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 natural removal rate of the pollution; q is the natural recycling rate of the pollution; ϵ_t is the rate of natural pollution at time t. 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 parameter uncertainty. The natural pollution ϵ_t follows a log-normal distribution with mean μ and variance σ^2. In contrast, the probability distributions of the parameters μ, σ, b, q, δ cannot be determined.

Please formalize this problem description as a simulation model and specify the key components, including state variables, decision variables, transition functions, objective functions, stochastic variables, constant parameters and deep uncertainties.

Below is a **formal simulation-model specification** of the lake problem, written in a way that can be directly used for computational experiments, optimization, or exploratory modeling under deep uncertainty.

---

1. Model Type

- **Type:** Discrete-time, stochastic, nonlinear dynamical system
- **Purpose:** Sequential decision-making under risk and deep uncertainty
- **Time horizon:**

$$t = 0, 1, \ldots, T$$

---

2. State Variables

Symbol	Description	Domain
X_t	Pollution concentration in the lake at time t	$X_t \geq 0$

Initial condition

$$X_0 = 0$$

---

3. Decision Variables (Controls)

Symbol	Description	Domain
a_t	Anthropogenic pollution emission rate at time t	$a_t \geq 0$

The decision-maker selects a_t at each time step.

---

4. Transition Function (System Dynamics)

The pollution evolves according to:

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

Interpretation of terms

- X_t: persistence of existing pollution
- a_t: controllable anthropogenic emissions
- $\frac{X_t^q}{1+X_t^q}$: nonlinear internal recycling (positive feedback)
- $-bX_t$: natural removal
- ϵ_t: stochastic natural inflow

---

5. Threshold and Irreversibility

Critical pollution level

$$X_{\text{crit}} : \frac{X^q}{1 + X^q} = bX$$

Constraint

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

Exceeding X_{crit} results in **irreversible eutrophication** and catastrophic loss, typically modeled as:

- infeasible states, or
- terminal penalty / absorbing failure state.

---

6. Objective Function

The decision-maker maximizes discounted economic benefits from pollution-generating activities:

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

Parameters

- $\alpha > 0$: benefit-to-pollution ratio
- $\delta \in (0,1)$: discount factor

---

7. Stochastic Variables (Aleatory Uncertainty)

Natural pollution inflow

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

- i.i.d. over time
- Represents uncontrollable environmental variability

---

8. Constant Parameters (Known or Assumed Fixed per Scenario)

Parameter	Meaning
α	Economic benefit per unit pollution
T	Planning horizon
X_{crit}	Critical pollution threshold

---

9. Deep Uncertainties (Epistemic Uncertainty)

The following parameters are **deeply uncertain**, meaning no agreed-upon probability distributions exist:

Parameter	Interpretation
μ	Mean of natural pollution
σ	Variability of natural pollution

Parameter	Interpretation
b	Natural removal rate
q	Recycling nonlinearity
δ	Discount rate

These are typically treated as:

- scenario parameters, or
- exploratory uncertainty ranges.

10. Simulation Model Summary

State:

$$X_t$$

Control:

$$a_t$$

Stochastic input:

$$\epsilon_t$$

Dynamics:

$$X_{t+1} = g(X_t, a_t, \epsilon_t; b, q)$$

Objective:

$$\max \ \mathbb{E}\left[\sum_t \alpha a_t \delta^t\right] \quad \text{s.t. } X_t < X_{\text{crit}}$$

Uncertainty treatment:

- Aleatory: ϵ_t
- Deep: $\mu, \sigma, b, q, \delta$

11. Typical Uses of This Formalization

- Robust decision-making (RDM)
- Dynamic optimization with constraints
- Scenario discovery under deep uncertainty
- Policy stress-testing (e.g., constant vs adaptive a_t)

If you'd like, I can also:

- express this as a **Markov Decision Process (MDP)**,
- write **pseudocode** for simulation,
- or show how this model is used in **robust control / exploratory modeling frameworks**.

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

You're right — thank you for the correction.
Below is a **refined and internally consistent model specification** in which

$$X_{\text{crit}}$$

is **endogenous** and depends on deeply uncertain parameters.

---

1. Model Type (unchanged)

- Discrete-time, stochastic, nonlinear dynamical system
- Sequential decision-making under **deep uncertainty**
- Time index: $t = 0, 1, \ldots, T$

---

2. State Variables

Symbol	Description	Domain
X_t	Pollution concentration in the lake at time t	$X_t \geq 0$

Initial condition

$$X_0 = 0$$

---

3. Decision Variables

Symbol	Description	Domain
a_t	Anthropogenic pollution emissions at time t	$a_t \geq 0$

The decision-maker selects a_t sequentially.

---

4. Transition Function (Dynamics)

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

Where:

- $b > 0$: natural removal rate (**deeply uncertain**)
- $q > 1$: recycling nonlinearity (**deeply uncertain**)

---

5. Endogenous Critical Threshold $X_{\text{crit}}(b, q)$

Definition

The critical pollution level is defined **implicitly** as the point where **natural recycling equals natural removal**:

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

This equation generally:

- has **multiple equilibria**, and
- must be solved numerically for each (b, q) realization.

Key implication

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

Therefore:

- X_{crit} is **not fixed**,
- it varies across scenarios,
- and it is **unknown to the decision-maker ex ante** under deep uncertainty.

---

6. Irreversibility and System Failure

Eutrophication condition

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

Once exceeded:

- the lake transitions to an **irreversible eutrophic regime**, or
- the system enters an **absorbing failure state**.

This is typically modeled as:

- termination of the simulation, or
- a catastrophic penalty in the objective function.

---

7. Objective Function

The decision-maker maximizes expected discounted economic benefits:

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

subject to:

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

Where:

- $\alpha > 0$: benefit-to-pollution ratio
- $\delta \in (0, 1)$: discount factor (**deeply uncertain**)

---

8. Stochastic Variables (Aleatory Uncertainty)

Natural pollution inflow

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

- i.i.d. across time
- Represents uncontrollable environmental variability

---

9. Deep Uncertainties (Epistemic)

No probability distributions are assumed for the following parameters:

Parameter	Role
μ	Mean of natural pollution
σ	Variability of natural pollution
b	Natural removal rate
q	Recycling nonlinearity
δ	Discount factor

These parameters:

- define alternative **plausible worlds**, and
- jointly determine both **system dynamics** and **failure thresholds**.

---

10. Scenario-Based Representation

Each **scenario** θ is defined as:

$$\theta = (\mu, \sigma, b, q, \delta)$$

For each scenario:

1. Compute $X_{\text{crit}}(b, q)$
2. Simulate the stochastic system
3. Evaluate performance of policy $\{a_t\}$

---

11. Final Compact Formulation

State evolution

$$X_{t+1} = g(X_t, a_t, \epsilon_t \mid b, q)$$

Failure boundary

$$\mathcal{F}(b, q) = \{X : X \geq X_{\text{crit}}(b, q)\}$$

Objective

$$\max_{\{a_t\}} \ \mathbb{E}_\epsilon \left[\sum_t \alpha a_t \delta^t \right] \quad \text{s.t. } X_t \notin \mathcal{F}(b, q)$$

---

12. Conceptual Implication

This refinement highlights why the lake problem is a **canonical example of deep uncertainty**:

- The **location of the tipping point is itself uncertain**
- Policies must be **robust to unknown thresholds**, not merely noisy dynamics

If you want, I can next:

- formalize this as a **robust MDP with uncertain constraints**, or
- show how this model is used in **Exploratory Modeling & Analysis (EMA)** or **Robust Decision Making (RDM)** frameworks.

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 decision variables. In the specification, specify the common environment, including the global parameters and variables. For each perspective, also specify the corresponding model and its key components, including the decision variables, transition functions, and objective functions. If there are multiple objectives, provide multiple objective functions instead of a weighted objective function.

Below is a **multi-stakeholder refinement** of the lake problem, formulated as a **shared stochastic environment** with **two decision-makers** who control **distinct decision variables**, observe the same system state, and pursue **different objectives**. This structure is consistent with multi-agent control, institutional analysis, and robust decision-making under deep uncertainty.

I. Common Environment (Shared by All Stakeholders)

This defines the **biophysical lake system** and the uncertain context.
Both stakeholders act *within* this environment but do not control it fully.

---

1. Time and State

- **Time:** $t = 0, 1, \dots, T$
- **State variable:**

$$X_t \geq 0 \quad \text{(lake pollution concentration)}$$

- **Initial condition:**

$$X_0 = 0$$

---

2. Environmental Dynamics (Shared Transition Function)

The lake pollution evolves as:

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

Where:

- a_t: anthropogenic pollution emissions (community-controlled)
- r_t: pollution reduction/remediation effort (regulator-controlled)

- b: natural removal rate (deeply uncertain)
- q: recycling nonlinearity (deeply uncertain)
- ϵ_t: stochastic natural pollution inflow

3. Stochastic Inputs (Aleatory Uncertainty)

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

4. Deeply Uncertain Parameters (Shared)

No probability distributions are assumed for:

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

These parameters define alternative plausible environments.

5. Endogenous Tipping Point (Shared)

The critical pollution threshold depends on uncertain biophysical parameters:

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

6. Failure Condition (Shared)

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

Once crossed:

- eutrophication is **irreversible**
- system enters an absorbing failure state

II. Stakeholder 1: Local Community Perspective

The local community prioritizes **economic development**, while being exposed to environmental risks and regulatory actions.

1. Decision Variable (Community Control)

$$a_t \geq 0$$

- Represents industrial and agricultural pollution emissions

- Chosen sequentially based on observed X_t

2. Community Model

State (Observed)

$$X_t$$

Control

$$a_t$$

Transition Function

The community **does not control** the full dynamics but recognizes:

$$X_{t+1} = g(X_t, a_t, r_t, \epsilon_t \mid b, q)$$

where r_t is taken as exogenous from the community's perspective.

3. Community Objectives (Multiple Objectives)

Objective 1: Maximize Economic Benefits

$$\max \ J_1^C = \mathbb{E}\left[\sum_{t=0}^{T} \alpha a_t \delta^t\right]$$

Objective 2: Avoid Economic Collapse from Eutrophication

This is a **satisficing / constraint objective**, not a weighted term:

$$J_2^C: \quad X_t < X_{\text{crit}}(b, q) \quad \forall t$$

or equivalently:

- minimize probability of eutrophication,
- or avoid terminal failure states.

4. Interpretation

From the community's perspective:

- Pollution creates **private economic gains**
- The tipping point is **uncertain and catastrophic**
- Regulation and natural dynamics are external constraints

III. Stakeholder 2: Environmental Regulator Perspective

The regulator prioritizes **environmental protection and long-term system viability**, potentially at economic or fiscal cost.

---

1. Decision Variable (Regulator Control)

$$r_t \geq 0$$

Where:

- r_t represents pollution abatement, cleanup, enforcement, or restoration
- Costs scale with intensity

---

2. Regulator Model

State (Observed)

$$X_t$$

Control

$$r_t$$

Transition Function

The regulator recognizes the same environment:

$$X_{t+1} = g(X_t, a_t, r_t, \epsilon_t \mid b, q)$$

with a_t treated as exogenous (community behavior).

---

3. Regulator Objectives (Multiple Objectives)

Objective 1: Prevent Eutrophication

$$J_1^R : \quad X_t < X_{\text{crit}}(b, q) \quad \forall t$$

This is a **primary, non-compensatory objective**.

Objective 2: Minimize Pollution Stock

$$\min \ J_2^R = \mathbb{E}\left[\sum_{t=0}^{T} X_t\right]$$

Objective 3: Minimize Regulatory Cost

$$\min \; J_3^R = \mathbb{E}\left[\sum_{t=0}^{T} c(r_t)\right]$$

Where:

- $c(\cdot)$ is an increasing convex cost function
- reflects budgetary and political constraints

---

4. Interpretation

From the regulator's perspective:

- The community is a **strategic actor**
- Environmental collapse is unacceptable
- Costs must be justified but cannot trade off against system failure

---

IV. Integrated View

Component	Shared	Community	Regulator
State X_t	✅	observes	observes
Emissions a_t	❌	controls	exogenous
Remediation r_t	❌	exogenous	controls
Dynamics $g(\cdot)$	✅	partial	partial
Deep uncertainty	✅	shared	shared
Objectives	❌	economic + safety	environmental + cost

V. Conceptual Framing

This formulation supports:

- **Multi-agent MDPs**
- **Institutional and governance analysis**
- **Robust decision-making with interacting policies**
- **Exploratory modeling under deep uncertainty**

If you want, I can next:

- formalize this as a **game-theoretic model**,

- define **policy classes** for each stakeholder,
- or show how to analyze **robust compromises** between the two perspectives.

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 decisions and objectives. The composed model should: 1. embed each perspective's decision variables and objectives; 2. enable modular evaluation of each perspective; 3. for each perspective, treat other perspectives' decisions as exogenous inputs; and 4. incorporate all other components from every perspective.

Below is a **single, unified, internally consistent composed model** that embeds both stakeholder perspectives into **one shared stochastic environment**, while preserving **modularity, perspective separation, and evaluability**. The structure follows principles from multi-agent control, modular MDPs, and exploratory modeling.

Unified Composed Lake Model (Multi-Perspective, Shared Environment)

1. Common Environment (Global, Shared by All Perspectives)

The environment represents the **true biophysical system**.
All perspectives observe and act within this environment, but **no single perspective controls it fully**.

1.1 Time and State

- **Time index**

$$t = 0, 1, \ldots, T$$

- **Global state**

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

Pollution concentration in the lake at time t.

- **Initial condition**

$$X_0 = 0$$

1.2 Global Transition Function

The environment evolves according to:

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

Where:

- a_t: anthropogenic emissions (community decision)
- r_t: remediation / enforcement effort (regulator decision)
- b: natural removal rate (deep uncertainty)
- q: recycling nonlinearity (deep uncertainty)
- ϵ_t: stochastic natural pollution inflow

This function is **global and authoritative**.

1.3 Stochastic Variables (Aleatory Uncertainty)

$$\epsilon_t \sim \mathrm{LogNormal}(\mu, \sigma^2), \quad \text{i.i.d.}$$

1.4 Deeply Uncertain Parameters (Shared)

No probability distributions are assumed for:

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

Each realization of Θ defines a **plausible world**.

1.5 Endogenous Critical Threshold (Shared)

The eutrophication threshold is defined implicitly as:

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

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

1.6 Failure State (Shared)

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

Crossing this threshold:

- causes **irreversible eutrophication**
- places the system in an **absorbing failure state**

---

2. Perspective-Specific Embedded Models

Each perspective is represented as a **module** embedded in the unified environment.
Each module:

- observes the same global state X_t,

- controls its own decision variables,
- treats other decisions as **exogenous inputs**, and
- evaluates its own objectives independently.

3. Perspective A: Local Community Module

3.1 Decision Variable (Controlled)

$$a_t \geq 0$$

Anthropogenic pollution emissions from industry and agriculture.

3.2 Inputs Treated as Exogenous

From the community's perspective:

- r_t: regulator remediation effort
- ϵ_t: natural pollution
- (b, q, δ): deeply uncertain parameters

3.3 Perceived Transition Function

$$X_{t+1} = g(X_t, a_t \mid r_t, \epsilon_t, b, q)$$

The community does **not** control r_t, but its actions affect future states.

3.4 Community Objectives (Evaluated Modularly)

Objective C1: Maximize Economic Benefits

$$J_1^C = \mathbb{E}\left[\sum_{t=0}^{T} \alpha a_t \delta^t\right]$$

Objective C2: Avoid Eutrophication (Safety Objective)

$$J_2^C: \quad X_t < X_{\text{crit}}(b, q) \quad \forall t$$

This is a **non-compensatory objective** (constraint or failure avoidance).

4. Perspective B: Environmental Regulator Module

4.1 Decision Variable (Controlled)

$$r_t \geq 0$$

Pollution abatement, enforcement, or restoration effort.

4.2 Inputs Treated as Exogenous

From the regulator's perspective:

- a_t: community emissions
- ϵ_t: natural pollution
- (b, q, δ): deeply uncertain parameters

4.3 Perceived Transition Function

$$X_{t+1} = g(X_t, r_t \mid a_t, \epsilon_t, b, q)$$

The regulator does **not** control a_t.

4.4 Regulator Objectives (Evaluated Modularly)

Objective R1: Prevent Eutrophication

$$J_1^R : \quad X_t < X_{\text{crit}}(b, q) \quad \forall t$$

Primary, non-negotiable objective.

Objective R2: Minimize Pollution Stock

$$J_2^R = \mathbb{E}\left[\sum_{t=0}^{T} X_t\right]$$

Objective R3: Minimize Regulatory Cost

$$J_3^R = \mathbb{E}\left[\sum_{t=0}^{T} c(r_t)\right]$$

where $c(\cdot)$ is increasing and convex.

---

5. Composed Model Structure (Summary)

5.1 Global Composition

$$\boxed{\text{Environment } + \{\text{Community Module}, \text{ Regulator Module}\}}$$

- One **shared state**: X_t
- One **shared transition function**
- Multiple **independent objective evaluations**

5.2 Control Decomposition

Variable	Controlled by	Exogenous to
a_t	Community	Regulator
r_t	Regulator	Community
ϵ_t	Nature	All
b, q, δ	Uncertain	All

5.3 Evaluation Modularity

For any candidate pair of policies (π^C, π^R):

- simulate the **single shared trajectory** $\{X_t\}$,
- evaluate:
  - (J_1^C, J_2^C) for the community,
  - (J_1^R, J_2^R, J_3^R) for the regulator.

No objective aggregation is required.

6. Conceptual Payoff of This Composition

This unified model:

- ensures **internal consistency** (one environment, one state),
- supports **independent stakeholder evaluation**,
- allows **robustness analysis under deep uncertainty**,
- enables extensions to:
  - game-theoretic analysis,
  - adaptive policies,
  - institutional design,
  - robustness trade-off exploration.

If you want, I can next:

- express this as a **multi-agent MDP**,
- define **policy parameterizations** for each module, or
- show how to evaluate **robust compromises across stakeholder objectives** under deep uncertainty.

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 take as inputs a dictionary of decision variable values, a dictionary of uncertain parameter values, and a dictionary of constant parameter values; simulate the model dynamics; and return the value of each objective function. This function does not need to be executed.

Below is a **modular, unified Python implementation** of the composed lake model.
It uses **classes** to represent the shared environment and each stakeholder perspective, while ensuring:

- a single shared state and transition function,
- perspective-specific decision variables and objectives,
- exogenous treatment of other stakeholders' decisions, and
- modular evaluation of objectives.

The code is **self-contained**, readable, and suitable for extension (e.g., robust analysis, adaptive policies).

```python
import numpy as np
from typing import Dict, Any, List, Tuple
```

---

1. Shared Environment

```python
class LakeEnvironment:
    """
    Shared biophysical environment.
    """

    def __init__(self, params: Dict[str, Any]):
        """
        params must include:
            b       : natural removal rate
            q       : recycling nonlinearity
            mu      : mean of lognormal natural pollution
            sigma   : std dev of lognormal natural pollution
            T       : time horizon
            X0      : initial pollution level
        """
        self.b = params["b"]
        self.q = params["q"]
        self.mu = params["mu"]
        self.sigma = params["sigma"]
        self.T = params["T"]
        self.X0 = params.get("X0", 0.0)

    def recycling(self, X: float) -> float:
        return (X ** self.q) / (1.0 + X ** self.q)

    def draw_natural_pollution(self) -> float:
        return np.random.lognormal(mean=self.mu, sigma=self.sigma)
```

```
    def step(self, X_t: float, a_t: float, r_t: float, epsilon_t: float) -> float:
        """
        Global transition function.
        """
        X_next = (
            X_t
            + a_t
            - r_t
            + self.recycling(X_t)
            - self.b * X_t
            + epsilon_t
        )
        return max(X_next, 0.0)
```

2. Critical Threshold (Shared Utility)

```python
def compute_x_crit(b: float, q: float, x_max: float = 100.0, tol: float = 1e-6) -> float:
    """
    Solve for X_crit where recycling equals natural removal:
        X^q / (1 + X^q) = b X

    Uses a simple bisection method.
    """
    def f(x):
        return (x ** q) / (1 + x ** q) - b * x

    low, high = 0.0, x_max
    while high - low > tol:
        mid = 0.5 * (low + high)
        if f(mid) > 0:
            low = mid
        else:
            high = mid
    return 0.5 * (low + high)
```

3. Local Community Perspective

```python
class LocalCommunity:
    """
    Community module: controls emissions a_t and evaluates economic objectives.
    """

    def __init__(self, params: Dict[str, Any]):
        """
        params must include:
            alpha  : benefit-to-pollution ratio
            delta  : discount factor
        """
        self.alpha = params["alpha"]
        self.delta = params["delta"]

    def economic_benefit(self, a_t: float, t: int) -> float:
        return self.alpha * a_t * (self.delta ** t)

    def evaluate_objectives(
        self,
        emissions: List[float],
        states: List[float],
        x_crit: float
```

```
    ) -> Dict[str, float]:
        """
        Returns community objectives.
        """
        economic_value = sum(
            self.economic_benefit(a_t, t)
            for t, a_t in enumerate(emissions)
        )

        safe = all(X < x_crit for X in states)

        return {
            "community_economic_benefit": economic_value,
            "community_avoids_eutrophication": float(safe)
        }
```

---

4. Environmental Regulator Perspective

```python
class EnvironmentalRegulator:
    """
    Regulator module: controls remediation r_t and evaluates environmental objectives.
    """

    def __init__(self, params: Dict[str, Any]):
        """
        params must include:
            cost_coeff : coefficient for remediation cost
        """
        self.cost_coeff = params["cost_coeff"]

    def remediation_cost(self, r_t: float) -> float:
        return self.cost_coeff * (r_t ** 2)

    def evaluate_objectives(
        self,
        remediations: List[float],
        states: List[float],
        x_crit: float
    ) -> Dict[str, float]:
        """
        Returns regulator objectives.
        """
        avoids_eutrophication = all(X < x_crit for X in states)

        pollution_stock = sum(states)

        total_cost = sum(self.remediation_cost(r_t) for r_t in remediations)

        return {
            "regulator_avoids_eutrophication": float(avoids_eutrophication),
            "regulator_total_pollution": pollution_stock,
            "regulator_total_cost": total_cost
        }
```

---

5. Unified Simulation Interface (Required Output)

```python
def simulate_lake_model(
    decisions: Dict[str, List[float]],
    uncertain_params: Dict[str, Any],
    constant_params: Dict[str, Any]
```

```python
) -> Dict[str, float]:
    """
    Unified interface function.

    Inputs
    ------
    decisions:
        {
            "a": [a_0, ..., a_T],   # community emissions
            "r": [r_0, ..., r_T]    # regulator remediation
        }

    uncertain_params:
        {
            "b", "q", "mu", "sigma", "delta"
        }

    constant_params:
        {
            "T", "X0", "alpha", "cost_coeff"
        }

    Returns
    -------
    Dictionary of all objective function values.
    """

    # --- Initialize environment ---
    env = LakeEnvironment({
        "b": uncertain_params["b"],
        "q": uncertain_params["q"],
        "mu": uncertain_params["mu"],
        "sigma": uncertain_params["sigma"],
        "T": constant_params["T"],
        "X0": constant_params.get("X0", 0.0),
    })

    # --- Initialize stakeholders ---
    community = LocalCommunity({
        "alpha": constant_params["alpha"],
        "delta": uncertain_params["delta"],
    })

    regulator = EnvironmentalRegulator({
        "cost_coeff": constant_params["cost_coeff"]
    })

    # --- Compute critical threshold ---
    x_crit = compute_x_crit(env.b, env.q)

    # --- Simulate dynamics ---
    X = env.X0
    states = [X]

    for t in range(env.T):
        a_t = decisions["a"][t]
        r_t = decisions["r"][t]
        epsilon_t = env.draw_natural_pollution()

        X = env.step(X, a_t, r_t, epsilon_t)
        states.append(X)

        if X >= x_crit:
            break  # irreversible failure

    # --- Evaluate objectives ---
    results = {}
    results.update(
        community.evaluate_objectives(decisions["a"], states, x_crit)
    )
    results.update(
        regulator.evaluate_objectives(decisions["r"], states, x_crit)
    )
```

```
    return results
```

---

6. Key Properties of This Implementation

- **Single shared environment** with one state trajectory
- **Perspective-specific control** (`a_t` vs `r_t`)
- **Exogenous treatment** of other stakeholders' actions
- **Multiple objectives returned explicitly** (no aggregation)
- Ready for:
  - robust decision-making,
  - multi-agent experiments,
  - exploratory modeling under deep uncertainty.

If you want next steps, I can:

- add **adaptive policies** instead of fixed decisions,
- show **robustness metrics** across uncertainty ensembles,
- or extend this to **more stakeholders or institutional rules**.

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 brentq. Please refine your implementation.

Below is a **refined version of the implementation**, updated to ensure that:

- X_{crit} is **explicitly constrained to the interval** $[0.01, 1.5]$,
- the root is computed using **Brent's method (`brentq`)**, which is robust and efficient,
- the change is localized and does **not break modularity** or the unified structure.

Only the **necessary parts** are modified; everything else remains internally consistent.

1. Updated Imports

```python
import numpy as np
from typing import Dict, Any, List
from scipy.optimize import brentq
```

2. Refined Critical Threshold Computation

```python
def compute_x_crit(
    b: float,
    q: float,
    x_min: float = 0.01,
    x_max: float = 1.5
) -> float:
    """
    Compute the critical pollution threshold X_crit using Brent's method.

    X_crit solves:
        X^q / (1 + X^q) = b X

    The root is searched for in the interval [0.01, 1.5].
    """

    def balance_function(x: float) -> float:
        recycling = (x ** q) / (1.0 + x ** q)
        removal = b * x
        return recycling - removal

    # Brent's method requires opposite signs at interval endpoints
    f_min = balance_function(x_min)
    f_max = balance_function(x_max)

    if f_min * f_max > 0:
        raise ValueError(
            "No root found for X_crit in [0.01, 1.5]. "
            "Check parameter values b and q."
        )
```

```
    x_crit = brentq(balance_function, x_min, x_max)
    return x_crit
```

3. Unified Simulation Interface (Updated Usage)

The unified interface remains unchanged **except** that it now uses the refined **`compute_x_crit`** internally.

```python
def simulate_lake_model(
    decisions: Dict[str, List[float]],
    uncertain_params: Dict[str, Any],
    constant_params: Dict[str, Any]
) -> Dict[str, float]:
    """
    Unified interface function.
    """

    # --- Initialize environment ---
    env = LakeEnvironment({
        "b": uncertain_params["b"],
        "q": uncertain_params["q"],
        "mu": uncertain_params["mu"],
        "sigma": uncertain_params["sigma"],
        "T": constant_params["T"],
        "X0": constant_params.get("X0", 0.0),
    })

    # --- Initialize stakeholders ---
    community = LocalCommunity({
        "alpha": constant_params["alpha"],
        "delta": uncertain_params["delta"],
    })

    regulator = EnvironmentalRegulator({
        "cost_coeff": constant_params["cost_coeff"]
    })

    # --- Compute endogenous tipping point ---
    x_crit = compute_x_crit(
        b=env.b,
        q=env.q,
        x_min=0.01,
        x_max=1.5
    )

    # --- Simulate dynamics ---
    X = env.X0
    states = [X]

    for t in range(env.T):
        a_t = decisions["a"][t]
        r_t = decisions["r"][t]
        epsilon_t = env.draw_natural_pollution()

        X = env.step(X, a_t, r_t, epsilon_t)
        states.append(X)

        if X >= x_crit:
            break  # irreversible eutrophication

    # --- Evaluate objectives ---
    results = {}
    results.update(
        community.evaluate_objectives(decisions["a"], states, x_crit)
    )
    results.update(
```

```
        regulator.evaluate_objectives(decisions["r"], states, x_crit)
    )

    return results
```

4. Why This Refinement Matters

This update ensures:

- **Physical realism**: X_{crit} is constrained to a scientifically meaningful range.
- **Numerical robustness**: **`brentq`** guarantees convergence when a root exists.
- **Consistency with deep uncertainty**: Different (b, q) realizations yield different tipping points, but always within a plausible domain.
- **Modularity preserved**: No stakeholder logic depends on how X_{crit} is computed.

If you want, I can next:

- add **explicit failure states** instead of early termination,
- expose X_{crit} as an **output diagnostic**,
- or show how to **stress-test policies** across ensembles of (b, q) values.

Your implementation of the lognormal distribution is incorrect. Please refine your implementation.

You're absolutely right — thank you for catching that.
The previous implementation **incorrectly treated** μ and σ as the *log-space parameters* of the lognormal distribution, whereas in your problem statement:

> the natural pollution ϵ_t follows a log-normal distribution **with mean μ and variance σ^2**

That distinction matters a lot.

Below is a **correct, internally consistent refinement** that maps the **specified mean and variance in real space** to the **underlying normal distribution parameters** required by **`np.random.lognormal`**.

---

1. Correct Lognormal Parameterization

If a random variable $\epsilon \sim \text{LogNormal}(m, s^2)$ in **real space**, then the corresponding normal parameters (μ_ℓ, σ_ℓ) satisfy:

$$\sigma_\ell^2 = \ln\left(1 + \frac{s^2}{m^2}\right)$$
$$\mu_\ell = \ln(m) - \frac{1}{2}\sigma_\ell^2$$

where:

- $m = \mathbb{E}[\epsilon]$,
- $s^2 = \text{Var}(\epsilon)$.

This transformation must be applied **before sampling**.

---

2. Refined Implementation in the Environment Class

🔧 Updated `LakeEnvironment` (only relevant parts shown)

```python
class LakeEnvironment:
    """
    Shared biophysical environment.
    """

    def __init__(self, params):
        """
        params must include:
            b       : natural removal rate
            q       : recycling nonlinearity
            mu      : mean of lognormal natural pollution (real space)
            sigma   : standard deviation of lognormal natural pollution (real space)
```

```
            T       : time horizon
            X0      : initial pollution level
        """
        self.b = params["b"]
        self.q = params["q"]
        self.mu = params["mu"]
        self.sigma = params["sigma"]
        self.T = params["T"]
        self.X0 = params.get("X0", 0.0)

        # --- convert real-space mean/variance to log-space parameters ---
        self._log_sigma_sq = np.log(1.0 + (self.sigma ** 2) / (self.mu ** 2))
        self._log_sigma = np.sqrt(self._log_sigma_sq)
        self._log_mu = np.log(self.mu) - 0.5 * self._log_sigma_sq

    def recycling(self, X):
        return (X ** self.q) / (1.0 + X ** self.q)

    def draw_natural_pollution(self):
        """
        Draw epsilon_t ~ LogNormal(mean=mu, var=sigma^2) in real space.
        """
        return np.random.lognormal(
            mean=self._log_mu,
            sigma=self._log_sigma
        )

    def step(self, X_t, a_t, r_t, epsilon_t):
        X_next = (
            X_t
            + a_t
            - r_t
            + self.recycling(X_t)
            - self.b * X_t
            + epsilon_t
        )
        return max(X_next, 0.0)
```

3. What Changed (and Why It's Correct Now)

Aspect	Before	Now
Interpretation of μ, σ	Log-space parameters ❌	Real-space mean & std ✅
Distribution used	Mis-specified	Mathematically correct
Sampling method	**`lognormal(mu, sigma)`**	**`lognormal(log_mu, log_sigma)`**
Consistency with problem statement	❌	✅

4. Implications for the Model

This correction ensures that:

- **Uncertainty is represented correctly**
  The expected natural pollution and its variability match the stated model.
- **Policy evaluation is meaningful**
  Over- or underestimating stochastic inflows can drastically distort eutrophication risk.

- **Robustness analysis is trustworthy**
  Especially important under deep uncertainty where distributions are stress-tested.

If you want, I can next:

- expose **realized** ϵ_t as an output diagnostic,
- add **variance checks** to validate sampling correctness,
- or show how this affects **failure probabilities** under identical policies.