Experiments 3 - Lake 4

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 essentially describing a **formal stochastic–dynamic decision model** under **deep uncertainty**, often used in environmental economics and sustainability planning (the "Lake Problem" is a canonical example). Let's formalize it carefully and systematically.

🧭 1. Model Structure

We define a **stochastic dynamic optimization model** over a discrete time horizon $t = 0, 1, 2, \ldots, T$.

The decision-maker chooses the **pollution emission rate** a_t each year to maximize discounted economic benefits, while controlling the risk of lake eutrophication.

🧩 2. Model Components

State Variables

Symbol	Description
X_t	Lake pollution stock at time t (state of the system).

- Domain: $X_t \in [0, X_{\text{crit}}]$ ideally (eutrophication occurs when $X_t > X_{\text{crit}}$).

Decision Variables

Symbol	Description
a_t	Anthropogenic (human-caused) pollution emission at time t.

- Control variable chosen by the decision-maker.
- $a_t \geq 0$.

Stochastic Variables

Symbol	Description
ϵ_t	Natural pollution inflow at time t, stochastic.

- $\epsilon_t \sim \text{LogNormal}(\mu, \sigma^2)$
- Represents **stochastic uncertainty** (yearly environmental fluctuations).

Constant Parameters

Symbol	Description
b	Natural pollution removal rate of the lake.
q	Natural recycling rate of the lake (nonlinear feedback parameter).
α	Economic benefit-to-pollution ratio.
δ	Discount factor, $0 < \delta \leq 1$.
X_{crit}	Critical pollution threshold (tipping point).
T	Time horizon (finite or infinite).

Uncertain Parameters (Deep Uncertainty)

Deep uncertainty affects the “structural” and “parametric” parts of the model:

Symbol	Description	Uncertainty Type
μ	Mean of lognormal natural pollution inflow	deep
σ	Std. deviation of lognormal inflow	deep

Symbol	Description	Uncertainty Type
b	Natural removal rate	deep
q	Recycling rate	deep
δ	Discount rate	deep

These parameters may take values from plausible ranges or distributions reflecting **deep uncertainty**, often explored through **robust decision-making** (RDM), **info-gap**, or **dynamic adaptive policy pathways** frameworks.

⚙️ 3. Transition Function (System Dynamics)

Pollution evolves according to:

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

where:

- X_t: current pollution stock
- a_t: chosen emissions
- $\frac{X_t^q}{1+X_t^q}$: nonlinear recycling feedback (captures self-reinforcing pollution accumulation)
- bX_t: natural removal
- ϵ_t: stochastic natural inflow

Initial condition: $X_0 = 0$

💰 4. Objective Function (Economic Benefit)

The decision-maker maximizes expected discounted economic benefit from pollution emissions:

$$\max_{\{a_t\}_{t=0}^T} \mathbb{E}\left[\sum_{t=0}^{T} \alpha a_t \, \delta^t\right]$$

subject to the transition function and the eutrophication constraint $X_t \leq X_{\text{crit}}$ (or a penalty if violated).

⚖️ 5. Constraints

1. **System Dynamics:** $X_{t+1} = X_t + a_t + \frac{X_t^q}{1+X_t^q} - bX_t + \epsilon_t$
2. **Non-negativity:** $a_t \geq 0, \;\; X_t \geq 0$
3. **Threshold Constraint (optional hard constraint):** $X_t \leq X_{\text{crit}}$
   - or soft constraint using penalty function in objective.

📈 6. Stochastic–Dynamic Formulation

Bellman Equation (Expected Value Formulation)

$$V_t(X_t) = \max_{a_t \geq 0} \left\{ \alpha a_t \delta^t + \mathbb{E}_{\epsilon_t} \left[V_{t+1} \left(X_t + a_t + \frac{X_t^q}{1 + X_t^q} - bX_t + \epsilon_t \right) \right] \right\}$$

with boundary condition $V_{T+1}(X) = 0$.

Deep uncertainty enters as **parametric ambiguity** in the functions $b, q, \mu, \sigma, \delta$.
Robust or adaptive extensions can be expressed as:

$$\max_{a_t} \min_{\theta \in \Theta} \mathbb{E}_{\epsilon_t | \theta} \left[\sum_{t=0}^{T} \alpha a_t \delta^t \right]$$

where Θ is the uncertainty set for deep uncertain parameters.

🧠 7. Model Summary

Category	Symbols	Description
State Variable	X_t	Lake pollution stock
Control Variable	a_t	Anthropogenic emissions
Stochastic Variable	$\epsilon_t \sim \mathrm{LogNormal}(\mu, \sigma^2)$	Natural inflow
Parameters	$\alpha, \delta, b, q, X_{\mathrm{crit}}$	Economic & physical constants
Deep Uncertainties	$\mu, \sigma, b, q, \delta$	Uncertain structural parameters
Transition Function	$X_{t+1} = X_t + a_t + \frac{X_t^q}{1+X_t^q} - bX_t + \epsilon_t$	System dynamics
Objective Function	$\max \mathbb{E} \left[\sum_t \alpha a_t \delta^t \right]$	Expected discounted benefit
Constraints	$X_t \leq X_{\mathrm{crit}}, a_t \geq 0$	Feasibility & tipping point

---

Would you like me to express this as a **fully specified mathematical program** (i.e., compact optimization problem notation with all expectations and constraints written explicitly) — or as a **computational model formulation** (e.g., in pseudocode or for implementation in a simulation/optimization framework)?

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

Excellent — you're absolutely right.
In the **canonical Lake Problem**, the *critical threshold* X_{crit} is *endogenous*, determined by the intersection of the **lake's recycling** and **removal** functions — both of which depend on *deeply uncertain parameters* b (removal rate) and q (recycling exponent).

Let's refine the formal model to capture that dependency explicitly.

🧩 Refined Model Specification (Endogenous X_{crit})

1. System Definition

We have a **stochastic dynamic decision problem** with **deep uncertainty** in both physical and economic parameters.

$$X_{t+1} = X_t + a_t + \frac{X_t^q}{1 + X_t^q} - bX_t + \epsilon_t, \quad X_0 = 0$$

2. Endogenous Threshold X_{crit}

The **critical pollution level** X_{crit} is *not constant*, but determined implicitly by the equilibrium condition:

$$\text{Recycling rate} = \text{Removal rate}$$

That is,

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

Solving for X_{crit} yields:

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

This nonlinear equation defines $X_{\text{crit}} = X_{\text{crit}}(b, q)$.

Hence, the eutrophication threshold is a derived quantity dependent on uncertain physical parameters b and q.

3. Model Components (Revised Table)

Category	Symbol	Description	Notes
State Variable	X_t	Lake pollution stock	evolves dynamically
Decision Variable	a_t	Anthropogenic emissions at time t	control variable
Stochastic Variable	ϵ_t	Natural inflow	$\epsilon_t \sim \text{LogNormal}(\mu, \sigma^2)$
Parameters (Nominal)	α, δ	Economic parameters	benefit-to-pollution ratio, discount rate
Physical Parameters	b, q	Lake removal rate and recycling exponent	subject to deep uncertainty
Derived Quantity	$X_{\text{crit}}(b, q)$	Critical pollution threshold	implicit function of b and q
Uncertain Parameters (Deep)	$\mu, \sigma, b, q, \delta$	Key sources of deep uncertainty	uncertain across plausible ranges

4. Transition Dynamics

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

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

---

5. Objective Function

$$\max_{\{a_t\}} \mathbb{E}_{\epsilon_t \sim \text{LogNormal}(\mu, \sigma^2)} \left[\sum_{t=0}^{T} \alpha a_t \delta^t \right]$$

subject to the pollution dynamics and the **tipping point condition** based on $X_{\text{crit}}(b, q)$.

---

6. Eutrophication Condition

Eutrophication occurs when:

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

which may be modeled as:

- A **hard constraint**: $X_t \leq X_{\text{crit}}(b, q)$, or
- A **soft penalty** in the objective, e.g.:

$$f_{\text{economic}} = \sum_t \alpha a_t \delta^t - \lambda \, \mathbb{E}[\mathbf{1}_{\{X_t > X_{\text{crit}}(b,q)\}}]$$

  where λ represents the economic loss from eutrophication.

---

7. Deep Uncertainty Structure

Define the set of deep uncertainties:

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

Each parameter has a plausible range or prior distribution:

$$\begin{aligned} \mu &\in [\mu_{\text{min}}, \mu_{\text{max}}] \\ \sigma &\in [\sigma_{\text{min}}, \sigma_{\text{max}}] \\ b &\in [b_{\text{min}}, b_{\text{max}}] \\ q &\in [q_{\text{min}}, q_{\text{max}}] \\ \delta &\in [\delta_{\text{min}}, \delta_{\text{max}}] \end{aligned}$$

---

8. Robust or Adaptive Objective (Optional Extension)

Under deep uncertainty, the problem can be formulated as a **robust stochastic control problem**:

$$\max_{\{a_t\}} \min_{\theta\in\Theta} \mathbb{E}_{\epsilon_t|\theta}\left[\sum_{t=0}^{T} \alpha a_t \delta^t(\theta)\right]$$

subject to:

$$X_{t+1} = X_t + a_t + \frac{X_t^{q(\theta)}}{1+X_t^{q(\theta)}} - b(\theta)X_t + \epsilon_t, \quad X_t \le X_{\text{crit}}(b(\theta), q(\theta))$$

9. Refined Summary Table

Model Component	Symbol / Expression	Notes
State variable	X_t	Lake pollution stock
Control variable	$a_t \ge 0$	Anthropogenic pollution
Stochastic variable	$\epsilon_t \sim \text{LogNormal}(\mu, \sigma^2)$	Random inflow
Transition function	$X_{t+1} = X_t + a_t + \frac{X_t^q}{1+X_t^q} - bX_t + \epsilon_t$	System evolution
Threshold	$X_{\text{crit}}(b,q) : \frac{X_{\text{crit}}^q}{1+X_{\text{crit}}^q} = bX_{\text{crit}}$	Implicit tipping point
Objective	$\max_{a_t} \mathbb{E}\left[\sum_t \alpha a_t \delta^t\right]$	Maximize discounted benefit
Deep uncertainty	$\Theta = \{\mu, \sigma, b, q, \delta\}$	Parameters with unknown “true” values
Robust formulation	$\max_{a_t} \min_{\theta\in\Theta} \mathbb{E}_\theta[\cdot]$	If decision-maker seeks robustness
Constraint	$X_t \le X_{\text{crit}}(b,q)$ or penalty	Avoid irreversible eutrophication

10. Interpretation

- **Endogeneity of X_{crit}:** Because it depends on b and q, the location of the tipping point shifts with deep uncertainty — this makes the policy problem **structurally uncertain**.
- **Decision-making challenge:** The policymaker must choose a_t that performs well **across a range of possible lake behaviors** (defined by plausible $b, q, \mu, \sigma, \delta$).
- **Solution approaches:** Dynamic programming with scenario sampling, robust optimization, or exploratory modeling (e.g., RDM frameworks).

Would you like me to show how to **numerically compute** $X_{\text{crit}}(b,q)$ from the implicit equation (e.g., derive its closed form or give an algorithmic expression for simulation)?

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 — you're now moving from a **single-agent stochastic control problem** to a **multi-agent (multi-stakeholder) dynamic game formulation** of the Lake Problem under deep uncertainty.

Let's formalize this carefully and systematically.
We will model:

1. A **shared environment** (the lake–economic–ecological system).
2. Two **independent decision-making agents**:
   - (1) the **Local Community**, prioritizing *economic benefits*, and
   - (2) the **Environmental Regulator**, prioritizing *ecological protection and sustainability*.

Both agents interact **indirectly through the shared state variable** (pollution stock in the lake).

🌍 1. Common Environment (Shared System)

This is the **biophysical system** both agents influence and depend on.

Global State Variable

Symbol	Description
X_t	Lake pollution stock at time t — shared environmental state.

Global Transition Function

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

where:

- $a_t^{(C)}$: anthropogenic emissions from the **local community** (industry, agriculture).
- $a_t^{(R)}$: regulator's intervention effect — e.g., *pollution abatement effort* (typically negative net emissions).
  - Positive $a_t^{(R)}$ increases pollution (e.g., permitting),
  - Negative $a_t^{(R)}$ reduces pollution (abatement or cleanup).
- $\frac{X_t^q}{1+X_t^q}$: nonlinear recycling / feedback process.
- bX_t: natural removal.
- $\epsilon_t \sim \mathrm{LogNormal}(\mu, \sigma^2)$: stochastic natural inflow.

Shared Deep-Uncertain Parameters

Parameter	Meaning	Uncertainty
b	Natural removal rate	deep
q	Recycling exponent	deep
μ, σ	Lognormal parameters for natural inflow	deep
$\delta^{(C)}, \delta^{(R)}$	Discount rates (agent-specific, uncertain)	deep
$X_{\text{crit}}(b, q)$	Endogenous eutrophication threshold	derived from $\frac{X^q}{1+X^q} = bX$

Thus both stakeholders operate under **shared environmental dynamics** but may have **different preferences and discounting**.

🧩 2. Perspective 1: Local Community

Goal

Maximize **economic and social well-being**, primarily through pollution-emitting activities (industry, agriculture).

Control Variable

Symbol	Description
$a_t^{(C)} \geq 0$	Community's pollution emissions (industrial/agricultural output proxy).

State Variable

Symbol	Description
X_t	Shared lake pollution stock (influenced by both agents).

Transition Function (as perceived by community)

The community views the lake as evolving according to:

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

but takes the regulator's action $a_t^{(R)}$ as exogenous (not under its control).

Objective Functions

The community has **two competing objectives**:

1. **Economic Benefit:**

$$f_{\text{econ}}^{(C)} = \sum_{t=0}^{T} \alpha^{(C)} a_t^{(C)} \left(\delta^{(C)}\right)^t$$

2. **Environmental Risk Minimization (optional secondary goal):**

$$f_{\text{env}}^{(C)} = -\mathbb{E}\left[\sum_{t=0}^{T} \mathbf{1}_{\{X_t > X_{\text{crit}}(b,q)\}}\right]$$

or equivalently, minimize the probability of eutrophication that would damage local livelihoods.

Community's Optimization Problem

$$\begin{aligned} \max_{\{a_t^{(C)}\}} \quad & \left(f_{\text{econ}}^{(C)},\ f_{\text{env}}^{(C)}\right) \\ \text{s.t.} \quad & X_{t+1} = X_t + a_t^{(C)} + a_t^{(R)} + \frac{X_t^q}{1+X_t^q} - bX_t + \epsilon_t, \\ & a_t^{(C)} \geq 0,\ X_t \geq 0. \end{aligned}$$

- Multi-objective formulation (no aggregation).
- May use Pareto-efficient trade-offs between economic growth and lake sustainability.
- Takes $a_t^{(R)}$ as given.

🧩 3. Perspective 2: Environmental Regulator

Goal

Protect the lake ecosystem, prevent eutrophication, and maintain long-term sustainability — possibly at the cost of short-term economic activity.

Control Variable

Symbol	Description
$a_t^{(R)}$	Regulator's intervention on pollution. Examples: taxes, abatement, restoration. Can be positive or negative.

State Variable

Symbol	Description
X_t	Shared pollution stock.

Transition Function (as perceived by regulator)

Same as for the community:

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

The regulator views $a_t^{(C)}$ as an *external input* determined by community behavior or policies.

Objective Functions

The regulator typically has **two or more objectives**:

1. **Environmental Preservation:**

$$f_{\text{env}}^{(R)} = -\mathbb{E}\left[\sum_{t=0}^{T}(X_t - X_{\text{crit}}(b,q))_+\right]$$

   where $(x)_+ = \max(0, x)$ penalizes pollution exceeding the critical threshold.

2. **Economic Welfare (Secondary Objective):**
   The regulator may also care about maintaining some local economic activity:

$$f_{\text{econ}}^{(R)} = \sum_{t=0}^{T} \alpha^{(R)} a_t^{(C)} (\delta^{(R)})^t$$

   reflecting tax revenues or shared welfare from economic output.

3. **Policy Cost Minimization (optional):**

$$f_{\text{cost}}^{(R)} = -\sum_{t=0}^{T} c^{(R)} (a_t^{(R)})^2 (\delta^{(R)})^t$$

   penalizing costly interventions.

Regulator's Optimization Problem

$$\begin{aligned}
\max_{\{a_t^{(R)}\}} \quad & \left(f_{\text{env}}^{(R)},\ f_{\text{econ}}^{(R)},\ f_{\text{cost}}^{(R)}\right) \\
\text{s.t.} \quad & X_{t+1} = X_t + a_t^{(C)} + a_t^{(R)} + \frac{X_t^q}{1+X_t^q} - bX_t + \epsilon_t, \\
& a_t^{(R)} \in [a_{\min}^{(R)}, a_{\max}^{(R)}].
\end{aligned}$$

🔄 4. Coupled Decision Environment (Game Formulation)

The **local community** and **regulator** make independent decisions each time step, influencing the shared environment.

Interaction Mechanism

- Both affect X_t through their emissions or abatement actions.
- The *state evolution* couples their objectives — actions by one alter the other's outcomes.
- They may act:

- **Sequentially (Stackelberg game):**
  - Regulator (leader) sets policy $a_t^{(R)}$, anticipating community response $a_t^{(C)}$.
- **Simultaneously (Nash game):**
  - Both choose $a_t^{(C)}$, $a_t^{(R)}$ independently, observing X_t.

🧠 5. Summary Tables

(a) Shared / Global Environment

Type	Symbol	Description
State	X_t	Lake pollution stock
Transition	$X_{t+1} = X_t + a_t^{(C)} + a_t^{(R)} + \frac{X_t^q}{1+X_t^q} - bX_t + \epsilon_t$	Shared dynamics
Stochastic	$\epsilon_t \sim \mathrm{LogNormal}(\mu, \sigma^2)$	Natural inflow
Deep Uncertainty	$\mu, \sigma, b, q, \delta^{(C)}, \delta^{(R)}$	Parameters with unknown true values
Threshold	$X_{\text{crit}}(b, q)$	Endogenous eutrophication level

---

(b) Local Community Model

Component	Symbol / Expression	Notes
Control variable	$a_t^{(C)} \geq 0$	Emission rate
Objective 1	$f_{\text{econ}}^{(C)} = \sum_t \alpha^{(C)} a_t^{(C)} (\delta^{(C)})^t$	Maximize economic benefit
Objective 2	$f_{\text{env}}^{(C)} = -\mathbb{E}\left[\sum_t \mathbf{1}_{\{X_t > X_{\text{crit}}\}}\right]$	Minimize eutrophication risk
Transition	$X_{t+1} = X_t + a_t^{(C)} + a_t^{(R)} + \frac{X_t^q}{1+X_t^q} - bX_t + \epsilon_t$	Shared with regulator

---

(c) Regulator Model

Component	Symbol / Expression	Notes
Control variable	$a_t^{(R)} \in [a_{\min}^{(R)}, a_{\max}^{(R)}]$	Abatement / policy
Objective 1	$f_{\text{env}}^{(R)} = -\mathbb{E}\left[\sum_t (X_t - X_{\text{crit}}(b, q))_+\right]$	Preserve ecosystem
Objective 2	$f_{\text{econ}}^{(R)} = \sum_t \alpha^{(R)} a_t^{(C)} (\delta^{(R)})^t$	Support sustainable economy
Objective 3	$f_{\text{cost}}^{(R)} = -\sum_t c^{(R)} (a_t^{(R)})^2 (\delta^{(R)})^t$	Minimize policy cost
Transition	Shared with community	Coupled dynamics

⚖️ 6. Conceptual Summary

- The lake pollution X_t is the **coupling variable** linking the agents.
- Each agent optimizes their objectives with **different priorities** under **shared uncertainty**.

- The system is a **stochastic dynamic game under deep uncertainty**.
- Key solution approaches:
  - **Dynamic Nash equilibrium** (simultaneous play),
  - **Stackelberg equilibrium** (leader–follower between regulator and community),
  - or **multi-objective robust optimization** for cooperative planning.

Would you like me to extend this into a **mathematical equilibrium formulation** (e.g., Stackelberg game with explicit best-response functions and equilibrium conditions) or keep it as a conceptual multi-agent model specification?

> 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.

Excellent — this is the right next step.
We'll now **compose** the multi-stakeholder lake problem into a **unified, modular, internally consistent system model** that:

1. Shares a **common environment and state dynamics**,
2. Embeds **both the local community and the environmental regulator** as distinct *decision modules*,
3. Allows each to be evaluated or optimized **independently**, while interacting through the shared state variable,
4. Retains **deep and stochastic uncertainty** consistently across all components.

🌍 1. Unified System Definition

We define a **shared dynamic stochastic environment** with **two interacting agents**, denoted $\mathcal{C}$ (local community) and $\mathcal{R}$ (environmental regulator).

Global Time Horizon

Discrete time: $t = 0, 1, \ldots, T$.

🧩 2. Common Environment (Global Subsystem)

Shared State Variable

Symbol	Description
X_t	Lake pollution stock at time t; common to all agents.

Stochastic Forcing

Symbol	Description
$\epsilon_t \sim \mathrm{LogNormal}(\mu, \sigma^2)$	Natural pollution inflow, representing **stochastic uncertainty**.

Deeply Uncertain Parameters (Global)

Symbol	Description
b	Natural removal rate of the lake.
q	Nonlinear recycling exponent.
μ, σ	Mean and standard deviation of stochastic natural inflow.
$\delta^{(C)}, \delta^{(R)}$	Stakeholder-specific discount rates.
$X_{\text{crit}}(b, q)$	Endogenous eutrophication threshold determined by $\frac{X_{\text{crit}}^q}{1+X_{\text{crit}}^q} = bX_{\text{crit}}$.

Define the set of all deep uncertainties:

$$\Theta = \{b, q, \mu, \sigma, \delta^{(C)}, \delta^{(R)}\}.$$

Global State Transition Function

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

This function is **common** to both perspectives — all controls feed into the same environmental state evolution.

👥 3. Embedded Stakeholder Submodels

The unified model embeds **two modular submodels**, each with its own control and objective(s), but coupled through the shared state X_t.

3.1 Local Community Module ($\mathcal{C}$)

Control Variable

Symbol	Description
$a_t^{(C)} \geq 0$	Community's anthropogenic pollution emissions (economic activity proxy).

Perceived Environment

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

with $a_t^{(R)}$ treated as **exogenous**.

Objective Functions

The community seeks to **maximize economic benefit** and **minimize ecological risk**:

1. **Economic Benefit Objective**

$$f_{\text{econ}}^{(C)} = \sum_{t=0}^{T} \alpha^{(C)} a_t^{(C)} (\delta^{(C)})^t$$

2. **Environmental Safety Objective**

$$f_{\text{env}}^{(C)} = -\mathbb{E}\left[\sum_{t=0}^{T} \mathbf{1}_{\{X_t > X_{\text{crit}}(b,q)\}}\right]$$

Community Submodel

$$\begin{aligned} \max_{\{a_t^{(C)}\}} \quad & \left(f_{\text{econ}}^{(C)},\ f_{\text{env}}^{(C)}\right) \\ \text{s.t.} \quad & X_{t+1} = X_t + a_t^{(C)} + a_t^{(R)} + \frac{X_t^q}{1 + X_t^q} - bX_t + \epsilon_t, \\ & a_t^{(C)} \geq 0, \quad X_t \geq 0. \end{aligned}$$

3.2 Environmental Regulator Module ($\mathcal{R}$)

Control Variable

Symbol	Description
$a_t^{(R)} \in [a_{\min}^{(R)}, a_{\max}^{(R)}]$	Regulator’s pollution intervention (abatement, restoration, or permitting).

Perceived Environment

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

with $a_t^{(C)}$ treated as **exogenous**.

Objective Functions

The regulator seeks to balance **environmental preservation**, **economic welfare**, and **policy cost**:

1. **Environmental Preservation**

$$f_{\text{env}}^{(R)} = -\mathbb{E}\left[\sum_{t=0}^{T}(X_t - X_{\text{crit}}(b,q))_+\right]$$

2. **Economic Welfare (from community activity)**

$$f_{\text{econ}}^{(R)} = \sum_{t=0}^{T} \alpha^{(R)} a_t^{(C)} (\delta^{(R)})^t$$

3. **Policy Cost**

$$f_{\text{cost}}^{(R)} = -\sum_{t=0}^{T} c^{(R)} (a_t^{(R)})^2 (\delta^{(R)})^t$$

Regulator Submodel

$$\begin{aligned}
\max_{\{a_t^{(R)}\}} \quad & \left(f_{\text{env}}^{(R)},\ f_{\text{econ}}^{(R)},\ f_{\text{cost}}^{(R)}\right) \\
\text{s.t.} \quad & X_{t+1} = X_t + a_t^{(C)} + a_t^{(R)} + \frac{X_t^q}{1+X_t^q} - bX_t + \epsilon_t, \\
& a_t^{(R)} \in [a_{\min}^{(R)}, a_{\max}^{(R)}].
\end{aligned}$$

🔗 4. Unified Composed Model (Integrating the Modules)

We now express the **composed model** that ties all components together.

System-Level Structure

$$\mathcal{M} = \Big(\mathcal{E},\ \mathcal{C},\ \mathcal{R},\ \Theta\Big)$$

where:

- $\mathcal{E} = \{X_t, \epsilon_t, b, q, X_{\text{crit}}(b,q)\}$: **shared environment**,
- $\mathcal{C} = \{a_t^{(C)}, f_{\text{econ}}^{(C)}, f_{\text{env}}^{(C)}\}$: **community module**,
- $\mathcal{R} = \{a_t^{(R)}, f_{\text{env}}^{(R)}, f_{\text{econ}}^{(R)}, f_{\text{cost}}^{(R)}\}$: **regulator module**,
- Θ: **deep uncertainty set**.

Coupled System Dynamics

$$\begin{cases}
X_{t+1} = X_t + a_t^{(C)} + a_t^{(R)} + \dfrac{X_t^q}{1+X_t^q} - bX_t + \epsilon_t, \\
\epsilon_t \sim \text{LogNormal}(\mu, \sigma^2), \\
X_0 = 0.
\end{cases}$$

Composition Logic

- **Shared State** X_t: single environmental trajectory for all agents.
- **Control Independence**: each agent selects its own control $a_t^{(i)}$ treating the other's control as given.
- **Modular Evaluation**: each module's objectives $f_\bullet^{(C)}$, $f_\bullet^{(R)}$ are evaluated separately.
- **Coupling** arises only through X_t.

Formally, the composed model defines a **two-agent stochastic dynamic game** under deep uncertainty:

$$\mathcal{M}: \begin{cases} \text{Agents: } \{\mathcal{C}, \mathcal{R}\} \\ \text{State: } X_t \in \mathbb{R}_+ \\ \text{Controls: } a_t^{(C)}, a_t^{(R)} \\ \text{Transition: } X_{t+1} = f(X_t, a_t^{(C)}, a_t^{(R)}, \epsilon_t; b, q) \\ \text{Objectives: } f_\bullet^{(C)},\ f_\bullet^{(R)} \\ \text{Uncertainties: } \Theta \end{cases}$$

⚖️ 5. Internal Consistency Properties

Property	How It's Ensured
Shared environment	Both agents use same X_t, same transition function, same stochastic inflow.
Perspective-specific control	Each has its own control variable and objectives.
Mutual influence	Controls enter the same state equation, linking outcomes.
Modular evaluation	Each perspective can be simulated or optimized independently by treating the other's actions as fixed inputs.
Deep uncertainty consistency	All uncertain parameters $(b, q, \mu, \sigma, \delta^{(C)}, \delta^{(R)})$ are shared draws across modules.
Endogenous threshold	$X_{\text{crit}}(b, q)$ used consistently in all objectives.

🔁 6. Optional Equilibrium or Scenario Evaluation Layer

Once the unified model $\mathcal{M}$ is specified, analysis can proceed in different modes:

- **Policy simulation:** iterate both modules with pre-specified strategies.
- **Nash equilibrium:** solve for $(a_t^{(C)*}, a_t^{(R)*})$ where neither can unilaterally improve its objectives.
- **Stackelberg equilibrium:** regulator as leader, community as follower.
- **Cooperative Pareto analysis:** joint exploration of multi-objective trade-offs.

🧠 7. Summary Schema

Level	Elements	Description
Global Environment	$X_t, b, q, \epsilon_t, X_{\text{crit}}(b,q)$	Shared ecological dynamics
Local Community ($\mathscr{C}$)	Control: $a_t^{(C)}$	Emissions decision
	Objectives: $f_{\text{econ}}^{(C)}, f_{\text{env}}^{(C)}$	Economic gain vs. risk
Regulator ($\mathscr{R}$)	Control: $a_t^{(R)}$	Policy/abatement decision
	Objectives: $f_{\text{env}}^{(R)}, f_{\text{econ}}^{(R)}, f_{\text{cost}}^{(R)}$	Environment, economy, cost
Coupling Mechanism	$X_{t+1} = X_t + a_t^{(C)} + a_t^{(R)} + \frac{X_t^q}{1+X_t^q} - bX_t + \epsilon_t$	Shared dynamics
Uncertainty Set	$\Theta = \{b, q, \mu, \sigma, \delta^{(C)}, \delta^{(R)}\}$	Deep uncertainties

✅ Final Unified Model (Compact Form)

$$\boxed{\begin{aligned}
&\text{State: } X_{t+1} = X_t + a_t^{(C)} + a_t^{(R)} + \frac{X_t^q}{1+X_t^q} - bX_t + \epsilon_t,\\
&\text{where } \epsilon_t \sim \text{LogNormal}(\mu, \sigma^2), \quad X_0 = 0, \quad X_{\text{crit}}(b,q) : \frac{X_{\text{crit}}^q}{1+X_{\text{crit}}^q} = bX_{\text{crit}}.\\
&\text{Community } (\mathcal{C}) : \begin{cases} \text{Control: } a_t^{(C)} \geq 0, \\ \text{Objectives: } \begin{aligned} f_{\text{econ}}^{(C)} &= \sum_t \alpha^{(C)} a_t^{(C)} (\delta^{(C)})^t, \\ f_{\text{env}}^{(C)} &= -\mathbb{E}\Big[\sum_t \mathbf{1}_{\{X_t > X_{\text{crit}}\}}\Big]. \end{aligned} \end{cases}\\
&\text{Regulator } (\mathcal{R}) : \begin{cases} \text{Control: } a_t^{(R)} \in [a_{\min}^{(R)}, a_{\max}^{(R)}], \\ \text{Objectives: } \begin{aligned} f_{\text{env}}^{(R)} &= -\mathbb{E}\Big[\sum_t (X_t - X_{\text{crit}})_+\Big], \\ f_{\text{econ}}^{(R)} &= \sum_t \alpha^{(R)} a_t^{(C)} (\delta^{(R)})^t, \\ f_{\text{cost}}^{(R)} &= -\sum_t c^{(R)} (a_t^{(R)})^2 (\delta^{(R)})^t. \end{aligned} \end{cases}
\end{aligned}}$$

This **unified model** provides a single, internally consistent system that:

- Shares all environmental dynamics and uncertainties,
- Embeds both perspectives modularly,
- Allows independent evaluation or optimization of each module, and
- Preserves the capacity to analyze interactive, equilibrium, or cooperative outcomes.

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.

Perfect — below is a **clean, modular Python implementation** of the **unified stochastic–dynamic lake problem** with two stakeholder perspectives:

- The **Local Community**, and
- The **Environmental Regulator**.

The design follows your requirements:
✅ Shared environment and dynamics
✅ Separate class modules for each perspective
✅ Each class has its own controls and objectives
✅ A unified interface function accepts control & uncertainty dictionaries, simulates the dynamics, and returns all objectives.

```python
import numpy as np

# ============================================================
# === Global Environment Definition ==========================
# ============================================================

class LakeEnvironment:
    """
    Shared environmental dynamics for the Lake Problem.
    Handles state evolution under given controls and uncertainties.
    """

    def __init__(self, b, q, mu, sigma, X0=0.0, T=50, seed=None):
        self.b = b                    # natural removal rate
        self.q = q                    # recycling rate
        self.mu = mu                  # mean of lognormal inflow
        self.sigma = sigma            # std dev of lognormal inflow
        self.T = T                    # time horizon
        self.X0 = X0                  # initial pollution
        self.rng = np.random.default_rng(seed)

        # compute endogenous threshold Xcrit(b,q)
        self.Xcrit = self.compute_Xcrit()

    def compute_Xcrit(self, tol=1e-8, max_iter=1000):
        """
        Compute Xcrit as the solution to X^q / (1 + X^q) = b * X
        using fixed-point iteration or bisection.
        """
        low, high = 0.0, 1e3
        for _ in range(max_iter):
            mid = 0.5 * (low + high)
            f_mid = (mid**self.q) / (1 + mid**self.q) - self.b * mid
            if abs(f_mid) < tol:
                return mid
            if f_mid > 0:
                high = mid
            else:
                low = mid
        return mid  # fallback

    def step(self, X_t, a_C, a_R):
        """
        One-step state transition given current state and controls.
        """
        eps_t = self.rng.lognormal(mean=self.mu, sigma=self.sigma)
```

```
        X_next = X_t + a_C + a_R + (X_t**self.q)/(1 + X_t**self.q) - self.b * X_t + eps_t
        return X_next, eps_t

    def simulate(self, a_C_series, a_R_series):
        """
        Simulate pollution dynamics given control trajectories.
        Returns time series of X_t and epsilon_t.
        """
        X = np.zeros(self.T + 1)
        eps = np.zeros(self.T)
        X[0] = self.X0

        for t in range(self.T):
            X[t+1], eps[t] = self.step(X[t], a_C_series[t], a_R_series[t])

        return X, eps


# ============================================================
# === Local Community Module ===============================
# ============================================================

class LocalCommunity:
    """
    Represents the local community perspective.
    Controls emissions and seeks to balance economic benefit and environmental safety.
    """

    def __init__(self, alpha_C, delta_C):
        self.alpha = alpha_C        # benefit-to-pollution ratio
        self.delta = delta_C        # discount rate

    def economic_benefit(self, a_C_series):
        """
        Compute discounted economic benefit.
        """
        T = len(a_C_series)
        return np.sum([self.alpha * a_C_series[t] * (self.delta**t) for t in range(T)])

    def environmental_safety(self, X_series, Xcrit):
        """
        Compute negative expected frequency of eutrophication events.
        """
        exceed = np.sum(X_series > Xcrit)
        return -exceed


# ============================================================
# === Environmental Regulator Module ========================
# ============================================================

class EnvironmentalRegulator:
    """
    Represents the environmental regulator's perspective.
    Controls pollution abatement and seeks to preserve environment,
    support economy, and minimize intervention costs.
    """

    def __init__(self, alpha_R, delta_R, c_R):
        self.alpha = alpha_R        # economic weight (e.g., shared welfare)
        self.delta = delta_R        # discount rate
        self.c_R = c_R              # cost coefficient

    def environmental_preservation(self, X_series, Xcrit):
        """
        Penalize pollution exceeding the critical threshold.
        """
        exceed_amount = np.maximum(0, X_series - Xcrit)
        return -np.sum(exceed_amount)

    def economic_welfare(self, a_C_series):
        """
        Proxy for total economic welfare (benefits from community activity).
        """
        T = len(a_C_series)
```

```
        return np.sum([self.alpha * a_C_series[t] * (self.delta**t) for t in range(T)])

    def policy_cost(self, a_R_series):
        """
        Penalize large intervention magnitudes.
        """
        T = len(a_R_series)
        return -np.sum([self.c_R * (a_R_series[t]**2) * (self.delta**t) for t in range(T)])


# ============================================================
# === Unified Model Interface ================================
# ============================================================

def simulate_lake_problem(controls, uncertainties, T=50, seed=None):
    """
    Unified interface function for simulating the multi-perspective Lake Problem.

    Parameters
    ----------
    controls : dict
        Dictionary with keys:
            - 'a_C': np.ndarray of community emissions (len T)
            - 'a_R': np.ndarray of regulator actions (len T)

    uncertainties : dict
        Dictionary of uncertain parameter values:
            - 'b': natural removal rate
            - 'q': recycling exponent
            - 'mu': mean of lognormal inflow
            - 'sigma': std dev of lognormal inflow
            - 'alpha_C': benefit-to-pollution ratio (community)
            - 'delta_C': discount rate (community)
            - 'alpha_R': economic weight (regulator)
            - 'delta_R': discount rate (regulator)
            - 'c_R': policy cost coefficient

    T : int
        Simulation horizon (default 50)

    Returns
    -------
    results : dict
        {
            'community': {
                'economic_benefit': ...,
                'environmental_safety': ...
            },
            'regulator': {
                'environmental_preservation': ...,
                'economic_welfare': ...,
                'policy_cost': ...
            },
            'state': {
                'X': np.ndarray,
                'epsilon': np.ndarray,
                'Xcrit': float
            }
        }
    """

    # Unpack uncertainties
    b = uncertainties["b"]
    q = uncertainties["q"]
    mu = uncertainties["mu"]
    sigma = uncertainties["sigma"]

    alpha_C = uncertainties["alpha_C"]
    delta_C = uncertainties["delta_C"]
    alpha_R = uncertainties["alpha_R"]
    delta_R = uncertainties["delta_R"]
    c_R = uncertainties["c_R"]

    # Instantiate shared environment
    env = LakeEnvironment(b=b, q=q, mu=mu, sigma=sigma, T=T, seed=seed)
```

```
    # Instantiate agents
    community = LocalCommunity(alpha_C, delta_C)
    regulator = EnvironmentalRegulator(alpha_R, delta_R, c_R)

    # Extract controls
    a_C_series = np.asarray(controls["a_C"])
    a_R_series = np.asarray(controls["a_R"])

    # Simulate environment
    X_series, eps_series = env.simulate(a_C_series, a_R_series)

    # Compute objectives
    results = {
        "community": {
            "economic_benefit": community.economic_benefit(a_C_series),
            "environmental_safety": community.environmental_safety(X_series, env.Xcrit)
        },
        "regulator": {
            "environmental_preservation": regulator.environmental_preservation(X_series,
env.Xcrit),
            "economic_welfare": regulator.economic_welfare(a_C_series),
            "policy_cost": regulator.policy_cost(a_R_series)
        },
        "state": {
            "X": X_series,
            "epsilon": eps_series,
            "Xcrit": env.Xcrit
        }
    }

    return results
```

🧠 Notes on Implementation

- **Modularity:**
  Each stakeholder is encapsulated in its own class (**`LocalCommunity`**, **`EnvironmentalRegulator`**) for independent evaluation or optimization.
- **Shared Dynamics:**
  The class **`LakeEnvironment`** maintains all global state transitions and deep uncertainty parameters.
- **Interface Function:**
  `simulate_lake_problem()` is the **single entry point**.
  It receives:
  - A dictionary of **controls** (**`a_C`**, **`a_R`**),
  - A dictionary of **uncertain parameters**,
    then returns all **objective values and simulation traces** in a structured dictionary.
- **Stochasticity:**
  The natural inflow **`ε_t`** is drawn from a lognormal distribution each time step, controlled by **`mu`** and **`sigma`**.
- **Endogenous Threshold:**
  `Xcrit` is computed numerically as the pollution level where recycling equals natural removal.

✅ Example of Usage (not executed)

```python
controls = {
    "a_C": np.full(50, 0.5),    # constant community emission
    "a_R": np.full(50, -0.1)    # constant abatement effort
}

uncertainties = {
    "b": 0.4,
    "q": 2.0,
    "mu": 0.1,
    "sigma": 0.05,
    "alpha_C": 1.0,
    "delta_C": 0.95,
    "alpha_R": 0.5,
    "delta_R": 0.97,
    "c_R": 0.2
}

# Simulate model (function not executed)
# results = simulate_lake_problem(controls, uncertainties)
```

This design yields a **clean, extensible foundation** suitable for:

- Scenario analysis under deep uncertainty,
- Multi-agent optimization (Nash or Stackelberg games),
- Multi-objective evaluation,
  while keeping environment–stakeholder coupling explicit and modular.

> 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 — both of your improvement points are exactly right.
Let's refine the implementation carefully:

🩹 Improvement 1 — Refine computation of X_{crit}

- The physically meaningful range of the lake's critical pollution threshold is $0.01, 1.5$.
- Instead of manual bisection, we should use a **numerical root-finding routine** (e.g. `scipy.optimize.root_scalar`) over that interval.
- The equation we solve is:

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

🩹 Improvement 2 — Correct the lognormal noise implementation

- In **`numpy.random.lognormal(mean, sigma)`**, the **`mean`** and **`sigma`** are **parameters of the underlying normal distribution**, not the lognormal's mean and standard deviation.
- To sample from a lognormal distribution with given *mean* μ and *variance* σ^2, we need to convert them to the underlying normal parameters (m, s):

$$s^2 = \ln\left(1 + \frac{\sigma^2}{\mu^2}\right), \qquad m = \ln(\mu) - \frac{s^2}{2}.$$

- Then we draw:

$$\epsilon_t \sim \mathrm{LogNormal}(m, s).$$

Here is the **refined and corrected unified implementation**:

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

# ==============================================================
# === Shared Environment =======================================
# ==============================================================

class LakeEnvironment:
    """
    Shared environmental system for the Lake Problem.
    Handles pollution dynamics, natural stochastic inflow, and
    endogenous eutrophication threshold.
    """

    def __init__(self, b, q, mu, sigma, X0=0.0, T=50, seed=None):
        self.b = b
        self.q = q
        self.mu = mu            # desired lognormal mean
        self.sigma = sigma      # desired lognormal std (variance = sigma^2)
        self.T = T
        self.X0 = X0
        self.rng = np.random.default_rng(seed)
        self.Xcrit = self.compute_Xcrit()  # within [0.01, 1.5]

    # ----------------------------------------------------------
    # Compute Xcrit using root-finding in [0.01, 1.5]
    # ----------------------------------------------------------
    def compute_Xcrit(self):
        """
        Compute Xcrit in [0.01, 1.5] such that
        f(X) = X^q / (1 + X^q) - b * X = 0.
        Returns root or np.nan if not found.
        """
        def f(X):
            return (X**self.q) / (1 + X**self.q) - self.b * X

        try:
            sol = root_scalar(f, bracket=[0.01, 1.5], method="bisect")
            if sol.converged:
                return sol.root
            else:
                return np.nan
        except ValueError:
            # if no sign change in bracket
            return np.nan

    # ----------------------------------------------------------
    # Draw lognormal natural inflow with correct parameterization
    # ----------------------------------------------------------
    def draw_lognormal(self):
```

```
        """
        Draw epsilon_t ~ LogNormal(mean=mu, std=sigma)
        using proper conversion from lognormal mean/std
        to underlying normal parameters.
        """
        # Avoid invalid values
        mu = max(self.mu, 1e-12)
        sigma = max(self.sigma, 1e-12)

        s2 = np.log(1 + (sigma**2) / (mu**2))
        s = np.sqrt(s2)
        m = np.log(mu) - 0.5 * s2

        return self.rng.lognormal(mean=m, sigma=s)

    # ---------------------------------------------------------
    # One-step transition
    # ---------------------------------------------------------
    def step(self, X_t, a_C, a_R):
        eps_t = self.draw_lognormal()
        X_next = X_t + a_C + a_R + (X_t**self.q) / (1 + X_t**self.q) - self.b * X_t + eps_t
        return X_next, eps_t

    # ---------------------------------------------------------
    # Full simulation
    # ---------------------------------------------------------
    def simulate(self, a_C_series, a_R_series):
        X = np.zeros(self.T + 1)
        eps = np.zeros(self.T)
        X[0] = self.X0

        for t in range(self.T):
            X[t + 1], eps[t] = self.step(X[t], a_C_series[t], a_R_series[t])

        return X, eps


# ==============================================================
# === Local Community ==========================================
# ==============================================================

class LocalCommunity:
    """
    Local community perspective: economic benefit vs. eutrophication risk.
    """
    def __init__(self, alpha_C, delta_C):
        self.alpha = alpha_C
        self.delta = delta_C

    def economic_benefit(self, a_C_series):
        T = len(a_C_series)
        return np.sum([self.alpha * a_C_series[t] * (self.delta**t) for t in range(T)])

    def environmental_safety(self, X_series, Xcrit):
        exceed = np.sum(X_series > Xcrit)
        return -exceed


# ==============================================================
# === Environmental Regulator ==================================
# ==============================================================

class EnvironmentalRegulator:
    """
    Environmental regulator perspective: preservation, welfare, and policy cost.
    """
    def __init__(self, alpha_R, delta_R, c_R):
        self.alpha = alpha_R
        self.delta = delta_R
        self.c_R = c_R

    def environmental_preservation(self, X_series, Xcrit):
        exceed_amount = np.maximum(0, X_series - Xcrit)
        return -np.sum(exceed_amount)
```

```
    def economic_welfare(self, a_C_series):
        T = len(a_C_series)
        return np.sum([self.alpha * a_C_series[t] * (self.delta**t) for t in range(T)])

    def policy_cost(self, a_R_series):
        T = len(a_R_series)
        return -np.sum([self.c_R * (a_R_series[t]**2) * (self.delta**t) for t in range(T)])


# ============================================================
# === Unified Interface Function =============================
# ============================================================

def simulate_lake_problem(controls, uncertainties, T=50, seed=None):
    """
    Unified multi-perspective Lake Problem simulator.

    Parameters
    ----------
    controls : dict
        'a_C' : np.ndarray, community emissions (length T)
        'a_R' : np.ndarray, regulator interventions (length T)

    uncertainties : dict
        {
          'b': float,
          'q': float,
          'mu': float,
          'sigma': float,
          'alpha_C': float,
          'delta_C': float,
          'alpha_R': float,
          'delta_R': float,
          'c_R': float
        }

    Returns
    -------
    results : dict
        {
          'community': {
              'economic_benefit': float,
              'environmental_safety': float
          },
          'regulator': {
              'environmental_preservation': float,
              'economic_welfare': float,
              'policy_cost': float
          },
          'state': {
              'X': np.ndarray,
              'epsilon': np.ndarray,
              'Xcrit': float
          }
        }
    """

    # Unpack uncertainties
    b = uncertainties["b"]
    q = uncertainties["q"]
    mu = uncertainties["mu"]
    sigma = uncertainties["sigma"]

    alpha_C = uncertainties["alpha_C"]
    delta_C = uncertainties["delta_C"]
    alpha_R = uncertainties["alpha_R"]
    delta_R = uncertainties["delta_R"]
    c_R = uncertainties["c_R"]

    # Initialize shared environment
    env = LakeEnvironment(b=b, q=q, mu=mu, sigma=sigma, T=T, seed=seed)

    # Initialize stakeholders
    community = LocalCommunity(alpha_C, delta_C)
    regulator = EnvironmentalRegulator(alpha_R, delta_R, c_R)
```

```
    # Extract controls
    a_C_series = np.asarray(controls["a_C"])
    a_R_series = np.asarray(controls["a_R"])

    # Simulate shared environment
    X_series, eps_series = env.simulate(a_C_series, a_R_series)

    # Evaluate objectives
    results = {
        "community": {
            "economic_benefit": community.economic_benefit(a_C_series),
            "environmental_safety": community.environmental_safety(X_series, env.Xcrit)
        },
        "regulator": {
            "environmental_preservation": regulator.environmental_preservation(X_series,
env.Xcrit),
            "economic_welfare": regulator.economic_welfare(a_C_series),
            "policy_cost": regulator.policy_cost(a_R_series)
        },
        "state": {
            "X": X_series,
            "epsilon": eps_series,
            "Xcrit": env.Xcrit
        }
    }

    return results
```

✅ Key Refinements Summary

Improvement	Description	Implementation
Bounded root-finding	Ensures $X_{\text{crit}} \in [0.01, 1.5]$ and finds root robustly.	**`scipy.optimize.root_scalar()`** with bracket **`[0.01, 1.5]`**
Correct lognormal sampling	Converts mean–std pair to lognormal's normal parameters.	Function **`draw_lognormal()`**
Modular classes preserved	Environment, Community, and Regulator remain independent.	Each still has its own class
Unified interface	**`simulate_lake_problem()`** integrates all components and returns all objectives.	Structured output dictionary

---

Example (not executed)

```python
controls = {
    "a_C": np.full(50, 0.4),
    "a_R": np.full(50, -0.1)
}

uncertainties = {
    "b": 0.45,
    "q": 2.5,
    "mu": 0.1,
    "sigma": 0.03,
    "alpha_C": 1.0,
    "delta_C": 0.95,
    "alpha_R": 0.6,
    "delta_R": 0.97,
    "c_R": 0.2
}
```

```
# results = simulate_lake_problem(controls, uncertainties)
```

This final implementation is **fully modular, numerically robust, and physically consistent** with the lake problem dynamics under both stochastic and deep uncertainties.