (\delta_R)^t$$

   (minimize pollution and risk of threshold breach).

2. **Economic Welfare Objective**

$$f_{\text{econ}}^{(R)} = \sum_{t=0}^{T} \beta_R \, \text{EconomicActivity}(a_t^{(L)}, a_t^{(R)}) \ (\delta_R)^t$$

- β_R: weight for economic wellbeing from community activities.
- $\text{EconomicActivity}(\cdot)$: could be proportional to $a_t^{(L)}$, capturing societal benefit.

Thus, the regulator's multi-objective problem is:

$$\max_{a_t^{(R)}} \left\{ f_{\text{env}}^{(R)},\ f_{\text{econ}}^{(R)} \right\}$$

⚖️ 4. Interaction Between Stakeholders

- Both agents affect the shared pollution state X_t.
- Their actions are **interdependent**:
  - The community's emissions increase pollution, triggering stricter regulation.
  - The regulator's actions (taxes, cleanup, or caps) constrain community behavior.
- The shared environment introduces **dynamic coupling** between agents' decisions.

This setup can be treated as:

- A **dynamic game** with shared state variable X_t;
- Or a **multi-agent stochastic dynamic system** solved using cooperative (Pareto) or noncooperative (Nash) formulations.

🧮 5. Combined System Overview

Element	Symbol	Description
Shared state	X_t	Lake pollution level
Community control	$a_t^{(L)}$	Industrial/agricultural emissions
Regulator control	$a_t^{(R)}$	Policy action (permits/remediation)
Stochastic variable	ϵ_t	Natural inflow (lognormal)
Transition function	$X_{t+1} = X_t + a_t^{(L)} + a_t^{(R)} + \frac{X_t^q}{1+X_t^q} - bX_t + \epsilon_t$	Shared dynamics
Critical threshold	$X_{\text{crit}}(b, q)$	Endogenous, deep uncertainty
Deep uncertainty vector	$\Theta = \{\mu, \sigma, b, q, \delta_L, \delta_R\}$	Parameter uncertainty
Community objectives	$f_{\text{econ}}^{(L)}, f_{\text{env}}^{(L)}$	Economic benefit, penalty avoidance
Regulator objectives	$f_{\text{env}}^{(R)}, f_{\text{econ}}^{(R)}$	Environmental protection, economic welfare

Optional Extensions

- **Game type**:
  - *Nash equilibrium* (non-cooperative)
  - *Stackelberg game* (regulator as leader, community as follower)
  - *Cooperative negotiation* (Pareto frontier)
- **Uncertainty treatment**:

- Stochastic dynamic programming for ϵ_t,
- Scenario-based exploration for deep uncertainty in Θ.

Would you like me to formalize this two-agent system as a **dynamic game** (e.g., Stackelberg or Nash formulation) with corresponding Bellman equations for each agent? That would be the next precise mathematical step in specifying their strategic interaction.

> To ensure internal consistency across these models, please compose them into a single, unified model that shares a common environment and states, while embedding perspective-specific controls and objectives. The composed model should: 1. embed each perspective's control variables and objectives; 2. enable modular evaluation of each perspective; 3. for each perspective, treat other perspectives' controls as exogenous inputs; and 4. incorporate all other components from every perspective.

Perfect — that's exactly the right next step.
We'll now **compose** the previous separate stakeholder models into a **unified, modular, internally consistent system**, in which:

- There is **one shared environment** (the lake system),
- Each **stakeholder** (local community and environmental regulator) has:
  - its own **control variables** and **objectives**,
  - views the other's controls as **exogenous inputs**,
- All stochastic and deep uncertainties are consistent and shared.

The resulting model can be viewed as a **modular dynamic system with perspective-specific submodels**, suitable for simulation, optimization, or game-theoretic analysis.

🌍 1. Unified Lake System Model (Composed Form)

We define a **composite model**

$$\mathcal{M} = \langle \mathcal{E}, \mathcal{L}, \mathcal{R} \rangle$$

where:

- $\mathcal{E}$: the **shared environment subsystem**,
- $\mathcal{L}$: the **local community module**,
- $\mathcal{R}$: the **regulator module**.

Each module interacts via the shared state variable X_t and the global uncertainty structure.

🧩 2. Shared Environment Subsystem $\mathcal{E}$

2.1 Global State Variable

$$X_t \in \mathbb{R}_{\geq 0}$$

Lake pollution level (shared by all stakeholders).

2.2 Exogenous Stochastic Variable

$$\epsilon_t \sim \text{LogNormal}(\mu, \sigma^2)$$

Natural inflow of pollution (stochastic uncertainty).

2.3 Transition Function (Global System Dynamics)

$$X_{t+1} = X_t + a_t^{(L)} + a_t^{(R)} + \frac{X_t^q}{1 + X_t^q} - bX_t + \epsilon_t$$

where:

- $a_t^{(L)}$: emissions chosen by the local community,
- $a_t^{(R)}$: regulatory intervention or remediation,
- b: natural removal rate,
- q: recycling exponent,
- ϵ_t: stochastic inflow.

2.4 Deep Uncertainty Parameters

$$\Theta = \{\mu, \sigma, b, q, \delta_L, \delta_R\}$$

All agents share these uncertain parameters (possibly with different subjective beliefs).

2.5 Endogenous Critical Threshold

$$X_{\text{crit}}(b, q) \text{ such that } \frac{X_{\text{crit}}^q}{1 + X_{\text{crit}}^q} = bX_{\text{crit}}$$

Crossing this threshold implies irreversible eutrophication.

👥 3. Local Community Module $\mathcal{L}$

3.1 Control Variable

$$a_t^{(L)} \in [0, a_{\max}^{(L)}]$$

Anthropogenic emissions rate (industrial and agricultural pollution).

3.2 Inputs (Exogenous to Community)

- $a_t^{(R)}$: regulator's policy (taken as given when optimizing locally),
- X_t: shared pollution level (state variable from $\mathcal{E}$).

3.3 Dynamics (From Shared Environment)

The community's control affects the shared system via:

$$X_{t+1} = F(X_t, a_t^{(L)}, a_t^{(R)}, \epsilon_t; b, q)$$

3.4 Objectives (Multi-Objective)

The local community evaluates two independent criteria:

1. **Economic Benefit**

$$f_{\text{econ}}^{(L)} = \sum_{t=0}^{T} \alpha_L \, a_t^{(L)} (\delta_L)^t$$

2. **Environmental Penalty / Livelihood Loss**

$$f_{\text{env}}^{(L)} = -\sum_{t=0}^{T} \pi_L(X_t, X_{\text{crit}}(b, q)) \, (\delta_L)^t$$

where $\pi_L(\cdot)$ penalizes local quality-of-life losses as the lake degrades (e.g., fishery loss, health damage).

3.5 Local Optimization Problem

Given exogenous $a_t^{(R)}$ and the stochastic environment:

$$\begin{aligned} \max_{\{a_t^{(L)}\}} \quad & \{f_{\text{econ}}^{(L)}, f_{\text{env}}^{(L)}\} \\ \text{s.t.} \quad & X_{t+1} = F(X_t, a_t^{(L)}, a_t^{(R)}, \epsilon_t; b, q), \\ & X_0 = 0, \; a_t^{(L)} \geq 0, \; X_t < X_{\text{crit}}(b, q) \end{aligned}$$

🏛️ 4. Regulator Module $\mathcal{R}$

4.1 Control Variable

$$a_t^{(R)} \in [a_{\min}^{(R)}, a_{\max}^{(R)}]$$

Regulatory intervention:

- $a_t^{(R)} > 0$: pollution permit or subsidy (relaxes emissions),
- $a_t^{(R)} < 0$: cleanup investment or stricter regulation (reduces pollution).

4.2 Inputs (Exogenous to Regulator)

- $a_t^{(L)}$: community's emission level (treated as given when optimizing regulator's policy),
- X_t: shared pollution level.

4.3 Dynamics (From Shared Environment)

$$X_{t+1} = F(X_t, a_t^{(L)}, a_t^{(R)}, \epsilon_t; b, q)$$

(same as in $\mathcal{E}$).

4.4 Objectives (Multi-Objective)

1. **Environmental Protection**

$$f_{\text{env}}^{(R)} = -\sum_{t=0}^{T} (X_t - X_{\text{crit}}(b,q))_+^2 \, (\delta_R)^t$$

   (minimize risk of crossing the critical threshold).

2. **Economic/Social Welfare**

$$f_{\text{econ}}^{(R)} = \sum_{t=0}^{T} \beta_R \, \text{Welfare}(a_t^{(L)}, a_t^{(R)}) (\delta_R)^t$$

   where $\text{Welfare}(\cdot)$ measures economic or employment benefits arising from regulated activity.

4.5 Regulatory Optimization Problem

Given exogenous $a_t^{(L)}$:

$$\begin{aligned} \max_{\{a_t^{(R)}\}} \quad & \{ f_{\text{env}}^{(R)}, f_{\text{econ}}^{(R)} \} \\ \text{s.t.} \quad & X_{t+1} = F(X_t, a_t^{(L)}, a_t^{(R)}, \epsilon_t; b, q), \\ & X_t < X_{\text{crit}}(b,q), \;\; a_t^{(R)} \in [a_{\min}^{(R)}, a_{\max}^{(R)}] \end{aligned}$$

🔄 5. Modular Composition and Interaction Rules

5.1 Shared Coupling Through Environment

Both submodels $\mathcal{L}$ and $\mathcal{R}$ are **coupled** through:

- The shared state X_t,
- The transition function $F(\cdot)$,
- The common stochastic inflow ϵ_t,
- Shared deep uncertainties Θ.

5.2 Modular Evaluation

Each module can be **evaluated independently** given exogenous inputs:

- $\mathcal{L}$: Evaluate local policy given fixed $a_t^{(R)}$,
- $\mathcal{R}$: Evaluate regulatory policy given fixed $a_t^{(L)}$.

This structure enables:

- **Independent optimization** (for simulation or Nash equilibria),
- **Sequential optimization** (for Stackelberg leader–follower formulations),
- **Joint evaluation** (for cooperative or Pareto analysis).

5.3 Unified Dynamics

All perspectives share:

$$X_{t+1} = X_t + a_t^{(L)} + a_t^{(R)} + \frac{X_t^q}{1 + X_t^q} - bX_t + \epsilon_t$$

which ensures internal consistency.

🧮 6. Unified Model Summary

Component	Symbol / Function	Description	Scope
State variable	X_t	Lake pollution	Shared
Stochastic variable	$\epsilon_t \sim \mathrm{LogNormal}(\mu, \sigma^2)$	Natural inflow	Shared
Deep uncertainties	$\Theta = \{\mu, \sigma, b, q, \delta_L, \delta_R\}$	Model parameters	Shared
Transition function	$X_{t+1} = X_t + a_t^{(L)} + a_t^{(R)} + \frac{X_t^q}{1+X_t^q} - bX_t + \epsilon_t$	Pollution dynamics	Shared
Threshold	$X_{\text{crit}}(b, q)$	Endogenous eutrophication limit	Shared
Control (community)	$a_t^{(L)}$	Industrial/agricultural emissions	Local
Control (regulator)	$a_t^{(R)}$	Policy/cleanup intervention	Regulator
Objectives (community)	$f_{\text{econ}}^{(L)}, f_{\text{env}}^{(L)}$	Economic benefit, penalty avoidance	Local
Objectives (regulator)	$f_{\text{env}}^{(R)}, f_{\text{econ}}^{(R)}$	Environmental protection, economic welfare	Regulator

---

🔧 7. Conceptual Interpretation

- **System level:**
  One shared nonlinear stochastic environmental process driven by combined human and natural inputs.

- **Agent level:**
  Each stakeholder acts based on its own objective set, taking the other's actions as exogenous.
- **Integration:**
  The unified model provides a **common simulation and policy framework**, supporting:
  - *Independent evaluation* (e.g., sensitivity of each perspective to the other),
  - *Strategic interaction* (e.g., Nash/Stackelberg solutions),
  - *Joint robust decision-making* (e.g., under deep uncertainty).

Would you like me to now **formalize this unified model as a coupled dynamic game system**, with explicit Bellman equations or equilibrium definitions for each stakeholder? That would make it ready for numerical analysis or simulation-based policy exploration.

> Now, provide a modular Python implementation of the unified model, using classes to represent different perspectives. At the end of the program, define a function as the interface. This function should accept a dictionary of control variables and a dictionary of uncertain parameter values as input, simulate the model dynamics, and return the value of each objective function as output. This function does not need to be executed.

Here's a **modular, object-oriented Python implementation** of the **unified lake problem model** that captures:

- a **shared environmental system**;
- two stakeholder perspectives — the **local community** and the **environmental regulator**;
- deep and stochastic uncertainty;
- flexible, modular evaluation of each stakeholder's objectives;
- and an external **interface function** for simulation.

This design supports independent or joint evaluation and can be easily extended for optimization or game-theoretic studies.

```python
import numpy as np

# ============================================================
# 1. Shared Environment Model
# ============================================================

class LakeEnvironment:
    """
    Shared environmental system representing lake pollution dynamics.
    """
    def __init__(self, params):
        """
        Initialize environment parameters.
        params: dict containing 'b', 'q', 'mu', 'sigma', and 'T' (time horizon)
        """
        self.b = params.get('b', 0.4)
```

```python
        self.q = params.get('q', 2.0)
        self.mu = params.get('mu', 0.0)
        self.sigma = params.get('sigma', 0.1)
        self.T = params.get('T', 50)

        # Initialize pollution state
        self.X = [0.0]  # initial pollution level

    def critical_threshold(self):
        """
        Compute the endogenous critical pollution threshold X_crit(b, q)
        defined implicitly by X^q / (1 + X^q) = b * X
        """
        # Numerical fixed-point solver for X_crit
        x = 1e-6
        for _ in range(10000):
            f = (x ** self.q) / (1 + x ** self.q) - self.b * x
            df = (self.q * (x ** (self.q - 1)) / (1 + x ** self.q)**2) - self.b
            x_new = x - f / df
            if abs(x_new - x) < 1e-8:
                return x_new
            x = max(x_new, 1e-8)
        return x  # fallback if no convergence

    def stochastic_inflow(self, rng=np.random.default_rng()):
        """
        Sample stochastic natural inflow epsilon_t from LogNormal(mu, sigma^2)
        """
        return rng.lognormal(mean=self.mu, sigma=self.sigma)

    def transition(self, a_L, a_R, epsilon_t):
        """
        Compute the next pollution state.
        """
        X_t = self.X[-1]
        X_next = X_t + a_L + a_R + (X_t ** self.q) / (1 + X_t ** self.q) - self.b * X_t +
epsilon_t
        self.X.append(X_next)
        return X_next

    def reset(self):
        """Reset pollution to initial state."""
        self.X = [0.0]


# ==============================================================
# 2. Local Community Model
# ==============================================================

class LocalCommunity:
    """
    Local community perspective: seeks economic benefit and avoids penalties.
    """
    def __init__(self, params):
        self.alpha = params.get('alpha_L', 1.0)
        self.delta = params.get('delta_L', 0.95)
        self.penalty_scale = params.get('penalty_scale_L', 10.0)

    def economic_benefit(self, a_L_series):
        """Compute discounted economic benefit."""
        return sum(self.alpha * a * (self.delta ** t) for t, a in enumerate(a_L_series))

    def environmental_penalty(self, X_series, X_crit):
        """Compute discounted environmental penalty."""
        penalties = [(max(0, X - X_crit))**2 for X in X_series]
        return -sum(self.penalty_scale * p * (self.delta ** t) for t, p in enumerate(penalties))

    def evaluate_objectives(self, a_L_series, X_series, X_crit):
        """Return both objective function values."""
        return {
            "f_econ_L": self.economic_benefit(a_L_series),
            "f_env_L": self.environmental_penalty(X_series, X_crit)
        }
```

```python
# ============================================================
# 3. Environmental Regulator Model
# ============================================================

class Regulator:
    """
    Environmental regulator perspective: protect environment & sustain welfare.
    """
    def __init__(self, params):
        self.beta = params.get('beta_R', 0.5)
        self.delta = params.get('delta_R', 0.95)
        self.env_weight = params.get('env_weight_R', 20.0)

    def environmental_protection(self, X_series, X_crit):
        """Penalty for pollution exceeding threshold."""
        penalties = [(max(0, X - X_crit))**2 for X in X_series]
        return -sum(self.env_weight * p * (self.delta ** t) for t, p in enumerate(penalties))

    def economic_welfare(self, a_L_series, a_R_series):
        """Social-economic welfare tied to activities."""
        welfare = [self.beta * (a_L + 0.5 * a_R) for a_L, a_R in zip(a_L_series, a_R_series)]
        return sum(w * (self.delta ** t) for t, w in enumerate(welfare))

    def evaluate_objectives(self, a_L_series, a_R_series, X_series, X_crit):
        """Return both objective function values."""
        return {
            "f_env_R": self.environmental_protection(X_series, X_crit),
            "f_econ_R": self.economic_welfare(a_L_series, a_R_series)
        }


# ============================================================
# 4. Unified Simulation Interface
# ============================================================

def simulate_lake_model(control_dict, uncertainty_dict, seed=None):
    """
    Simulate the unified lake problem dynamics under given controls and uncertainties.

    Inputs:
    -------
    control_dict : dict
        {
            "a_L": [list or array of local community emissions over time],
            "a_R": [list or array of regulator interventions over time]
        }

    uncertainty_dict : dict
        {
            "b": float, "q": float, "mu": float, "sigma": float,
            "delta_L": float, "delta_R": float,
            "alpha_L": float, "beta_R": float,
            "T": int
        }

    Returns:
    --------
    results : dict
        {
            "f_econ_L": ..., "f_env_L": ...,
            "f_env_R": ..., "f_econ_R": ...
        }
    """
    # Instantiate shared environment
    env = LakeEnvironment(uncertainty_dict)
    X_crit = env.critical_threshold()
    rng = np.random.default_rng(seed)

    # Instantiate stakeholder modules
    local = LocalCommunity(uncertainty_dict)
    regulator = Regulator(uncertainty_dict)

    # Controls
    a_L_series = control_dict.get("a_L", [0.0] * env.T)
    a_R_series = control_dict.get("a_R", [0.0] * env.T)
```

```
    # Simulation loop
    for t in range(env.T):
        eps_t = env.stochastic_inflow(rng)
        env.transition(a_L=a_L_series[t], a_R=a_R_series[t], epsilon_t=eps_t)

    X_series = env.X

    # Evaluate objectives
    local_results = local.evaluate_objectives(a_L_series, X_series, X_crit)
    regulator_results = regulator.evaluate_objectives(a_L_series, a_R_series, X_series, X_crit)

    # Combine results
    return {**local_results, **regulator_results}


# ============================================================
# Example (not executed)
# ============================================================
# Example usage:
# control_inputs = {
#     "a_L": [0.3] * 50,
#     "a_R": [-0.1] * 50
# }
# uncertain_params = {
#     "b": 0.4, "q": 2.0, "mu": 0.0, "sigma": 0.1,
#     "delta_L": 0.95, "delta_R": 0.95,
#     "alpha_L": 1.0, "beta_R": 0.5,
#     "T": 50
# }
# results = simulate_lake_model(control_inputs, uncertain_params)
# print(results)
```

🧩 Key Features of This Implementation

1. **Shared Environment (`LakeEnvironment`)**
   Encapsulates all global dynamics, stochastic inflows, and the endogenous threshold.
2. **Perspective Classes**
   - **`LocalCommunity`**: manages local control, benefit, and environmental penalties.
   - **`Regulator`**: manages regulatory control, protection, and welfare objectives.
3. **Interface Function (`simulate_lake_model`)**
   - Accepts **`control_dict`** and **`uncertainty_dict`**.
   - Simulates the coupled dynamics over the time horizon.
   - Returns all perspective-specific objective values in one dictionary.
4. **Modular Evaluation**
   Each perspective can be replaced or extended independently, supporting multi-agent or robust decision analyses.

Would you like me to extend this code to **allow adaptive or policy-based controls** (e.g., feedback functions $a_t^{(L)}(X_t)$ and $a_t^{(R)}(X_t)$) for dynamic decision-making or optimization next?

There are two areas where your implementation could be improved: 1. The possible value of X_{crit} is in the range [0.01, 1.5], and its exact value can be estimated using the root-finding function; 2. Your implementation of the lognormal distribution is incorrect. Please refine your implementation.

Excellent feedback — both points are **precise and important for model realism**.

Let's refine the Python implementation accordingly:

1. **Critical threshold** X_{crit}
   - The valid range is now constrained to $[0.01, 1.5]$.
   - We'll use a **robust root-finding algorithm** (`scipy.optimize.root_scalar`) to solve

     $$f(X) = \frac{X^q}{1 + X^q} - bX = 0$$

     within this interval.
2. **Lognormal stochastic inflow** ϵ_t
   - The correct way to specify a lognormal random variable is to treat μ, σ as the **mean and standard deviation in the linear domain**, not the underlying normal domain.
   - The transformation from desired mean μ and variance σ^2 to the lognormal parameters $\mu_{\log}$ and $\sigma_{\log}$ is:

     $$\sigma_{\log}^2 = \ln\left(1 + \frac{\sigma^2}{\mu^2}\right), \quad \mu_{\log} = \ln(\mu) - \frac{1}{2}\sigma_{\log}^2$$

   - We'll incorporate this correction.

✅ Refined Unified Model Implementation

```python
import numpy as np
from scipy.optimize import root_scalar

# ============================================================
# 1. Shared Environment Model
# ============================================================

class LakeEnvironment:
    """
    Shared environmental system representing lake pollution dynamics.
    """
    def __init__(self, params):
        """
        Initialize environment parameters.
        params: dict containing 'b', 'q', 'mu', 'sigma', and 'T' (time horizon)
        """
        self.b = params.get('b', 0.4)
        self.q = params.get('q', 2.0)
        self.mu = params.get('mu', 0.05)      # mean of lognormal (linear domain)
        self.sigma = params.get('sigma', 0.02) # std dev of lognormal (linear domain)
        self.T = params.get('T', 50)

        # Initialize pollution state
```

```
        self.X = [0.0]  # initial pollution level

    # -------------------------------------------------------
    # Compute the critical threshold within [0.01, 1.5]
    # -------------------------------------------------------
    def critical_threshold(self):
        """
        Compute the endogenous critical pollution threshold X_crit(b, q)
        defined implicitly by X^q / (1 + X^q) = b * X.
        The solution is constrained to [0.01, 1.5].
        """
        def f(x):
            return (x ** self.q) / (1 + x ** self.q) - self.b * x

        sol = root_scalar(f, bracket=[0.01, 1.5], method='brentq')
        if sol.converged:
            return sol.root
        else:
            raise RuntimeError("Failed to find X_crit in [0.01, 1.5].")

    # -------------------------------------------------------
    # Sample stochastic natural inflow ε_t from lognormal(μ, σ)
    # -------------------------------------------------------
    def stochastic_inflow(self, rng=np.random.default_rng()):
        """
        Sample ε_t ~ LogNormal(μ, σ^2), where μ and σ are the mean and std dev
        in the linear domain. The corresponding log-space parameters are derived.
        """
        mu_log = np.log(self.mu**2 / np.sqrt(self.sigma**2 + self.mu**2))
        sigma_log = np.sqrt(np.log(1 + (self.sigma**2 / self.mu**2)))
        return rng.lognormal(mean=mu_log, sigma=sigma_log)

    # -------------------------------------------------------
    # Pollution transition dynamics
    # -------------------------------------------------------
    def transition(self, a_L, a_R, epsilon_t):
        """
        Compute the next pollution state.
        """
        X_t = self.X[-1]
        X_next = X_t + a_L + a_R + (X_t ** self.q) / (1 + X_t ** self.q) - self.b * X_t +
epsilon_t
        self.X.append(X_next)
        return X_next

    def reset(self):
        """Reset pollution to initial state."""
        self.X = [0.0]


# ============================================================
# 2. Local Community Model
# ============================================================

class LocalCommunity:
    """
    Local community perspective: seeks economic benefit and avoids penalties.
    """
    def __init__(self, params):
        self.alpha = params.get('alpha_L', 1.0)
        self.delta = params.get('delta_L', 0.95)
        self.penalty_scale = params.get('penalty_scale_L', 10.0)

    def economic_benefit(self, a_L_series):
        """Compute discounted economic benefit."""
        return sum(self.alpha * a * (self.delta ** t) for t, a in enumerate(a_L_series))

    def environmental_penalty(self, X_series, X_crit):
        """Compute discounted environmental penalty."""
        penalties = [(max(0, X - X_crit))**2 for X in X_series]
        return -sum(self.penalty_scale * p * (self.delta ** t) for t, p in enumerate(penalties))

    def evaluate_objectives(self, a_L_series, X_series, X_crit):
        """Return both objective function values."""
        return {
```

```
            "f_econ_L": self.economic_benefit(a_L_series),
            "f_env_L": self.environmental_penalty(X_series, X_crit)
        }


# ============================================================
# 3. Environmental Regulator Model
# ============================================================

class Regulator:
    """
    Environmental regulator perspective: protect environment & sustain welfare.
    """
    def __init__(self, params):
        self.beta = params.get('beta_R', 0.5)
        self.delta = params.get('delta_R', 0.95)
        self.env_weight = params.get('env_weight_R', 20.0)

    def environmental_protection(self, X_series, X_crit):
        """Penalty for pollution exceeding threshold."""
        penalties = [(max(0, X - X_crit))**2 for X in X_series]
        return -sum(self.env_weight * p * (self.delta ** t) for t, p in enumerate(penalties))

    def economic_welfare(self, a_L_series, a_R_series):
        """Social-economic welfare tied to activities."""
        welfare = [self.beta * (a_L + 0.5 * a_R) for a_L, a_R in zip(a_L_series, a_R_series)]
        return sum(w * (self.delta ** t) for t, w in enumerate(welfare))

    def evaluate_objectives(self, a_L_series, a_R_series, X_series, X_crit):
        """Return both objective function values."""
        return {
            "f_env_R": self.environmental_protection(X_series, X_crit),
            "f_econ_R": self.economic_welfare(a_L_series, a_R_series)
        }


# ============================================================
# 4. Unified Simulation Interface
# ============================================================

def simulate_lake_model(control_dict, uncertainty_dict, seed=None):
    """
    Simulate the unified lake problem dynamics under given controls and uncertainties.

    Inputs:
    -------
    control_dict : dict
        {
            "a_L": [list or array of local community emissions over time],
            "a_R": [list or array of regulator interventions over time]
        }

    uncertainty_dict : dict
        {
            "b": float, "q": float, "mu": float, "sigma": float,
            "delta_L": float, "delta_R": float,
            "alpha_L": float, "beta_R": float,
            "T": int
        }

    Returns:
    --------
    results : dict
        {
            "f_econ_L": ..., "f_env_L": ...,
            "f_env_R": ..., "f_econ_R": ...
        }
    """
    # Instantiate shared environment
    env = LakeEnvironment(uncertainty_dict)
    X_crit = env.critical_threshold()
    rng = np.random.default_rng(seed)

    # Instantiate stakeholder modules
    local = LocalCommunity(uncertainty_dict)
```

```
    regulator = Regulator(uncertainty_dict)

    # Controls
    a_L_series = control_dict.get("a_L", [0.0] * env.T)
    a_R_series = control_dict.get("a_R", [0.0] * env.T)

    # Simulation loop
    for t in range(env.T):
        eps_t = env.stochastic_inflow(rng)
        env.transition(a_L=a_L_series[t], a_R=a_R_series[t], epsilon_t=eps_t)

    X_series = env.X

    # Evaluate objectives
    local_results = local.evaluate_objectives(a_L_series, X_series, X_crit)
    regulator_results = regulator.evaluate_objectives(a_L_series, a_R_series, X_series, X_crit)

    # Combine results
    return {**local_results, **regulator_results}


# ============================================================
# Example (not executed)
# ============================================================
# control_inputs = {
#     "a_L": [0.3] * 50,
#     "a_R": [-0.1] * 50
# }
# uncertain_params = {
#     "b": 0.4, "q": 2.0, "mu": 0.05, "sigma": 0.02,
#     "delta_L": 0.95, "delta_R": 0.95,
#     "alpha_L": 1.0, "beta_R": 0.5,
#     "T": 50
# }
# results = simulate_lake_model(control_inputs, uncertain_params)
# print(results)
```

🔧 Key Refinements

Improvement	Before	After
Critical Threshold X_{crit}	Iterative solver without bound	Uses **`scipy.optimize.root_scalar`** within [0.01, 1.5]
Lognormal Sampling	Treated μ, σ as log-space parameters	Converts mean & std from linear to log-space correctly
Stability	No guarantee of valid solution	Guaranteed root within plausible ecological range
Interpretability	μ and σ ambiguous	Explicitly defined as mean & std in linear space

This refined version is now **numerically stable**, **ecologically plausible**, and **statistically correct** — suitable for both exploratory simulations and policy optimization.