Lake_1

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 clean **formalization of the Lake Problem as a simulation / decision model**, separating deterministic dynamics, stochastic elements, and deep uncertainty. I present it in a structure commonly used in **dynamic decision-making, control, and robust policy analysis**.

---

1. Model Structure Overview

The lake problem can be formalized as a **discrete-time stochastic dynamic system** with:

- A single state variable (lake pollution)
- A control (decision) variable (anthropogenic emissions)
- Nonlinear system dynamics
- An economic objective with discounting

- A hard ecological constraint (irreversible eutrophication)
- Both **aleatory uncertainty** (stochastic pollution shocks) and **deep uncertainty** (unknown parameter distributions)

---

2. Time Index

- Time horizon:

$$t = 0, 1, 2, \ldots, T$$

---

3. State Variable

- **Lake pollution stock**

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

- Initial condition:

$$X_0 = 0$$

---

4. Decision Variable (Control)

- **Anthropogenic pollution emissions**

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

- Chosen by decision-makers at each time step t.

---

5. Transition (State Dynamics) Function

The evolution of lake pollution is governed by:

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

where:

$$g(X_t, a_t, \epsilon_t) = X_t + a_t + \frac{X_t^q}{1 + X_t^q} - bX_t + \epsilon_t$$

Interpretation of Terms

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

6. Ecological Threshold (Constraint)

- **Critical pollution level**:

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

- **Irreversibility constraint**:

$$X_t \geq X_{\text{crit}} \;\Rightarrow\; \text{lake eutrophication (absorbing failure state)}$$

Once exceeded, the system transitions into a permanently degraded state with very large or infinite loss.

7. Objective Function

The goal is to **maximize discounted economic benefits** from pollution emissions:

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

where:

- $\alpha > 0$: benefit-to-pollution ratio
- $\delta \in (0, 1)$: discount factor
- Expectation is taken over the stochastic process $\{\epsilon_t\}$

Optionally, the objective may be written as a **constrained optimization**:

$$\max \; \mathbb{E}[f_{economic}] \quad \text{s.t.} \quad \Pr(X_t < X_{\text{crit}} \; \forall t) \geq \theta$$

for some safety threshold θ.

8. Stochastic Variables (Aleatory Uncertainty)

Natural Pollution

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

- Independent and identically distributed across time
- Mean: μ
- Variance: σ^2

This represents **environmental variability** that cannot be controlled.

9. Constant Parameters (Known or Assumed Fixed)

Parameter	Description
α	Economic benefit per unit pollution
T	Planning horizon
X_{crit}	Pollution threshold for eutrophication

10. Deep Uncertainty (Epistemic Uncertainty)

The following parameters are **deeply uncertain**, meaning their probability distributions are unknown or contested:

- μ: mean of natural pollution
- σ: variance of natural pollution
- b: natural removal rate
- q: recycling nonlinearity parameter
- δ: discount factor

Formally:

$$(\mu, \sigma, b, q, \delta) \in \Theta$$

where Θ is an **uncertain but bounded parameter set**, not a known probability distribution.

11. Simulation Model Summary

The simulation model consists of:

1. **State update equation**

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

2. **Decision rule (policy)**

$$a_t = \pi_t(X_t)$$

3. **Random sampling of ϵ_t**
4. **Failure condition**

$$X_t \geq X_{\text{crit}} \Rightarrow \text{terminal failure}$$

5. **Performance metrics**
   - Expected economic benefit
   - Probability of eutrophication
   - Time-to-failure (if any)

12. Interpretation

This is a **stochastic, nonlinear, dynamic decision problem with irreversible thresholds and deep uncertainty**, making it well-suited for:

- Robust Decision Making (RDM)
- Stochastic dynamic programming (with approximations)
- Adaptive policies
- Scenario discovery and stress testing

If you want, I can next:

- Express this as a **Markov Decision Process**
- Provide **pseudo-code for simulation**
- Reformulate it for **robust or adaptive policy optimization**

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

Good point — the threshold is **endogenous** and inherits deep uncertainty from the biophysical parameters. Below is a **refined and internally consistent model specification** that treats X_{crit} correctly.

---

1. Revised Role of the Critical Threshold

Definition (Endogenous Threshold)

The critical pollution level is **not a fixed constant**, but a function of uncertain ecological parameters:

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

where X_{crit} is defined implicitly as the **non-trivial equilibrium** at which:

$$\text{natural recycling} = \text{natural removal}$$

That is,

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

For given (b, q), equation (1) may have:

- a trivial equilibrium $X = 0$
- one or more positive equilibria
  The smallest **unstable positive equilibrium** is taken as X_{crit}.

Thus, eutrophication risk depends directly on the uncertain parameters b and q.

---

2. Updated Model Components

2.1 Time Index

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

2.2 State Variable

- Lake pollution stock:

$$X_t \geq 0, \quad X_0 = 0$$

2.3 Decision Variable

- Anthropogenic pollution emissions:

$$a_t \geq 0$$

Chosen at each time step based on available information.

2.4 Transition Function (Unchanged)

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

This defines a **nonlinear stochastic dynamical system** parameterized by (b, q).

---

3. Stochastic Variables (Aleatory Uncertainty)

Natural Pollution Inflow

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

---

4. Deeply Uncertain Parameters

The following parameters are **deeply uncertain**, with unknown or disputed probability distributions:

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

They belong to an uncertainty set:

$$\theta \in \Theta$$

No probabilistic weighting over Θ is assumed.

---

5. Endogenous Failure Condition

Parameter-Dependent Failure Set

For a given realization of (b, q), define:

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

where $X_{\text{crit}}(b, q)$ satisfies equation (1).

Irreversibility

Once the system enters $\mathcal{F}(b, q)$, the lake transitions to an **absorbing eutrophic state**, with catastrophic loss.

---

6. Objective Function (Refined)

The planner seeks to maximize expected discounted benefits **while avoiding failure across deep uncertainties**:

$$\max_{\pi} \ \mathbb{E}_{\epsilon} \left[\sum_{t=0}^{T} \alpha a_t \delta^t \ \middle| \ \theta \right]$$

subject to:

$$a_t = \pi_t(X_t)$$

and evaluated across:

$$\theta \in \Theta$$

---

7. Robust / Safety-Oriented Performance Metrics

Because X_{crit} is uncertain, performance is assessed using **robust criteria**, for example:

(a) Worst-Case Constraint

$$\sup_{\theta \in \Theta} \Pr_{\epsilon} \left(\exists t : X_t \geq X_{\text{crit}}(b, q) \right) \leq \eta$$

(b) Regret-Based Objective

$$\min_{\pi} \sup_{\theta \in \Theta} [V^*(\theta) - V^{\pi}(\theta)]$$

(c) Satisficing / Safe Operating Space

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

---

8. Simulation Model (Final Form)

A single simulation run proceeds as:

1. Sample a parameter vector

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

2. Compute $X_{\text{crit}}(b, q)$ by solving (1)
3. Initialize $X_0 = 0$
4. For $t = 0, \ldots, T$:
    - Choose $a_t = \pi_t(X_t)$
    - Sample $\epsilon_t \sim \text{LogNormal}(\mu, \sigma^2)$

- Update X_{t+1}
- Check failure: $X_{t+1} \geq X_{\text{crit}}(b, q)$

5. Record:
    - Economic benefits
    - Failure occurrence and timing

---

9. Key Conceptual Refinement

Crucially:

- The lake does **not** have one threshold
- Each plausible ecological reality (b, q) induces its **own tipping point**
- Policies must therefore be **robust to a family of tipping points**, not a single known boundary

This makes the lake problem a canonical example of a **deeply uncertain tipping-point control problem**.

If you want, I can next:

- Express this as a **parameterized MDP**
- Show how this is implemented in **exploratory modeling / RDM**
- Derive the **bifurcation structure** of the recycling–removal balance

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 **two-stakeholder refinement** of the lake problem as a **shared-environment, multi-actor simulation model**. The local community and the environmental regulator interact through the same lake dynamics but **control different decision variables and pursue different objectives**, without collapsing objectives into weighted sums.

I. Common Environment (Shared by All Stakeholders)

This layer defines the **physical lake system, uncertainties, and shared state variables**. Neither stakeholder controls this layer directly.

I.1 Time

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

I.2 Global State Variables

- **Lake pollution stock**

$$X_t \in \mathbb{R}_{\geq 0}, \quad X_0 = 0$$

- **Eutrophication indicator**

$$E_t = \begin{cases} 0 & X_t < X_{\text{crit}}(b, q) \\ 1 & X_t \geq X_{\text{crit}}(b, q) \end{cases}$$

Once $E_t = 1$, the system enters an **absorbing degraded state**.

I.3 Global Transition Function (Environment Dynamics)

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

where:

- A_t = **total anthropogenic pollution** (aggregated from agents)
- ϵ_t = stochastic natural pollution

I.4 Stochastic Process (Aleatory Uncertainty)

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

I.5 Deeply Uncertain Parameters (Global)

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

No probability distribution over Θ is assumed.

I.6 Endogenous Critical Threshold (Global)

$$X_{\text{crit}}(b, q) \;\; \text{solves} \;\; \frac{X^q}{1 + X^q} = bX$$

This threshold is **shared** and **parameter-dependent**, but unknown ex ante.

II. Stakeholder 1: Local Community Perspective

The local community represents **industry and agriculture** seeking economic benefits from pollution-generating activities.

II.1 Community Decision Variables

- **Anthropogenic emissions**

$$a_t^C \geq 0$$

II.2 Information Available to the Community

- Observes current pollution level X_t
- Does **not know** the true values of (b, q, μ, σ)
- May or may not anticipate regulatory actions

II.3 Community Contribution to Environment

$$A_t = a_t^C \quad \text{(before regulation)}$$

or, if regulation applies,

$$A_t = a_t^C - r_t$$

where r_t is imposed externally by the regulator.

II.4 Community-Controlled State Transition (Induced)

The community does **not** control the full transition, but its action affects it via a_t^C:

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

II.5 Community Objectives (Multiple, Non-Aggregated)

Objective C1: Maximize Economic Benefits

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

Objective C2: Avoid Economic Collapse from Eutrophication

$$\min_{\pi^C} \Pr\left(\exists t : X_t \geq X_{\text{crit}}(b, q)\right)$$

Objective C3 (Optional): Stability of Production

$$\min_{\pi^C} \operatorname{Var}(a_t^C)$$

This reflects political and economic resistance to volatile production levels.

II.6 Community Policy Form

$$a_t^C = \pi_t^C(X_t)$$

Possibly adaptive, reactive, or myopic.

III. Stakeholder 2: Environmental Regulator Perspective

The regulator represents a public authority tasked with **protecting the lake**, potentially at the cost of reduced economic output.

III.1 Regulator Decision Variables

The regulator does **not** emit pollution. Instead, it controls constraints on emissions.

Examples:

Option R1: Emission Cap

$$a_t^C \leq \bar{a}_t$$

Option R2: Pollution Abatement / Removal

$$r_t \geq 0 \quad \Rightarrow \quad A_t = a_t^C - r_t$$

Option R3: Pollution Tax (implicit control)

$$\tau_t \geq 0 \quad \text{(affects community behavior)}$$

III.2 Regulator-Controlled State Transition (Induced)

With abatement:

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

III.3 Regulator Objectives (Multiple, Non-Weighted)

Objective R1: Prevent Eutrophication

$$\min_{\pi^R} \Pr\left(\exists t : X_t \geq X_{\text{crit}}(b, q)\right)$$

Objective R2: Minimize Expected Pollution Levels

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

Objective R3: Minimize Regulatory Intervention Cost

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

where $c(\cdot)$ is an increasing convex cost function.

III.4 Regulator Policy Form

$$r_t = \pi_t^R(X_t) \quad \text{or} \quad \bar{a}_t = \pi_t^R(X_t)$$

Typically conservative and precautionary under deep uncertainty.

IV. Coupled Multi-Actor System Summary

Component	Description
Shared state	X_t, E_t
Shared dynamics	Nonlinear stochastic lake model
Community control	Pollution emissions a_t^C
Regulator control	Caps, abatement, or enforcement
Interaction	Regulator constrains community action
Uncertainty	Both stochastic (ϵ_t) and deep (θ)
Failure	Parameter-dependent tipping point

V. Conceptual Interpretation

- This is a **multi-objective, multi-agent dynamic system**
- Stakeholders interact **only through the shared environment**
- No stakeholder controls the full system
- Tradeoffs emerge endogenously rather than via weighted objectives
- The problem naturally supports:
  - Game-theoretic analysis
  - Adaptive governance
  - Robust and precautionary policy design

If you want, I can next:

- Cast this as a **dynamic game**
- Show **Pareto fronts** for regulator vs community
- Specify **adaptive trigger-based policies**
- Translate this into an **agent-based simulation architecture**

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 composed model** that embeds both stakeholder perspectives into **one internally consistent simulation architecture**, while preserving **modularity, separability of objectives, and exogeneity of other actors' decisions** for evaluation.

The model is written in a way that can be implemented directly as a **multi-actor simulation / dynamic game / exploratory model**.

Unified Composed Lake Model

1. Global Model Structure

The system is a **shared stochastic dynamical environment** with:

- One common state space
- One common transition function
- Multiple decision-makers
- Perspective-specific objectives
- Deep uncertainty affecting all components

Each stakeholder observes the same environment but **optimizes its own objectives**, treating the other's actions as **exogenous inputs**.

---

2. Common Environment (Shared by All Perspectives)

2.1 Time

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

2.2 Global State Variables

- **Lake pollution stock**

$$X_t \in \mathbb{R}_{\geq 0}, \quad X_0 = 0$$

- **Eutrophication indicator (absorbing)**

$$E_t = \begin{cases} 0 & X_t < X_{\text{crit}}(b,q) \\ 1 & X_t \geq X_{\text{crit}}(b,q) \end{cases}$$

Once $E_t = 1$, the system remains in the eutrophic state.

2.3 Global Stochastic Variable

- **Natural pollution inflow**

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

2.4 Deeply Uncertain Global Parameters

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

No probability distribution over Θ is assumed.

2.5 Endogenous, Parameter-Dependent Threshold

The critical pollution level is **not a constant**, but a function of uncertain ecological parameters:

$$X_{\text{crit}}(b,q) \quad \text{satisfying} \quad \frac{X^q}{1+X^q} = bX$$

This function is computed separately for each realization of (b, q).

2.6 Global Transition Function

The shared lake dynamics are:

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

where:

$$A_t = a_t^C - r_t^R$$

is the **net anthropogenic pollution**, aggregated from stakeholder decisions.

---

3. Stakeholder-Specific Embedded Models

Each stakeholder model is defined as a **projection** of the global model, consisting of:

- Its own decision variables
- Its own policy
- Its own objectives
- The shared environment treated as given

4. Stakeholder 1: Local Community Module

4.1 Community Decision Variables

- **Emissions decision**

$$a_t^C \geq 0$$

4.2 Community Policy

$$a_t^C = \pi_t^C(X_t)$$

The community observes the shared state X_t and chooses emissions.

4.3 Community-Induced State Evolution

From the community's perspective, the state evolves as:

$$X_{t+1} = X_t + a_t^C - \underbrace{r_t^R}_{\text{exogenous}} + \frac{X_t^q}{1 + X_t^q} - bX_t + \epsilon_t$$

where the regulator's action r_t^R is treated as **exogenous**.

4.4 Community Objectives (Non-Aggregated)

The community evaluates outcomes using the following objectives:

C1. Economic benefit

$$\max \; J_C^{(1)} = \mathbb{E}\left[\sum_{t=0}^{T} \alpha a_t^C \delta^t\right]$$

C2. Avoidance of eutrophication

$$\min \; J_C^{(2)} = \Pr\left(\exists t : X_t \geq X_{\text{crit}}(b, q)\right)$$

(Optional additional objectives such as stability can be added without changing the environment.)

5. Stakeholder 2: Environmental Regulator Module

5.1 Regulator Decision Variables

- **Abatement / removal effort**

$$r_t^R \geq 0$$

(Other instruments such as caps or taxes can be represented equivalently.)

5.2 Regulator Policy

$$r_t^R = \pi_t^R(X_t)$$

5.3 Regulator-Induced State Evolution

From the regulator's perspective:

$$X_{t+1} = X_t + \underbrace{a_t^C}_{\text{exogenous}} - r_t^R + \frac{X_t^q}{1 + X_t^q} - bX_t + \epsilon_t$$

The community's emissions a_t^C are treated as **exogenous inputs**.

5.4 Regulator Objectives (Non-Aggregated)

R1. Prevent eutrophication

$$\min\ J_R^{(1)} = \Pr\left(\exists t : X_t \geq X_{\text{crit}}(b, q)\right)$$

R2. Minimize pollution levels

$$\min\ J_R^{(2)} = \mathbb{E}\left[\sum_{t=0}^{T} X_t\right]$$

R3. Minimize regulatory cost

$$\min\ J_R^{(3)} = \mathbb{E}\left[\sum_{t=0}^{T} c(r_t^R)\right]$$

6. Composition Logic (Key Requirement Satisfaction)

Requirement 1: Embed each perspective's decisions and objectives

✔ Each stakeholder has its own decision variables, policies, and objective vector.

Requirement 2: Enable modular evaluation

✔ Each module evaluates performance using its own objective functions while sharing the same simulated trajectory $\{X_t\}$.

Requirement 3: Treat other perspectives' decisions as exogenous

✔ In each stakeholder's induced transition function, the other actor's decisions appear explicitly as exogenous inputs.

Requirement 4: Incorporate all components

✔️ The composed model includes:

- Nonlinear lake dynamics
- Stochastic inflows
- Deep uncertainty
- Endogenous tipping points
- Irreversible failure
- Multi-objective evaluation

---

7. Unified Simulation Algorithm

For a given realization of $\theta \in \Theta$:

1. Compute $X_{\text{crit}}(b, q)$
2. Initialize $X_0 = 0$
3. For $t = 0, \ldots, T$:
    - Community chooses $a_t^C = \pi_t^C(X_t)$
    - Regulator chooses $r_t^R = \pi_t^R(X_t)$
    - Sample ϵ_t
    - Update X_{t+1}
    - Check eutrophication condition
4. Evaluate $\{J_C^{(i)}\}$ and $\{J_R^{(j)}\}$ separately

---

8. Conceptual Interpretation

This unified formulation is:

- A **multi-actor stochastic dynamic system**
- Equivalent to a **dynamic game with deep uncertainty**
- Suitable for:
    - Robust decision making (RDM)
    - Adaptive governance analysis
    - Pareto tradeoff exploration
    - Policy stress testing across tipping points

If you want, I can next:

- Formalize this as a **dynamic game with equilibrium concepts**
- Show how to implement this in **exploratory modeling frameworks**

- Define **adaptive trigger-based policies** for each stakeholder

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 **clean, modular Python implementation** of the unified model.
It uses **classes to represent the shared environment and each stakeholder perspective**, and ends with a **single interface function** that matches your specification.

The code is written for **clarity and extensibility**, not execution efficiency.

Design Overview (Mapping to the Unified Model)

- **`LakeEnvironment`**
  Shared environment: states, dynamics, stochastic inflow, eutrophication logic
- **`LocalCommunity`**
  Perspective-specific decisions and objectives
- **`EnvironmentalRegulator`**
  Perspective-specific decisions and objectives
- **`simulate_lake_model(...)`**
  Interface function:
  - Inputs:
    - decision variables (time series)
    - uncertain parameters
    - constant parameters
  - Outputs:
    - values of **each objective function**, separated by perspective

---

Python Implementation

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

---

1. Shared Environment

```python
class LakeEnvironment:
    """
    Shared lake environment with nonlinear stochastic dynamics
    and endogenous eutrophication threshold.
    """

    def __init__(self, uncertain_params: Dict[str, float], constants: Dict[str, float]):
        self.mu = uncertain_params["mu"]
        self.sigma = uncertain_params["sigma"]
        self.b = uncertain_params["b"]
        self.q = uncertain_params["q"]

        self.T = constants["T"]

        self.X_crit = self._compute_critical_threshold()

    def _compute_critical_threshold(self) -> float:
        """
        Solve for X_crit from:
            X^q / (1 + X^q) = b X
        Uses a simple numerical search over a bounded domain.
        """

        X_grid = np.linspace(1e-6, 10.0, 10_000)
        lhs = X_grid**self.q / (1 + X_grid**self.q)
        rhs = self.b * X_grid

        diff = np.abs(lhs - rhs)
        return X_grid[np.argmin(diff)]

    def sample_natural_pollution(self) -> float:
        """
        Sample stochastic natural pollution inflow.
        """
        return np.random.lognormal(mean=self.mu, sigma=self.sigma)

    def step(self, X_t: float, a_c: float, r_r: float, epsilon_t: float) -> float:
        """
        One-step lake pollution dynamics.
        """

        net_anthropogenic = a_c - r_r
        recycling = X_t**self.q / (1 + X_t**self.q)
        removal = self.b * X_t

        X_next = X_t + net_anthropogenic + recycling - removal + epsilon_t
        return max(X_next, 0.0)
```

2. Local Community Perspective

```python
class LocalCommunity:
    """
    Local community controlling pollution emissions.
    """

    def __init__(self, constants: Dict[str, float]):
        self.alpha = constants["alpha"]
        self.delta = constants["delta"]

        self.economic_benefit = 0.0
        self.failed = False

    def reset(self):
        self.economic_benefit = 0.0
        self.failed = False
```

```
    def observe_and_act(self, t: int, X_t: float, decisions: Dict[str, List[float]]) -> float:
        """
        Return community emissions at time t.
        """
        return decisions["a_community"][t]

    def update_objectives(self, t: int, a_c: float):
        """
        Update economic objective.
        """
        self.economic_benefit += self.alpha * a_c * (self.delta ** t)

    def check_failure(self, X_t: float, X_crit: float):
        if X_t >= X_crit:
            self.failed = True
```

---

3. Environmental Regulator Perspective

```python
class EnvironmentalRegulator:
    """
    Environmental regulator controlling abatement.
    """

    def __init__(self):
        self.total_pollution = 0.0
        self.regulation_cost = 0.0
        self.failed = False

    def reset(self):
        self.total_pollution = 0.0
        self.regulation_cost = 0.0
        self.failed = False

    def observe_and_act(self, t: int, X_t: float, decisions: Dict[str, List[float]]) -> float:
        """
        Return regulator abatement at time t.
        """
        return decisions["r_regulator"][t]

    def update_objectives(self, X_t: float, r_r: float):
        self.total_pollution += X_t
        self.regulation_cost += r_r**2  # convex cost example

    def check_failure(self, X_t: float, X_crit: float):
        if X_t >= X_crit:
            self.failed = True
```

---

4. Unified Simulation Interface (Required Function)

```python
def simulate_lake_model(
    decisions: Dict[str, List[float]],
    uncertain_params: Dict[str, float],
    constants: Dict[str, float],
) -> Dict[str, Any]:
    """
    Unified interface for the composed lake model.

    Inputs
    ------
    decisions:
        {
```

```
            "a_community": [a_0, a_1, ..., a_T],
            "r_regulator": [r_0, r_1, ..., r_T]
        }

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

    constants:
        {
            "T": int,
            "alpha": float,
            "delta": float
        }

    Returns
    -------
    Dictionary containing values of each objective function.
    """

    # Initialize environment and agents
    env = LakeEnvironment(uncertain_params, constants)
    community = LocalCommunity(constants)
    regulator = EnvironmentalRegulator()

    community.reset()
    regulator.reset()

    # Initial state
    X_t = 0.0
    T = constants["T"]

    for t in range(T):
        # Agents act (treating each other as exogenous)
        a_c = community.observe_and_act(t, X_t, decisions)
        r_r = regulator.observe_and_act(t, X_t, decisions)

        # Sample stochastic inflow
        epsilon_t = env.sample_natural_pollution()

        # Environment transition
        X_t = env.step(X_t, a_c, r_r, epsilon_t)

        # Update objectives
        community.update_objectives(t, a_c)
        regulator.update_objectives(X_t, r_r)

        # Check eutrophication
        community.check_failure(X_t, env.X_crit)
        regulator.check_failure(X_t, env.X_crit)

        if X_t >= env.X_crit:
            break  # absorbing failure state

    return {
        "community": {
            "economic_benefit": community.economic_benefit,
            "eutrophication_occurred": community.failed,
        },
        "regulator": {
            "total_pollution": regulator.total_pollution,
            "regulation_cost": regulator.regulation_cost,
            "eutrophication_occurred": regulator.failed,
        },
        "environment": {
            "X_crit": env.X_crit,
            "final_pollution": X_t,
        },
    }
```

Key Properties of This Implementation

✔️ **Single unified environment**
✔️ **Perspective-specific decision variables and objectives**
✔️ **Other perspectives treated as exogenous inputs**
✔️ **Modular and extensible (easy to add new stakeholders)**
✔️ **Compatible with robust decision-making and exploratory modeling**

If you want, I can next:

- Add **adaptive policies instead of fixed decision sequences**
- Extend this to **multiple communities**
- Show how to use this in **robust policy search or RDM experiments**
- Convert this into a **Gym-style environment**

According to the specification, "delta" is an uncertain parameter instead of a constant. Please correct your implementation.

You're right — thank you for catching that. Below is a **corrected implementation** in which **`delta` is treated as a deeply uncertain parameter**, not a constant.

The fix involves:

1. Moving **`delta`** from **`constants`** to **`uncertain_params`**
2. Passing **`delta`** into the **`LocalCommunity`** from **`uncertain_params`**
3. Updating the interface documentation accordingly

No other model logic changes.

Corrected Modular Python Implementation

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

---

1. Shared Environment (unchanged)

```python
class LakeEnvironment:
    """
    Shared lake environment with nonlinear stochastic dynamics
    and endogenous eutrophication threshold.
    """

    def __init__(self, uncertain_params: Dict[str, float], constants: Dict[str, float]):
        self.mu = uncertain_params["mu"]
        self.sigma = uncertain_params["sigma"]
        self.b = uncertain_params["b"]
        self.q = uncertain_params["q"]

        self.T = constants["T"]

        self.X_crit = self._compute_critical_threshold()

    def _compute_critical_threshold(self) -> float:
        """
        Solve X^q / (1 + X^q) = b X using a simple grid search.
        """
        X_grid = np.linspace(1e-6, 10.0, 10_000)
        lhs = X_grid**self.q / (1 + X_grid**self.q)
        rhs = self.b * X_grid
        diff = np.abs(lhs - rhs)
        return X_grid[np.argmin(diff)]

    def sample_natural_pollution(self) -> float:
```

```
        return np.random.lognormal(mean=self.mu, sigma=self.sigma)

    def step(self, X_t: float, a_c: float, r_r: float, epsilon_t: float) -> float:
        net_anthropogenic = a_c - r_r
        recycling = X_t**self.q / (1 + X_t**self.q)
        removal = self.b * X_t
        X_next = X_t + net_anthropogenic + recycling - removal + epsilon_t
        return max(X_next, 0.0)
```

2. Local Community (corrected: `delta` is uncertain)

```python
class LocalCommunity:
    """
    Local community controlling pollution emissions.
    """

    def __init__(self, uncertain_params: Dict[str, float], constants: Dict[str, float]):
        self.alpha = constants["alpha"]
        self.delta = uncertain_params["delta"]

        self.economic_benefit = 0.0
        self.failed = False

    def reset(self):
        self.economic_benefit = 0.0
        self.failed = False

    def observe_and_act(self, t: int, X_t: float, decisions: Dict[str, List[float]]) -> float:
        return decisions["a_community"][t]

    def update_objectives(self, t: int, a_c: float):
        self.economic_benefit += self.alpha * a_c * (self.delta ** t)

    def check_failure(self, X_t: float, X_crit: float):
        if X_t >= X_crit:
            self.failed = True
```

3. Environmental Regulator (unchanged)

```python
class EnvironmentalRegulator:
    """
    Environmental regulator controlling abatement.
    """

    def __init__(self):
        self.total_pollution = 0.0
        self.regulation_cost = 0.0
        self.failed = False

    def reset(self):
        self.total_pollution = 0.0
        self.regulation_cost = 0.0
        self.failed = False

    def observe_and_act(self, t: int, X_t: float, decisions: Dict[str, List[float]]) -> float:
        return decisions["r_regulator"][t]

    def update_objectives(self, X_t: float, r_r: float):
        self.total_pollution += X_t
        self.regulation_cost += r_r**2  # convex cost
```

```
    def check_failure(self, X_t: float, X_crit: float):
        if X_t >= X_crit:
            self.failed = True
```

---

4. Corrected Unified Interface Function

```python
def simulate_lake_model(
    decisions: Dict[str, List[float]],
    uncertain_params: Dict[str, float],
    constants: Dict[str, float],
) -> Dict[str, Any]:
    """
    Unified interface for the composed lake model.

    Parameters
    ----------
    decisions:
        {
            "a_community": [a_0, ..., a_T],
            "r_regulator": [r_0, ..., r_T]
        }

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

    constants:
        {
            "T": int,
            "alpha": float
        }
    """

    env = LakeEnvironment(uncertain_params, constants)
    community = LocalCommunity(uncertain_params, constants)
    regulator = EnvironmentalRegulator()

    community.reset()
    regulator.reset()

    X_t = 0.0
    T = constants["T"]

    for t in range(T):
        a_c = community.observe_and_act(t, X_t, decisions)
        r_r = regulator.observe_and_act(t, X_t, decisions)

        epsilon_t = env.sample_natural_pollution()
        X_t = env.step(X_t, a_c, r_r, epsilon_t)

        community.update_objectives(t, a_c)
        regulator.update_objectives(X_t, r_r)

        community.check_failure(X_t, env.X_crit)
        regulator.check_failure(X_t, env.X_crit)

        if X_t >= env.X_crit:
            break  # absorbing eutrophic state

    return {
        "community": {
            "economic_benefit": community.economic_benefit,
```

```
            "eutrophication_occurred": community.failed,
        },
        "regulator": {
            "total_pollution": regulator.total_pollution,
            "regulation_cost": regulator.regulation_cost,
            "eutrophication_occurred": regulator.failed,
        },
        "environment": {
            "X_crit": env.X_crit,
            "final_pollution": X_t,
        },
    }
```

Summary of the Correction

- ✅ **`delta`** is now **fully part of deep uncertainty**
- ✅ Each simulation run evaluates objectives under a **specific realization of `delta`**
- ✅ The implementation is now **internally consistent** with the formal specification and stakeholder models

If you want, I can next:

- Add **heterogeneous discounting across stakeholders**
- Make **`delta`** **time-varying or ambiguous**
- Integrate this with **robust or exploratory policy search**

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 and internally consistent correction** to the implementation that:

- Constrains X_{crit} to the range **[0.01, 1.5]**
- Computes X_{crit} using a **proper root-finding method (`brentq`)**
- Leaves all other components of the unified model unchanged

Only the **environment component** needs modification.

Key Refinement (Conceptual)

We now compute X_{crit} as the root of:

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

on the bounded interval:

$$X \in [0.01, 1.5]$$

This is:

- **Numerically well-posed**
- **Consistent with the ecological interpretation**
- **Robust to deep uncertainty in b, q**

If no root exists in this interval, we explicitly flag that case.

Refined Implementation (Only the Changed Parts)

Required Import

```python
from scipy.optimize import brentq
```

Corrected `LakeEnvironment` Class

```python
class LakeEnvironment:
    """
    Shared lake environment with nonlinear stochastic dynamics
    and endogenous eutrophication threshold.
    """

    def __init__(self, uncertain_params: Dict[str, float], constants: Dict[str, float]):
        self.mu = uncertain_params["mu"]
        self.sigma = uncertain_params["sigma"]
        self.b = uncertain_params["b"]
        self.q = uncertain_params["q"]

        self.T = constants["T"]

        self.X_crit = self._compute_critical_threshold()

    def _threshold_equation(self, X: float) -> float:
        """
        Equation whose root defines X_crit:
            X^q / (1 + X^q) - b X = 0
        """
        return (X**self.q) / (1 + X**self.q) - self.b * X

    def _compute_critical_threshold(self) -> float:
        """
        Compute X_crit using Brent's method over [0.01, 1.5].
        """

        lower, upper = 0.01, 1.5

        f_lower = self._threshold_equation(lower)
        f_upper = self._threshold_equation(upper)

        if f_lower * f_upper > 0:
            raise ValueError(
                "No root for X_crit found in [0.01, 1.5] "
                f"for b={self.b}, q={self.q}"
            )

        return brentq(self._threshold_equation, lower, upper)

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

    def step(self, X_t: float, a_c: float, r_r: float, epsilon_t: float) -> float:
        net_anthropogenic = a_c - r_r
        recycling = X_t**self.q / (1 + X_t**self.q)
        removal = self.b * X_t
        X_next = X_t + net_anthropogenic + recycling - removal + epsilon_t
        return max(X_next, 0.0)
```

Why This Refinement Matters

- ✔️ **Ecological realism**: avoids spurious equilibria at extreme pollution levels
- ✔️ **Numerical robustness**: Brent's method guarantees convergence if a root exists
- ✔️ **Deep uncertainty compatibility**: different (b, q) realizations yield different tipping points
- ✔️ **Explicit failure handling**: infeasible parameter combinations are detected, not hidden

Final Consistency Check

Requirement	Status
X_{crit} endogenous	✅
Bounded to [0.01, 1.5]	✅
Root-finding via **`brentq`**	✅
Shared across stakeholders	✅
Compatible with modular simulation	✅

If you want next, I can:

- Add **robust handling** when no root exists (e.g. precautionary defaults)
- Visualize how X_{crit} varies over (b, q)
- Extend the model to **multiple possible tipping points**
- Show how this integrates into **robust decision making (RDM)**

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

You're absolutely right — thank you for pointing this out.
The issue is **conceptual, not just syntactic**:

> In the problem specification,
> μ **and** σ^2 **are the mean and variance of the log-normal random variable** ϵ_t,
> **not** the mean and variance of the underlying normal distribution.

However, `np.random.lognormal(mean=…, sigma=…)` expects the **parameters of the underlying normal distribution**, usually denoted (μ_N, σ_N).

Below is a **correct and internally consistent refinement**.

---

1. Correct Lognormal Parameterization

Let:

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

where:

- $\mu = \mathbb{E}[\epsilon_t]$
- $\sigma^2 = \mathrm{Var}(\epsilon_t)$

The corresponding parameters of the underlying normal distribution are:

$$\sigma_N^2 = \ln\left(1 + \frac{\sigma^2}{\mu^2}\right)$$

$$\mu_N = \ln(\mu) - \frac{1}{2}\sigma_N^2$$

These are the values that must be passed to `np.random.lognormal`.

---

2. Refined Implementation

Corrected `LakeEnvironment` (only changed parts shown)

```python
class LakeEnvironment:
    """
    Shared lake environment with nonlinear stochastic dynamics
    and endogenous eutrophication threshold.
    """
```

```python
    def __init__(self, uncertain_params: Dict[str, float], constants: Dict[str, float]):
        self.mu_eps = uncertain_params["mu"]          # mean of epsilon_t
        self.var_eps = uncertain_params["sigma"]**2 # variance of epsilon_t
        self.b = uncertain_params["b"]
        self.q = uncertain_params["q"]

        self.T = constants["T"]

        # Convert to underlying normal parameters
        self._compute_lognormal_parameters()

        self.X_crit = self._compute_critical_threshold()

    def _compute_lognormal_parameters(self):
        """
        Convert (mean, variance) of lognormal distribution
        to parameters of underlying normal distribution.
        """
        self.sigma_N_sq = np.log(1.0 + self.var_eps / self.mu_eps**2)
        self.sigma_N = np.sqrt(self.sigma_N_sq)
        self.mu_N = np.log(self.mu_eps) - 0.5 * self.sigma_N_sq

    def sample_natural_pollution(self) -> float:
        """
        Sample epsilon_t from LogNormal(mu, sigma^2),
        where mu and sigma^2 are the mean and variance
        of the lognormal distribution.
        """
        return np.random.lognormal(mean=self.mu_N, sigma=self.sigma_N)
```

---

3. What Was Wrong Before (Explicitly)

❌ Previous (Incorrect)

```
python

np.random.lognormal(mean=self.mu, sigma=self.sigma)
```

This incorrectly treated:

- **`mu`** as the **mean of the underlying normal**
- **`sigma`** as the **standard deviation of the underlying normal**

That contradicts the model specification.

---

4. Why This Correction Matters

Aspect	Impact
Statistical correctness	✔ Ensures correct moments of ϵ_t
Policy evaluation	✔ Prevents biased risk estimates
Tipping-point analysis	✔ Correct tail behavior near X_{crit}
Deep uncertainty	✔ Properly reflects ambiguity in natural inflows

An incorrect parameterization **systematically under- or over-estimates extreme pollution events**, which is especially dangerous in a **tipping-point problem**.

5. Final Consistency Check

Component	Status
Lognormal mean = μ	✅
Lognormal variance = σ^2	✅
Correct conversion to normal	✅
Shared across all perspectives	✅
Compatible with deep uncertainty	✅

If you want next, I can:

- Add **input validation** to ensure $\mu > 0$
- Make ϵ_t **temporally correlated**
- Show analytically how tail risk affects eutrophication probability
- Extend the model to **multiple stochastic inflows**

Market_1

An energy company is interested in investing in wind-power production. To assess the viability of this investment, the company wants to evaluate the expected revenue of a wind-power producer in a competitive spot market.

Suppose there are five energy producers participating in the day-ahead spot market. Three of them use conventional sources for energy production, such as coal, while the other two producers use solar and wind power, respectively. The market is cleared once per day for 24 hourly intervals of the following day. For each dispatch interval $t=1, \dots, 24$, every producer submits a bid (b_t, p_t) specifying the energy quantity b_t (MWh) it is willing to supply and the corresponding minimum acceptable price p_t (per MWh). Then, the market operator follows a merit-order process to determine the market-clearing price c_t and dispatch schedule, ensuring the total market demand D_t is met, where $D_t \sim \mathcal{N}(\mu_D, {\sigma_D}^2)$ with constant mean and variance. Merit-order ensures that any bid (b_t, p_t) at interval t, if $p_t \leq c_t$, the bid is accepted, and the producer is committed to delivering b_t at c_t. Otherwise, the producer earns zero revenue.

From the perspective of the wind-power producer, the bid (b_{it}, p_{it}) from a conventional producer i can be modeled as follows: b_{it} can be assumed to be stable, while $p_{it} \sim \mathcal{N}(\mu_{pi}, {\sigma_{pi}}^2)$ with deeply uncertain mean and variance since its bidding strategy is unknown. For the solar-power producer, its b_{st} is associated with the time t and assumed to be $b_{st} = max(0, a+b\cos(\frac{2\pi t}{24}))$, where a and b can be forecast based on historical data thereby assumed to be constants. Its p_{st} is also uncertain, where $p_{st} \sim \mathcal{N}(\mu_{ps}, {\sigma_{ps}}^2)$ with deeply uncertain mean and variance.

The actual energy production of the wind-power producer G_t is also uncertain, as it depends on weather conditions. The production G_t can be forecast from historical data and is modeled as $G_t \sim \mathcal{N}(\mu_G, {\sigma_G}^2)$ with constant mean and variance. In this case, the actual production may differ from the quantity committed to dispatch one day in advance. If the producer under-delivers relative to its dispatched quantity, the shortfall incurs a penalty of q_u per MWh. This cost is imposed by the market operator to maintain the grid's safe operational conditions and to ensure supply-demand balance in real time.

The energy company aims to build a simulation model to evaluate the expected revenue of the wind-power producer on any future day by selecting the hourly bid quantity and price (b_{wt}, p_{wt}), while accounting for uncertain generation and stochastic market clearing prices. This model can then be used to explore outcomes across various short-term and long-term scenarios, thereby informing the viability of the investment.

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 clean **formalization of the simulation model** from the wind-power producer's perspective. The structure is suitable for Monte-Carlo simulation and later extension to optimization or robust analysis.

1. Time Structure and Market Setting

- **Time horizon:** One day, discretized into hourly intervals

$$t = 1, \ldots, 24$$

- **Market:** Competitive day-ahead spot market with **merit-order clearing** and **uniform pricing**.
- **Participants:**
  - 3 conventional producers $i \in \mathcal{C} = \{1, 2, 3\}$
  - 1 solar producer s
  - 1 wind producer w (the decision maker)

2. State Variables

At each hour t, the system state consists of exogenous realizations:

$$S_t = (D_t,\ G_t,\ \{p_{it}\}_{i \in \mathcal{C}},\ p_{st})$$

where:

- D_t: total market demand
- G_t: realized wind generation
- p_{it}: bid prices of conventional producers
- p_{st}: bid price of solar producer

The state is **fully stochastic** and independent across hours.

3. Decision Variables (Wind Producer)

For each hour t, the wind producer chooses:

$$(b_{wt},\ p_{wt})$$

where:

- $b_{wt} \geq 0$: quantity offered (MWh)
- $p_{wt} \geq 0$: minimum acceptable price (€/MWh)

These are chosen **day-ahead**, before uncertainties are realized.

4. Deterministic Parameters

Market and Penalties

- $q_u > 0$: penalty per MWh of under-delivery

Solar Generation Profile

$$b_{st} = \max\left(0,\ a + b\cos\left(\frac{2\pi t}{24}\right)\right)$$

with constants a, b.

Conventional Producer Quantities

- $b_{it} = \bar{b}_i$, constant and known

---

5. Stochastic Variables (Aleatory Uncertainty)

Market Demand

$$D_t \sim \mathcal{N}(\mu_D, \sigma_D^2)$$

Wind Generation

$$G_t \sim \mathcal{N}(\mu_G, \sigma_G^2)$$

Conventional Producer Bid Prices

$$p_{it} \sim \mathcal{N}(\mu_{pi}, \sigma_{pi}^2), \quad i \in \mathcal{C}$$

Solar Producer Bid Price

$$p_{st} \sim \mathcal{N}(\mu_{ps}, \sigma_{ps}^2)$$

All stochastic variables are assumed **independent across producers and time**, unless extended later.

---

6. Deep Uncertainties (Epistemic Uncertainty)

The following parameters are **poorly known** and treated as scenario-dependent inputs:

- μ_{pi}, σ_{pi}^2: conventional bidding behavior
- μ_{ps}, σ_{ps}^2: solar bidding behavior

These define **outer-loop scenarios** in the simulation (e.g., strategic vs aggressive bidding).

---

7. Market Clearing Mechanism (Transition Function)

Merit-Order Clearing

For each hour t:

1. Collect all bids:

$$\mathcal{B}_t = \{(b_{it}, p_{it}),\ (b_{st}, p_{st}),\ (b_{wt}, p_{wt})\}$$

2. Sort bids by increasing price.
3. Accept bids sequentially until total accepted quantity meets D_t.
4. Market-clearing price:

$$c_t = \max\{p : (b, p) \text{ is accepted}\}$$

---

8. Wind Producer Dispatch and Imbalance

Accepted Quantity

$$x_{wt} = \begin{cases} b_{wt}, & p_{wt} \le c_t \\ 0, & p_{wt} > c_t \end{cases}$$

Under-Delivery

$$u_t = \max(0, x_{wt} - G_t)$$

---

9. Revenue Function

Hourly Revenue

$$R_t = c_t \cdot x_{wt} - q_u \cdot u_t$$

Daily Revenue

$$R = \sum_{t=1}^{24} R_t$$

---

10. Objective Function

The wind producer seeks to maximize **expected daily revenue**:

$$\max_{\{b_{wt}, p_{wt}\}_{t=1}^{24}} \mathbb{E}[R] = \mathbb{E}\left[\sum_{t=1}^{24} (c_t x_{wt} - q_u \max(0, x_{wt} - G_t))\right]$$

Expectation is taken over:

$$(D_t,\ G_t,\ p_{it},\ p_{st})$$

---

11. Simulation Architecture

Inner Loop (Monte Carlo)

- Sample stochastic variables
- Clear market
- Compute revenues

Outer Loop (Scenarios)

- Vary deep uncertainty parameters
- Stress-test bidding strategies
- Assess revenue distributions, downside risk, and robustness

---

12. Outputs of Interest

- Expected daily revenue
- Revenue variance and downside risk
- Probability of under-delivery
- Sensitivity to penalty q_u
- Impact of competing renewable penetration

Summary

This formulation yields a **two-layer stochastic simulation model**:

- **Aleatory uncertainty:** weather, demand, price noise
- **Epistemic uncertainty:** competitor bidding behavior

It is directly extensible to:

- Risk-averse objectives (e.g. CVaR)
- Multi-day investment analysis
- Strategic bidding or learning models

If you'd like, I can also:

- Translate this into a **mathematical program**
- Provide **pseudo-code** for Monte-Carlo simulation
- Extend it to **robust or distributionally robust optimization**

The means and variances of the market demand D_t and the wind generation G_t are also constants. Please refine your specification.

Below is a **refined and clarified specification** of the simulation model, explicitly distinguishing **constants**, **stochastic variables**, and **deep uncertainties**, and incorporating your clarification that the **means and variances of market demand and wind generation are also constants**.

---

1. Time and Market Structure

- **Time index:**

$$t = 1, \ldots, 24$$

- **Market:** Day-ahead, hourly, uniform-price spot market cleared via **merit order**.
- **Producers:**
  - Conventional: $i \in \mathcal{C} = \{1, 2, 3\}$
  - Solar: s
  - Wind: w (decision maker)

---

2. Constant Parameters (Known and Fixed)

These parameters are **assumed known and fixed** in the simulation.

Demand

$$D_t \sim \mathcal{N}(\mu_D, \sigma_D^2)$$

- μ_D: mean hourly demand
- σ_D^2: variance of hourly demand

Wind Generation

$$G_t \sim \mathcal{N}(\mu_G, \sigma_G^2)$$

- μ_G: mean wind generation
- σ_G^2: variance of wind generation

Solar Production Profile

$$b_{st} = \max\left(0,\ a + b\cos\left(\frac{2\pi t}{24}\right)\right)$$

with constants a, b.

Conventional Producer Quantities

$$b_{it} = \bar{b}_i, \quad i \in \mathcal{C}$$

Imbalance Penalty

- $q_u > 0$: under-delivery penalty (€/MWh)

---

3. Deep Uncertainties (Epistemic)

The following parameters are **not known with confidence** and define **scenarios** in the simulation:

Conventional Producer Bidding

$$p_{it} \sim \mathcal{N}(\mu_{pi}, \sigma_{pi}^2), \quad i \in \mathcal{C}$$

- μ_{pi}, σ_{pi}^2: unknown bidding behavior parameters

Solar Producer Bidding

$$p_{st} \sim \mathcal{N}(\mu_{ps}, \sigma_{ps}^2)$$

- μ_{ps}, σ_{ps}^2: unknown solar bidding behavior

These parameters vary across **long-term scenarios**, but remain fixed within a scenario.

---

4. Stochastic Variables (Aleatory Uncertainty)

Conditioned on fixed parameters, the following variables are random and independently sampled for each hour:

$$\begin{aligned}
D_t &\sim \mathcal{N}(\mu_D, \sigma_D^2) \\
G_t &\sim \mathcal{N}(\mu_G, \sigma_G^2) \\
p_{it} &\sim \mathcal{N}(\mu_{pi}, \sigma_{pi}^2) \\
p_{st} &\sim \mathcal{N}(\mu_{ps}, \sigma_{ps}^2)
\end{aligned}$$

---

5. Decision Variables (Wind Producer)

Chosen **day-ahead**:

$$(b_{wt},\, p_{wt}), \quad t = 1, \ldots, 24$$

where:

- $b_{wt} \geq 0$: bid quantity (MWh)

- $p_{wt} \geq 0$: bid price (€/MWh)

6. State Variables

At each hour t, the realized state is:

$$S_t = (D_t,\ G_t,\ \{p_{it}\}_{i \in \mathcal{C}},\ p_{st})$$

No intertemporal state dependence is assumed.

7. Market Clearing (Transition Function)

Given bids:

$$\mathcal{B}_t = \{(b_{it}, p_{it})\}_{i \in \mathcal{C}} \cup (b_{st}, p_{st}) \cup (b_{wt}, p_{wt})$$

1. Sort bids by ascending price.
2. Accept bids until cumulative supply meets D_t.
3. Market-clearing price:

$$c_t = \max\{p : (b, p) \text{ accepted at } t\}$$

8. Wind Dispatch and Imbalance

Accepted Quantity

$$x_{wt} = \begin{cases} b_{wt}, & p_{wt} \leq c_t \\ 0, & p_{wt} > c_t \end{cases}$$

Under-Delivery

$$u_t = \max(0, x_{wt} - G_t)$$

9. Revenue Model

Hourly Revenue

$$R_t = c_t\, x_{wt} - q_u\, u_t$$

Daily Revenue

$$R = \sum_{t=1}^{24} R_t$$

OpenAI

10. Objective Function

The wind producer seeks to maximize **expected daily revenue**:

$$\max_{\{b_{wt}, p_{wt}\}_{t=1}^{24}} \mathbb{E}\left[\sum_{t=1}^{24} (c_t x_{wt} - q_u \max(0, x_{wt} - G_t))\right]$$

Expectation is taken over:

$$(D_t,\ G_t,\ p_{it},\ p_{st})$$

---

11. Simulation Structure (Final)

Inner Loop (Short-Term Uncertainty)

- Sample D_t, G_t, p_{it}, p_{st}
- Clear market
- Compute revenues

Outer Loop (Long-Term Scenarios)

- Vary deep uncertainty parameters $(\mu_{pi}, \sigma_{pi}^2, \mu_{ps}, \sigma_{ps}^2)$
- Evaluate robustness and downside risk

---

12. Key Refinement Summary

Component	Classification
μ_D, σ_D^2	**Constants**
μ_G, σ_G^2	**Constants**
D_t, G_t	Aleatory stochastic variables
Competitor price parameters	**Deep uncertainty**
Wind bids (b_{wt}, p_{wt})	Decision variables

If you want, I can now:

- Reformulate this as a **two-stage stochastic program**
- Add **correlation structures** (e.g. wind–solar, demand–price)
- Introduce **risk measures** (CVaR, chance constraints)

Various stakeholders approach this problem from different perspectives. Please refine the model specification from the perspectives of (1) the wind-power producer; and (2) the system 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 **refined multi-stakeholder specification** that cleanly separates the **common environment** from the **individual decision problems** of
(1) the **wind-power producer**, and
(2) the **system regulator**.

The two stakeholders **share the same stochastic market environment**, but **control different decision variables** and pursue **distinct objectives**.

I. Common Environment (Shared by All Stakeholders)

The environment defines the physical system, market rules, uncertainties, and exogenous parameters. All agents observe the same realizations but act independently.

I.1 Time and Market Structure

- Time horizon:

$$t = 1, \dots, 24$$

- Market:
  - Day-ahead, hourly spot market
  - Uniform pricing
  - Merit-order clearing
  - No strategic coordination among producers

I.2 Global Constant Parameters

These parameters are **known and fixed**.

Demand Distribution

$$D_t \sim \mathcal{N}(\mu_D, \sigma_D^2)$$

- μ_D: mean hourly demand
- σ_D^2: variance of hourly demand

Wind Generation Distribution

$$G_t \sim \mathcal{N}(\mu_G, \sigma_G^2)$$

- μ_G: mean wind generation
- σ_G^2: variance of wind generation

Solar Production Profile

$$b_{st} = \max\left(0,\ a + b\cos\left(\frac{2\pi t}{24}\right)\right)$$

with constants a, b.

Conventional Production Capacities

$$b_{it} = \bar{b}_i, \quad i \in \mathcal{C} = \{1, 2, 3\}$$

I.3 Deep Uncertainties (Epistemic)

These parameters are **unknown to all agents** and define **long-term scenarios**:

- Conventional bidding behavior:

$$p_{it} \sim \mathcal{N}(\mu_{pi}, \sigma_{pi}^2)$$

- Solar bidding behavior:

$$p_{st} \sim \mathcal{N}(\mu_{ps}, \sigma_{ps}^2)$$

The parameters

$$(\mu_{pi}, \sigma_{pi}^2, \mu_{ps}, \sigma_{ps}^2)$$

are fixed within a scenario but vary across scenarios.

I.4 Stochastic Variables (Aleatory)

Conditioned on parameters, hourly random variables:

$$\{D_t,\ G_t,\ p_{it},\ p_{st}\}$$

Independent across hours.

I.5 Market Clearing (Shared Transition Rule)

At each hour t:

1. Collect bids:

$$\mathcal{B}_t = \{(b_{it}, p_{it})\}_{i \in \mathcal{C}} \cup (b_{st}, p_{st}) \cup (b_{wt}, p_{wt})$$

2. Sort bids by ascending price.
3. Accept bids until total accepted supply meets D_t.
4. Market-clearing price:

$$c_t = \max\{p : (b, p) \text{ accepted}\}$$

II. Perspective 1: Wind-Power Producer

The wind producer is a **profit-seeking agent** operating under uncertainty.

II.1 State Variables (Observed)

At hour t:

$$S_t^{(w)} = (D_t,\ c_t,\ G_t)$$

II.2 Decision Variables

Chosen **day-ahead**:

$$(b_{wt}, p_{wt}), \quad t = 1, \ldots, 24$$

- $b_{wt} \geq 0$: bid quantity
- $p_{wt} \geq 0$: bid price

II.3 Transition Functions

Dispatch Decision

$$x_{wt} = \begin{cases} b_{wt}, & p_{wt} \leq c_t \\ 0, & p_{wt} > c_t \end{cases}$$

Imbalance

$$u_t = \max(0, x_{wt} - G_t)$$

II.4 Objective Functions

The wind producer may pursue **multiple objectives**.

Objective 1: Maximize Expected Profit

$$\max \ \mathbb{E}\left[\sum_{t=1}^{24}(c_t x_{wt} - q_u u_t)\right]$$

Objective 2: Minimize Downside Risk (Reliability Exposure)

$$\min \ \mathbb{E}\left[\sum_{t=1}^{24} u_t\right]$$

Objective 3: Minimize Revenue Variability

$$\min \ \mathrm{Var}\left(\sum_{t=1}^{24} c_t x_{wt}\right)$$

(Handled via Pareto analysis, not scalarization.)

III. Perspective 2: System Regulator

The regulator **does not bid energy**, but **sets market rules** to ensure system reliability and economic efficiency.

III.1 State Variables (Observed)

$$S_t^{(r)} = (D_t, \ \{x_{jt}\}, \ G_t, \ u_t, \ c_t)$$

where j indexes all producers.

III.2 Decision Variables

The regulator controls **policy parameters**, fixed for the day:

- Imbalance penalty:

$$q_u \geq 0$$

- (Optional extensions)
  - Price caps $c^{\max}$
  - Renewable priority rules
  - Reserve requirements

III.3 Transition Functions

The regulator **does not alter clearing directly**, but influences outcomes through q_u, which affects:

- Wind bidding behavior
- Under-delivery incentives
- Real-time system balance

III.4 Objective Functions

The regulator has **multiple, non-commensurate objectives**.

Objective 1: Minimize Expected Supply Shortfall

$$\min \ \mathbb{E}\left[\sum_{t=1}^{24} \max(0, D_t - \sum_j x_{jt})\right]$$

Objective 2: Minimize Total System Imbalance

$$\min \ \mathbb{E}\left[\sum_{t=1}^{24} \sum_j |x_{jt} - G_{jt}|\right]$$

Objective 3: Minimize Consumer Cost

$$\min \ \mathbb{E}\left[\sum_{t=1}^{24} c_t D_t\right]$$

Objective 4: Support Renewable Integration

$$\max \ \mathbb{E}\left[\sum_{t=1}^{24} x_{wt}\right]$$

IV. Interaction Structure Summary

Layer	Description
Environment	Demand, generation, bidding distributions, clearing rules
Wind Producer	Chooses (b_{wt}, p_{wt}) to trade profit vs risk
Regulator	Chooses q_u to trade efficiency vs reliability
Coupling	Wind decisions respond to q_u; system outcomes feed regulator objectives

---

V. Interpretation

- This is a **multi-agent stochastic system** with **shared uncertainty**
- The wind producer solves a **stochastic control / bidding problem**
- The regulator solves a **policy design problem**
- No agent internalizes the other's objective → **strategic misalignment**
- The framework supports:
  - Scenario discovery
  - Robust policy design
  - Market-design stress testing

If you want next steps, I can:

- Reformulate this as a **bi-level stochastic game**
- Add **learning or adaptive bidding**
- Introduce **distributionally robust objectives**

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 composed model** that ensures **internal consistency**, while allowing **modular, perspective-specific evaluation** for the **wind-power producer** and the **system regulator**. The model is structured as a **shared stochastic environment** with **multiple decision-making modules**, each treating the others' decisions as **exogenous inputs**.

Unified Composed Model

1. Global Structure

The composed model is defined as a tuple:

$$\mathcal{M} = \Big(\mathcal{T}, \mathcal{E}, \{\mathcal{A}^k\}_{k\in\{w,r\}}, \mathcal{P}, \{\mathcal{O}^k\}_{k\in\{w,r\}}\Big)$$

where:

- $\mathcal{T}$: time structure
- $\mathcal{E}$: shared environment and state space
- $\mathcal{A}^k$: decision module of stakeholder k
- $\mathcal{P}$: transition and market-clearing dynamics
- $\mathcal{O}^k$: objective functions of stakeholder k

2. Time Structure

$$\mathcal{T} = \{1, \dots, 24\}$$

All decisions are **day-ahead** and all uncertainty resolves **hourly**.

3. Shared Environment and State Space

3.1 Global Parameters (Constants)

$$\Theta = \{\mu_D, \sigma_D^2,\ \mu_G, \sigma_G^2,\ a, b,\ \{\bar{b}_i\}_{i\in\mathcal{C}}\}$$

These parameters are **fixed and known** to all perspectives.

3.2 Deep Uncertainty Parameters (Scenario Inputs)

$$\Xi = \{\mu_{pi}, \sigma_{pi}^2,\ \mu_{ps}, \sigma_{ps}^2\}$$

These parameters are:

- Fixed within a scenario
- Exogenous to all decision makers
- Used for long-term stress testing

3.3 Stochastic Variables

For each hour t:

$$\omega_t = \big(D_t,\ G_t,\ \{p_{it}\}_{i\in\mathcal{C}},\ p_{st}\big)$$

with:

$$\begin{aligned} D_t &\sim \mathcal{N}(\mu_D, \sigma_D^2) \\ G_t &\sim \mathcal{N}(\mu_G, \sigma_G^2) \\ p_{it} &\sim \mathcal{N}(\mu_{pi}, \sigma_{pi}^2) \\ p_{st} &\sim \mathcal{N}(\mu_{ps}, \sigma_{ps}^2) \end{aligned}$$

3.4 Global State

At hour t, the shared system state is:

$$S_t = \big(\omega_t,\ c_t,\ \{x_{jt}\}_{j\in\mathcal{J}},\ \{u_{jt}\}_{j\in\mathcal{J}}\big)$$

where:

- c_t: market-clearing price
- x_{jt}: dispatched quantity
- u_{jt}: imbalance (under-delivery)
- $\mathcal{J} = \mathcal{C} \cup \{s, w\}$

---

4. Decision Modules (Perspective-Specific)

Each perspective controls its own decisions while treating others' decisions as **exogenous parameters**.

4.1 Wind-Power Producer Module ($k = w$)

OpenAI

Decision Variables

$$A^w = \{(b_{wt}, p_{wt})\}_{t=1}^{24}$$

Exogenous Inputs

$$A^r = \{q_u\}$$

(the regulator's penalty choice)

4.2 System Regulator Module ($k = r$)

Decision Variables

$$A^r = \{q_u\}$$

Exogenous Inputs

$$A^w = \{(b_{wt}, p_{wt})\}_{t=1}^{24}$$

(the wind producer's bidding strategy)

---

5. Shared Transition and Market-Clearing Dynamics

5.1 Merit-Order Clearing

For each hour t:

1. Construct bid set:

$$\mathcal{B}_t = \{(\bar{b}_i, p_{it})\}_{i \in \mathcal{C}} \cup (b_{st}, p_{st}) \cup (b_{wt}, p_{wt})$$

2. Sort bids by ascending price.
3. Accept bids until:

$$\sum_j x_{jt} \geq D_t$$

4. Market-clearing price:

$$c_t = \max\{p : (b, p) \text{ accepted}\}$$

5.2 Dispatch and Imbalance

For each producer j:

$$x_{jt} = \begin{cases} b_{jt}, & p_{jt} \leq c_t \\ 0, & \text{otherwise} \end{cases}$$

Wind under-delivery:

$$u_{wt} = \max(0, x_{wt} - G_t)$$

6. Perspective-Specific Objective Functions

The environment produces **common trajectories**, which are evaluated **separately** by each stakeholder.

6.1 Wind-Power Producer Objectives ($\mathcal{O}^w$)

Objective 1: Maximize Expected Profit

$$\max_{A^w} \mathbb{E} \left[\sum_{t=1}^{24} \left(c_t x_{wt} - q_u u_{wt} \right) \right]$$

Objective 2: Minimize Expected Imbalance

$$\min_{A^w} \mathbb{E} \left[\sum_{t=1}^{24} u_{wt} \right]$$

Objective 3: Minimize Revenue Variance

$$\min_{A^w} \mathrm{Var} \left(\sum_{t=1}^{24} c_t x_{wt} \right)$$

6.2 System Regulator Objectives ($\mathcal{O}^r$)

Objective 1: Minimize Expected System Shortfall

$$\min_{A^r} \mathbb{E} \left[\sum_{t=1}^{24} \max \left(0,\ D_t - \sum_j x_{jt} \right) \right]$$

Objective 2: Minimize Total Imbalance

$$\min_{A^r} \mathbb{E} \left[\sum_{t=1}^{24} \sum_j |x_{jt} - G_{jt}| \right]$$

Objective 3: Minimize Consumer Payments

$$\min_{A^r} \mathbb{E} \left[\sum_{t=1}^{24} c_t D_t \right]$$

Objective 4: Maximize Wind Utilization

$$\max_{A^r} \mathbb{E}\left[\sum_{t=1}^{24} x_{wt}\right]$$

---

7. Modularity and Evaluation Logic

- **Single simulation run** produces:

$$\{S_t\}_{t=1}^{24}$$

- Each perspective:
  - Takes other agents' actions as **fixed inputs**
  - Evaluates its **own objectives only**
  - Does not internalize other objectives

This allows:

- Independent policy evaluation
- Counterfactual analysis
- Scenario discovery
- Robustness testing

---

8. Interpretation

This unified model is:

- **Internally consistent** (single environment, single state)
- **Perspective-modular** (separable objectives)
- **Non-cooperative by construction**
- Suitable for:
  - Bi-level or equilibrium extensions
  - Robust decision-making under deep uncertainty
  - Market design stress testing

If you'd like, I can next:

- Reformulate this as a **stochastic game**
- Provide **simulation pseudocode**
- Extend it to **adaptive or learning agents**

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 Python implementation** of the **unified composed model**.
It is written to be **internally consistent**, **perspective-modular**, and **simulation-ready**, while remaining **agnostic about optimization**.

Key design choices:

- A **shared Environment** class encapsulates uncertainty, market clearing, and state evolution.
- Each **Perspective** (Wind Producer, Regulator) is a class with its own objectives.
- Other perspectives' decisions are treated as **exogenous inputs**.
- A **single interface function** runs the simulation and evaluates all objectives.

> ⚠️ This code is a *specification-level implementation*: it is not intended to be executed as-is, but it is complete and logically consistent.

---

1. Core Imports

```python
import numpy as np
```

---

2. Shared Environment

```python
class MarketEnvironment:
    """
    Shared stochastic environment:
    - samples uncertainty
    - clears the market
    - produces system-wide trajectories
    """

    def __init__(self, constants, uncertainties, decisions):
        self.constants = constants
        self.uncertainties = uncertainties
        self.decisions = decisions

        self.T = 24
        self.conventional_ids = [0, 1, 2]

    def sample_uncertainty(self):
        """Sample all stochastic variables for one day."""
```

```
        mu_D, sigma_D = self.constants["mu_D"], self.constants["sigma_D"]
        mu_G, sigma_G = self.constants["mu_G"], self.constants["sigma_G"]

        D = np.random.normal(mu_D, sigma_D, self.T)
        G = np.random.normal(mu_G, sigma_G, self.T)

        p_conv = {}
        for i in self.conventional_ids:
            mu_pi, sigma_pi = self.uncertainties["mu_pi"][i], self.uncertainties["sigma_pi"][i]
            p_conv[i] = np.random.normal(mu_pi, sigma_pi, self.T)

        p_solar = np.random.normal(
            self.uncertainties["mu_ps"],
            self.uncertainties["sigma_ps"],
            self.T
        )

        return D, G, p_conv, p_solar

    def solar_quantity(self, t):
        a, b = self.constants["a"], self.constants["b"]
        return max(0.0, a + b * np.cos(2 * np.pi * (t + 1) / 24))

    def clear_market(self, t, D_t, p_conv_t, p_solar_t, wind_bid):
        """
        Merit-order market clearing for one hour.
        """
        bids = []

        # Conventional producers
        for i in self.conventional_ids:
            bids.append((
                self.constants["b_conv"][i],
                p_conv_t[i],
                f"conv_{i}"
            ))

        # Solar producer
        bids.append((
            self.solar_quantity(t),
            p_solar_t,
            "solar"
        ))

        # Wind producer
        b_w, p_w = wind_bid
        bids.append((b_w, p_w, "wind"))

        # Sort bids by price
        bids.sort(key=lambda x: x[1])

        accepted = {}
        supplied = 0.0
        clearing_price = 0.0

        for qty, price, name in bids:
            if supplied >= D_t:
                break
            accepted[name] = qty
            supplied += qty
            clearing_price = price

        return accepted, clearing_price
```

3. Wind-Power Producer Perspective

```python
class WindProducer:
    """
```

```
    Wind producer perspective:
    evaluates profit and risk objectives
    """

    def __init__(self, decisions, constants):
        self.bids = decisions["wind_bids"]
        self.q_u = decisions["q_u"]
        self.constants = constants

    def evaluate(self, trajectory):
        profits = []
        imbalances = []

        for t in range(24):
            c_t = trajectory["prices"][t]
            x_w = trajectory["dispatch"]["wind"][t]
            G_t = trajectory["wind_generation"][t]

            under_delivery = max(0.0, x_w - G_t)
            profit = c_t * x_w - self.q_u * under_delivery

            profits.append(profit)
            imbalances.append(under_delivery)

        return {
            "expected_profit": np.sum(profits),
            "total_imbalance": np.sum(imbalances),
            "revenue_variance": np.var(profits),
        }
```

---

4. System Regulator Perspective

```python
class SystemRegulator:
    """
    Regulator perspective:
    evaluates system-wide reliability and cost objectives
    """

    def __init__(self):
        pass

    def evaluate(self, trajectory):
        shortages = []
        total_imbalance = []
        consumer_costs = []
        wind_utilization = []

        for t in range(24):
            D_t = trajectory["demand"][t]
            total_supply = sum(
                trajectory["dispatch"][agent][t]
                for agent in trajectory["dispatch"]
            )

            shortage = max(0.0, D_t - total_supply)
            shortages.append(shortage)

            imbalance_t = sum(
                abs(trajectory["dispatch"][agent][t] -
trajectory["realized_generation"].get(agent, 0.0))
                for agent in trajectory["dispatch"]
            )
            total_imbalance.append(imbalance_t)

            consumer_costs.append(trajectory["prices"][t] * D_t)
            wind_utilization.append(trajectory["dispatch"]["wind"][t])

        return {
```

```
            "expected_shortfall": np.sum(shortages),
            "total_system_imbalance": np.sum(total_imbalance),
            "total_consumer_payment": np.sum(consumer_costs),
            "total_wind_dispatch": np.sum(wind_utilization),
        }
```

5. Unified Simulation Interface Function

```python
def simulate_unified_model(decisions, uncertainties, constants):
    """
    Unified simulation interface.

    Parameters
    ----------
    decisions : dict
        - "wind_bids": list of (b_wt, p_wt) for t=1..24
        - "q_u": imbalance penalty

    uncertainties : dict
        - "mu_pi": list of means for conventional bids
        - "sigma_pi": list of std devs for conventional bids
        - "mu_ps": mean solar bid price
        - "sigma_ps": std dev solar bid price

    constants : dict
        - demand, wind, solar, and capacity parameters

    Returns
    -------
    dict
        Objective values for each perspective
    """

    env = MarketEnvironment(constants, uncertainties, decisions)

    D, G, p_conv, p_solar = env.sample_uncertainty()

    trajectory = {
        "demand": D,
        "wind_generation": G,
        "prices": [],
        "dispatch": {"wind": [], "solar": [], "conv_0": [], "conv_1": [], "conv_2": []},
        "realized_generation": {"wind": G},
    }

    for t in range(24):
        accepted, price = env.clear_market(
            t,
            D[t],
            {i: p_conv[i][t] for i in env.conventional_ids},
            p_solar[t],
            decisions["wind_bids"][t],
        )

        trajectory["prices"].append(price)

        for agent in trajectory["dispatch"]:
            trajectory["dispatch"][agent].append(accepted.get(agent, 0.0))

    wind_agent = WindProducer(decisions, constants)
    regulator = SystemRegulator()

    wind_objectives = wind_agent.evaluate(trajectory)
    regulator_objectives = regulator.evaluate(trajectory)

    return {
        "wind_producer_objectives": wind_objectives,
```

```
        "regulator_objectives": regulator_objectives,
    }
```

---

6. What This Implementation Enables

- ✅ Single shared stochastic environment
- ✅ Modular evaluation of each perspective
- ✅ Exogenous treatment of other agents' decisions
- ✅ Easy extension to:
  - Monte Carlo loops
  - Robust or distributionally robust analysis
  - Learning or adaptive strategies

If you want next, I can:

- Add **Monte Carlo wrappers**
- Convert this into a **bi-level optimization template**
- Extend it with **correlated uncertainties**

Please refine your implementation so that any bid with a price less than or equal to the clearing price is accepted, even if the total market demand has already been met.

Good catch — you're right.
The earlier implementation implicitly enforced **partial acceptance** by stopping once demand was met, which is **not consistent with the stated uniform-price, merit-order rule**.

Below is a **refined implementation** that ensures:

> **Any bid with price ≤ clearing price is fully accepted**,
> even if aggregate accepted supply exceeds demand.

I'll first explain the logic change, then give the **corrected code**. Everything else in the model remains internally consistent.

Conceptual Refinement (What Changes)

The correct clearing logic is:

1. **Sort bids by price**
2. **Find the clearing price**:
   - The *lowest price* such that cumulative supply **meets or exceeds demand**
3. **Accept all bids with price ≤ clearing price**
   - No partial acceptance
   - Oversupply is allowed
   - All accepted bids are paid the same clearing price

This mirrors standard **uniform-price auction** logic.

Refined Market-Clearing Implementation

✅ Corrected `clear_market` Method

```python
def clear_market(self, t, D_t, p_conv_t, p_solar_t, wind_bid):
    """
    Merit-order market clearing with uniform pricing:
    all bids with price <= clearing price are accepted.
    """

    bids = []

    # Conventional producers
    for i in self.conventional_ids:
```

```
        bids.append((
            self.constants["b_conv"][i],
            p_conv_t[i],
            f"conv_{i}"
        ))

    # Solar producer
    bids.append((
        self.solar_quantity(t),
        p_solar_t,
        "solar"
    ))

    # Wind producer
    b_w, p_w = wind_bid
    bids.append((b_w, p_w, "wind"))

    # Sort bids by increasing price
    bids.sort(key=lambda x: x[1])

    # Step 1: determine clearing price
    cumulative_supply = 0.0
    clearing_price = None

    for qty, price, _ in bids:
        cumulative_supply += qty
        if cumulative_supply >= D_t:
            clearing_price = price
            break

    # Step 2: accept all bids with price <= clearing price
    accepted = {}
    for qty, price, name in bids:
        if price <= clearing_price:
            accepted[name] = qty
        else:
            accepted[name] = 0.0

    return accepted, clearing_price
```

Why This Fix Matters

This refinement ensures:

- ✔️ **Logical consistency with your model specification**
- ✔️ Wind may be dispatched even when marginal
- ✔️ Oversupply is possible (important for imbalance analysis)
- ✔️ Clearing price is **well-defined and unique**
- ✔️ Regulator objectives (imbalance, consumer cost) are now meaningful

What Remains Unchanged (and Correct)

- Wind under-delivery penalties are still computed **ex post**
- The regulator does **not** ration supply
- Wind, solar, and conventional producers are treated symmetrically
- Perspective modularity is preserved

If you'd like, I can now:

- Add **partial acceptance as an optional market design variant**
- Make clearing price **robust to extreme demand realizations**
- Add **price caps or scarcity pricing rules**

Market_brief_1

An energy company is interested in investing in wind-power production. To assess the viability of this investment, the company wants to evaluate the expected revenue of a wind-power producer in a competitive spot market.

Suppose there are five energy producers participating in the day-ahead spot market. Three of them use conventional sources for energy production, such as coal, while the other two producers use solar and wind power, respectively. The market is cleared once per day for 24 hourly intervals of the following day. For each dispatch interval, every producer submits a bid specifying the energy quantity (MWh) it is willing to supply and the corresponding minimum acceptable price (per MWh). Then, the market operator follows a merit-order process to determine the market-clearing price and dispatch schedule, ensuring the total market demand is met. If a bid is accepted, the producer is committed to delivering the bid quantity at the market-clearing price. Otherwise, the producer earns zero revenue.

The energy company aims to build a simulation model to evaluate the expected revenue of the wind-power producer on any future day by selecting the hourly bid quantity and price, while accounting for uncertainties. This model can then be used to explore outcomes across various short-term and long-term scenarios, thereby informing the viability of the investment.

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 clean **formalization of the wind-power producer revenue problem as a simulation model** suitable for scenario analysis, Monte Carlo simulation, and policy evaluation. The emphasis is on *decision making under uncertainty* in a competitive electricity spot market.

---

1. Modeling Framework

We model the problem as a **stochastic simulation of a day-ahead electricity market**, repeated over many simulated days.
Each simulated day consists of **24 hourly market-clearing problems** with uncertain demand, renewable availability, and competitor bids.

The wind producer is a **price-taker but strategic bidder**, choosing hourly bid quantities and prices to maximize expected revenue.

---

2. Time Structure

- **Decision horizon**: One day ahead

- **Time steps**:

$$t \in \{1, 2, \dots, 24\} \quad \text{(hourly dispatch intervals)}$$

- **Simulation horizon**: Many days (Monte Carlo or scenario-based)

---

3. State Variables

State variables describe the system conditions at each hour t:

$$S_t = \left(D_t,\ W_t,\ S_t^{\text{solar}},\ B_{-w,t}\right)$$

Where:

- D_t: electricity demand (MWh)
- W_t: available wind generation capacity (MWh)
- S_t^{solar}: available solar generation
- $B_{-w,t}$: competitors' bid stacks (coal + solar)

These states are **partially observed** and uncertain at bidding time.

---

4. Decision Variables (Wind Producer)

For each hour t, the wind producer chooses:

$$x_t = (q_t^w, p_t^w)$$

Where:

- q_t^w: bid quantity (MWh)
- p_t^w: bid price (€/MWh)

Subject to:

$$0 \le q_t^w \le W_t$$

---

5. Stochastic Variables

Uncertainty is represented by random variables:

Short-Term (Operational) Uncertainty

- $D_t \sim F_D(t)$: demand uncertainty
- $W_t \sim F_W(t)$: wind availability
- $S_t^{\text{solar}} \sim F_S(t)$: solar availability

- $B_{-w,t} \sim F_B(t)$: competitor bids

Correlations

- Temporal correlation (e.g., wind persistence)
- Cross-correlation (e.g., wind and solar anticorrelation)

---

6. Market-Clearing Transition Function

For each hour t, the market operator applies a **merit-order dispatch**:

$$(P_t^*, \delta_{i,t}) = \mathcal{M}\big(D_t,\ \{(q_{i,t}, p_{i,t})\}_{i=1}^5\big)$$

Where:

- P_t^*: market-clearing price
- $\delta_{i,t} \in \{0, 1\}$: acceptance indicator of producer i

The wind producer is accepted if:

$$p_t^w \leq P_t^*$$

---

7. Revenue Function

Hourly revenue of the wind producer:

$$R_t = \delta_{w,t} \cdot q_t^w \cdot P_t^*$$

Daily revenue:

$$R = \sum_{t=1}^{24} R_t$$

---

8. Objective Function

The wind producer seeks to maximize **expected daily revenue**:

$$\max_{\{q_t^w, p_t^w\}_{t=1}^{24}} \mathbb{E}\left[\sum_{t=1}^{24} \delta_{w,t} \cdot q_t^w \cdot P_t^*\right]$$

Expectation is taken over all stochastic variables.

---

9. Constant Parameters

Parameter	Description
$N = 5$	Number of producers
$T = 24$	Dispatch intervals
c_i	Marginal cost of conventional producers
$\underline{p}$	Market price floor
$\overline{p}$	Market price cap
Network assumptions	No congestion, single node

10. Deep Uncertainties (Structural & Long-Term)

These are **not probabilistically well-defined** and require scenario analysis:

Market Design

- Price caps and scarcity pricing rules
- Renewable subsidies or penalties
- Carbon pricing

Competitor Behavior

- Strategic vs marginal-cost bidding
- Entry/exit of generators
- Capacity expansion or retirement

Technology & Climate

- Long-term wind patterns
- Turbine efficiency improvements
- Climate-driven demand shifts

Policy & Regulation

- Renewable mandates
- Capacity markets
- Market coupling across regions

11. Simulation Outputs

The simulation model can generate:

- Expected daily and annual revenue

- Revenue distributions (risk measures)
- Sensitivity to bidding strategies
- Stress tests under extreme scenarios

---

12. Use of the Model

This framework supports:

- **Investment viability analysis**
- **Bid-strategy optimization**
- **Risk and downside exposure analysis**
- **Robust decision-making under deep uncertainty**

If you'd like, I can next:

- Convert this into a **Markov Decision Process (MDP)**
- Show **pseudo-code or Python structure**
- Extend it to **storage + wind**
- Add **risk-averse objectives (CVaR)**

You can assume that the market demand follows a normal distribution with constant mean and variance. Please refine your specification accordingly.

Below is a **refined and more explicit specification** of the simulation model incorporating the assumption that **market demand follows a Normal distribution with constant mean and variance**. I keep the structure rigorous so it can be directly implemented.

---

1. Time and Scope (unchanged)

- Dispatch intervals:

$$t = 1, 2, \ldots, 24$$

- Simulation evaluates one day-ahead market and is repeated over many simulated days.

---

2. State Variables (Refined)

At each hour t, the system state is:

$$S_t = \left(D_t,\ W_t,\ S_t^{\text{solar}},\ B_{-w,t}\right)$$

Where:

- D_t: system demand (MWh)
- W_t: wind availability (MWh)
- S_t^{solar}: solar availability (MWh)
- $B_{-w,t}$: competitors' bid curves

---

3. Stochastic Variables (Refined)

3.1 Demand Uncertainty (Key Refinement)

Demand is modeled as an **hourly independent normal random variable**:

$$D_t \sim \mathcal{N}(\mu_D, \sigma_D^2)$$

with:

- μ_D: constant mean demand (MWh)
- σ_D^2: constant variance

To ensure physical feasibility:

$$D_t := \max(0, D_t)$$

(or equivalently, a truncated normal distribution).

Optional refinement (still consistent with your assumption):

- Hourly demand is i.i.d. across t
- No demand learning between hours

3.2 Renewable Availability

- Wind availability:

$$W_t \sim F_W(t)$$

(e.g., Weibull-derived or empirical distribution)

- Solar availability:

$$S_t^{\text{solar}} \sim F_S(t)$$

These may be independent or correlated with each other, but **independent of demand** under the simplified assumption.

3.3 Competitor Bids

Competitor bids are modeled as random bid stacks:

$$B_{-w,t} \sim F_B$$

Each bid consists of price–quantity pairs reflecting:

- Marginal-cost bidding for coal
- Near-zero bids for solar

---

4. Decision Variables (Wind Producer)

For each hour t, the wind producer selects:

$$x_t = (q_t^w, p_t^w)$$

Subject to:

$$0 \le q_t^w \le W_t$$

Decisions are made **before realizations of** D_t.

---

5. Market-Clearing Transition Function

The market operator clears the market via merit order:

$$(P_t^*, \delta_{i,t}) = \mathcal{M}\left(D_t,\ \{(q_{i,t}, p_{i,t})\}_{i=1}^5\right)$$

Where:

- P_t^*: clearing price
- $\delta_{i,t} \in \{0, 1\}$: acceptance indicator

The wind producer is dispatched if:

$$p_t^w \leq P_t^*$$

---

6. Revenue Function

Hourly revenue:

$$R_t = \delta_{w,t} \cdot q_t^w \cdot P_t^*$$

Daily revenue:

$$R = \sum_{t=1}^{24} R_t$$

---

7. Objective Function (Refined Expectation)

The wind producer maximizes **expected daily revenue**, where expectation is now explicit over the normal demand distribution:

$$\max_{\{q_t^w, p_t^w\}_{t=1}^{24}} \mathbb{E}_{D,W,S,B}\left[\sum_{t=1}^{24} \delta_{w,t}(D_t, B_{-w,t}) \cdot q_t^w \cdot P_t^*(D_t, B_{-w,t})\right]$$

with:

$$D_t \sim \mathcal{N}(\mu_D, \sigma_D^2)$$

---

8. Transition Dynamics (Simplified)

Because demand is i.i.d.:

$$\mathbb{P}(S_{t+1} \mid S_t, x_t) = \mathbb{P}(S_{t+1})$$

This makes the model a **stage-wise independent stochastic program**, not a fully dynamic MDP.

---

9. Constant Parameters (Updated)

Parameter	Description
μ_D	Mean hourly demand
σ_D	Std. dev. of hourly demand
$T = 24$	Number of dispatch intervals
$N = 5$	Number of producers
$\underline{p}, \overline{p}$	Market price bounds
c_i	Marginal costs of coal units

10. Deep Uncertainty (Unchanged but Clearer)

Even with normally distributed demand, **deep uncertainties remain**:

- Whether μ_D and σ_D are stable over years
- Market-rule changes affecting clearing prices
- Strategic bidding by competitors
- Long-term climate effects on wind availability

These are handled via **scenario ensembles**, not probabilities.

11. Key Modeling Implication of Normal Demand

Because demand has **unbounded support**, rare high-demand events:

- Increase clearing prices
- Increase upside revenue for wind
- Create heavy right tails in revenue distribution

This justifies Monte Carlo simulation rather than closed-form evaluation.

If you want, I can next:

- Derive the **acceptance probability** of the wind bid as a function of (μ_D, σ_D)
- Show how to approximate expected revenue analytically
- Extend the objective to **risk-averse metrics** (e.g., CVaR)
- Provide **pseudo-code** for the simulation loop

You can also make the following assumptions regarding the behavioral logic of the producers: (1) the bid quantities of the conventional producers are stable, while their bid prices follow normal distributions with deeply uncertain means and variances; (2) the bid quantities of the solar-power producer can be modeled as $max(0, a+b\cos(\frac{2\pi t}{24}))$, and its bid prices also follow a normal distribution with deeply uncertain mean and variance; and (3) the actual wind-power production is stochastic and follows a normal distribution with constant mean and variance. Please refine your specification accordingly.

Below is a **fully refined, internally consistent simulation specification** that incorporates your new **behavioral assumptions for all producers**, while keeping the model suitable for Monte Carlo analysis and deep-uncertainty exploration.

---

1. Time Structure

- Dispatch intervals:

$$t \in \{1, 2, \ldots, 24\}$$

- Market cleared independently for each hour.
- Simulation repeated over many synthetic days.

---

2. Producers

$$\mathcal{I} = \{\mathrm{Coal}_1, \mathrm{Coal}_2, \mathrm{Coal}_3, \mathrm{Solar}, \mathrm{Wind}\}$$

---

3. State Variables (Refined)

At each hour t, the system state is:

$$S_t = \left(D_t,\ \tilde{W}_t,\ Q_t^{\mathrm{solar}},\ B_{-w,t}\right)$$

Where:

- D_t: market demand
- $\tilde{W}_t$: realized wind production
- Q_t^{solar}: realized solar quantity
- $B_{-w,t}$: competitors' bid prices and quantities

---

4. Stochastic Variables (Explicit)

4.1 Market Demand

$$D_t \sim \mathcal{N}(\mu_D, \sigma_D^2), \quad D_t \geq 0$$

Mean and variance are constant across hours and days.

4.2 Conventional (Coal) Producers

For each coal producer $i \in \{\text{Coal}_1, \text{Coal}_2, \text{Coal}_3\}$:

- **Bid quantity (deterministic):**

$$Q_{i,t} = \bar{Q}_i$$

- **Bid price (stochastic):**

$$P_{i,t} \sim \mathcal{N}(\mu_i, \sigma_i^2)$$

with:

- μ_i, σ_i^2 = **deeply uncertain parameters**, varied across scenarios
- Prices truncated below at marginal cost if desired

4.3 Solar Producer

- **Bid quantity (deterministic, diurnal pattern):**

$$Q_t^{\text{solar}} = \max\left(0, a + b\cos\left(\frac{2\pi t}{24}\right)\right)$$

Parameters:

- a: baseline solar output
- b: diurnal amplitude
- **Bid price (stochastic):**

$$P_t^{\text{solar}} \sim \mathcal{N}(\mu_{\text{solar}}, \sigma_{\text{solar}}^2)$$

with deeply uncertain $\mu_{\text{solar}}, \sigma_{\text{solar}}^2$.

4.4 Wind Producer

- **Actual wind production (stochastic):**

$$\tilde{W}_t \sim \mathcal{N}(\mu_W, \sigma_W^2), \quad \tilde{W}_t \geq 0$$

- **Bid quantity (decision variable):**

$$0 \leq q_t^w \leq \tilde{W}_t$$

- **Bid price (decision variable):**

$$p_t^w \in [\underline{p}, \overline{p}]$$

5. Decision Variables (Wind Producer)

For each hour t:

$$x_t = (q_t^w, p_t^w)$$

Decisions are made **ex ante**, before realizations of:

$$(D_t, P_{i,t}, P_t^{\text{solar}}, \tilde{W}_t)$$

6. Market-Clearing Mechanism

For each hour t, the market operator executes a merit-order dispatch:

$$(P_t^*, \delta_{i,t}) = \mathcal{M}\left(D_t,\ \{(Q_{i,t}, P_{i,t})\}_{i \in \mathcal{I}}\right)$$

Where:

- P_t^*: clearing price
- $\delta_{i,t} \in \{0, 1\}$: acceptance indicator

The wind producer is accepted if:

$$p_t^w \leq P_t^*$$

7. Feasibility and Curtailment

If:

$$q_t^w > \tilde{W}_t$$

then actual delivery is:

$$q_t^{\text{del}} = \min(q_t^w, \tilde{W}_t)$$

(Optionally include imbalance penalties; omitted here for simplicity.)

8. Revenue Function

Hourly revenue of the wind producer:

$$R_t = \delta_{w,t} \cdot q_t^{\text{del}} \cdot P_t^*$$

Daily revenue:

$$R = \sum_{t=1}^{24} R_t$$

9. Objective Function

The wind producer maximizes expected daily revenue:

$$\max_{\{q_t^w, p_t^w\}_{t=1}^{24}} \mathbb{E}\left[\sum_{t=1}^{24} \delta_{w,t}(D_t, P_{-w,t}) \cdot \min(q_t^w, \tilde{W}_t) \cdot P_t^*\right]$$

Expectation taken over:

$$D_t,\ P_{i,t},\ P_t^{\text{solar}},\ \tilde{W}_t$$

10. Constant Parameters (Expanded)

Parameter	Meaning
μ_D, σ_D	Demand mean and std. dev.
$\bar{Q}_i$	Coal bid quantities
a, b	Solar diurnal parameters
μ_W, σ_W	Wind production parameters
$\underline{p}, \overline{p}$	Price bounds

11. Deep Uncertainty Layer (Explicit)

The following parameters are **deeply uncertain** and explored via scenario ensembles:

Behavioral & Cost Uncertainty

- μ_i, σ_i^2 (coal bid prices)
- $\mu_{\text{solar}}, \sigma_{\text{solar}}^2$

Structural & Policy Uncertainty

- Price caps
- Renewable support mechanisms
- Entry/exit of coal units

Climate & Technology

- Long-term changes in μ_W, σ_W

- Solar intermittency patterns

12. Model Class

- **Mathematical type**:
  Two-stage stochastic program (per hour), repeated 24 times
- **Solution method**:
  Monte Carlo simulation with parametric scenario sampling
- **Decision use**:
  Evaluate revenue distributions and robustness of wind investment

Next steps (optional):

- Derive **closed-form acceptance probabilities**
- Add **imbalance penalties or reserve markets**
- Introduce **risk-averse objectives (CVaR)**
- Translate this into **Python or Julia simulation code**

If you want, tell me which direction you'd like to go.

Given that the actual wind-power production is stochastic, it may differ from the quantity committed to dispatch one day in advance. If the wind-power producer under-delivers relative to its dispatched quantity, the shortfall incurs a constant penalty cost per MWh. Please also implement this penalty mechanism. Then, please provide a comprehensive model specification that specifies the key components, including state variables, decision variables, transition functions, objective functions, stochastic variables, constant parameters and deep uncertainties.

Below is a **comprehensive, self-contained simulation model specification** that incorporates **stochastic wind production with an explicit under-delivery penalty**, while fully reflecting all assumptions introduced so far.

---

1. Modeling Framework

- **Type**: Stochastic simulation of a competitive day-ahead electricity spot market
- **Structure**: 24 independent hourly clearing problems per day
- **Purpose**: Evaluate expected revenue and risk of a wind-power producer under operational and deep uncertainty
- **Solution approach**: Monte Carlo simulation + scenario discovery

---

2. Time Structure

- Dispatch intervals:

$$t \in \{1, 2, \dots, 24\}$$

- One simulated day consists of 24 hourly markets.
- Simulation is repeated across many synthetic days.

---

3. Market Participants

$$\mathcal{I} = \{\text{Coal}_1, \text{Coal}_2, \text{Coal}_3, \text{Solar}, \text{Wind}\}$$

---

4. State Variables

The system state at hour t is:

$$S_t = \left(D_t,\ \tilde{W}_t,\ Q_t^{\text{solar}},\ P_{-w,t}\right)$$

Where:

- D_t: realized system demand (MWh)
- $\tilde{W}_t$: realized wind production (MWh)
- Q_t^{solar}: realized solar quantity (MWh)
- $P_{-w,t}$: realized bid prices of coal and solar producers

The state is **revealed after bids are submitted**.

5. Stochastic Variables

5.1 Demand

$$D_t \sim \mathcal{N}(\mu_D, \sigma_D^2), \qquad D_t \geq 0$$

- i.i.d. across hours
- Mean and variance constant across days

5.2 Conventional (Coal) Producers

For each coal producer i:

- **Bid quantity (deterministic):**

$$Q_{i,t} = \bar{Q}_i$$

- **Bid price (stochastic):**

$$P_{i,t} \sim \mathcal{N}(\mu_i, \sigma_i^2)$$

with:

- μ_i, σ_i^2 = **deeply uncertain parameters**

5.3 Solar Producer

- **Bid quantity (deterministic diurnal pattern):**

$$Q_t^{\text{solar}} = \max\left(0, a + b\cos\left(\frac{2\pi t}{24}\right)\right)$$

- **Bid price (stochastic):**

$$P_t^{\text{solar}} \sim \mathcal{N}(\mu_{\text{solar}}, \sigma_{\text{solar}}^2)$$

with deeply uncertain $\mu_{\text{solar}}, \sigma_{\text{solar}}^2$.

5.4 Wind Production

- **Actual production (stochastic):**

$$\tilde{W}_t \sim \mathcal{N}(\mu_W, \sigma_W^2), \qquad \tilde{W}_t \geq 0$$

Mean and variance are constant across hours and days.

---

6. Decision Variables (Wind Producer)

For each hour t, the wind producer selects **day-ahead bids**:

$$x_t = (q_t^w, p_t^w)$$

Where:

- q_t^w: committed dispatch quantity (MWh)
- p_t^w: bid price (€/MWh)

Constraints:

$$0 \leq q_t^w \leq \bar{W}$$

where $\bar{W}$ is installed wind capacity.

Decisions are made **before realizations** of all stochastic variables.

---

7. Market-Clearing Transition Function

For each hour t, the market operator clears the market via merit order:

$$(P_t^*, \delta_{i,t}) = \mathcal{M}\left(D_t,\ \{(Q_{i,t}, P_{i,t})\}_{i \in \mathcal{I}}\right)$$

Where:

- P_t^*: market-clearing price
- $\delta_{i,t} \in \{0, 1\}$: indicator of whether producer i is dispatched

The wind producer is dispatched if:

$$\delta_{w,t} = \mathbb{I}(p_t^w \leq P_t^*)$$

---

8. Delivery, Imbalance, and Penalty Mechanism

8.1 Actual Delivery

If dispatched, actual delivered energy is:

$$q_t^{\text{del}} = \delta_{w,t} \cdot \min(q_t^w, \tilde{W}_t)$$

8.2 Under-Delivery (Shortfall)

The under-delivery quantity is:

$$s_t = \delta_{w,t} \cdot \max(0, q_t^w - \tilde{W}_t)$$

8.3 Penalty Cost

A **constant imbalance penalty** applies per MWh of shortfall:

$$C_t^{\text{pen}} = \pi \cdot s_t$$

Where:

- $\pi > 0$: penalty cost per MWh (€/MWh)

9. Revenue Function (with Penalty)

9.1 Gross Revenue

$$R_t^{\text{gross}} = \delta_{w,t} \cdot q_t^w \cdot P_t^*$$

(Note: payment is based on committed quantity, as typical in day-ahead markets.)

9.2 Net Revenue

$$R_t = R_t^{\text{gross}} - C_t^{\text{pen}}$$

9.3 Daily Revenue

$$R = \sum_{t=1}^{24} R_t$$

10. Objective Function

The wind producer maximizes **expected daily net revenue**:

$$\max_{\{q_t^w, p_t^w\}_{t=1}^{24}} \mathbb{E}\left[\sum_{t=1}^{24} \left(\delta_{w,t} \cdot q_t^w \cdot P_t^* - \pi \cdot \delta_{w,t} \cdot \max(0, q_t^w - \tilde{W}_t)\right)\right]$$

Expectation is taken over:

$$D_t,\ P_{i,t},\ P_t^{\text{solar}},\ \tilde{W}_t$$

11. Transition Dynamics

Because all stochastic variables are i.i.d. across hours:

$$\mathbb{P}(S_{t+1} \mid S_t, x_t) = \mathbb{P}(S_{t+1})$$

Thus:

- No intertemporal coupling
- Model is a **stage-wise independent stochastic program**

12. Constant Parameters

Parameter	Description
μ_D, σ_D	Demand distribution
$\bar{Q}_i$	Coal bid quantities
a, b	Solar diurnal parameters
μ_W, σ_W	Wind production distribution
π	Under-delivery penalty
$\underline{p}, \overline{p}$	Market price bounds
$\bar{W}$	Installed wind capacity

13. Deep Uncertainties

The following parameters are **not probabilistically known** and are explored via scenario ensembles:

Market Behavior

- μ_i, σ_i^2: coal bid prices
- $\mu_{\text{solar}}, \sigma_{\text{solar}}^2$: solar bid prices

Regulation & Market Design

- Penalty magnitude π
- Price caps and imbalance settlement rules

Technology & Climate

- Long-term shifts in μ_W, σ_W
- Structural changes in demand (μ_D, σ_D)

14. Key Modeling Implications

- Higher q_t^w increases expected revenue **and** penalty exposure
- Penalty introduces an **implicit risk-aversion mechanism**
- Optimal bids trade off:
  - Acceptance probability
  - Market price
  - Wind shortfall risk

Summary

This specification yields a **realistic, policy-relevant simulation model** that:

- Captures day-ahead market mechanics
- Represents renewable intermittency
- Explicitly prices imbalance risk
- Supports robust investment and bidding analysis under deep uncertainty

If you'd like, I can now:

- Derive the **optimal bidding logic** under penalties
- Add **risk-averse objectives (CVaR)**
- Translate this into **executable simulation code**
- Extend it to include **storage or balancing markets**

You do not need to consider deep uncertainties arising from long-term shifts in μ_D, σ_D, μ_W or σ_W. You can assume that they are constant over time. Please refine your specification.

Below is a **refined, consolidated model specification** that **removes deep uncertainty about demand and wind statistics** ($\mu_D, \sigma_D, \mu_W, \sigma_W$ are now fixed constants), while **retaining deep uncertainty only where you explicitly allow it** (producer bidding behavior, market rules). The specification is complete and internally consistent.

Comprehensive Simulation Model Specification

(Day-Ahead Electricity Market with Wind Imbalance Penalties)

---

1. Modeling Framework

- **Model class**: Stochastic simulation / two-stage stochastic program
- **Purpose**: Evaluate expected revenue and risk exposure of a wind-power producer bidding into a competitive day-ahead spot market
- **Time resolution**: Hourly
- **Horizon**: One day (24 hours), repeated across simulated days
- **Solution approach**: Monte Carlo simulation with scenario sampling

---

2. Time Structure

$$t \in \{1, 2, \ldots, 24\}$$

Each hour is cleared independently by the market operator.

---

3. Market Participants

$$\mathcal{I} = \{\mathrm{Coal}_1, \mathrm{Coal}_2, \mathrm{Coal}_3, \mathrm{Solar}, \mathrm{Wind}\}$$

---

4. State Variables

The realized system state at hour t is:

$$S_t = \left(D_t,\ \tilde{W}_t,\ Q_t^{\mathrm{solar}},\ P_{-w,t}\right)$$

Where:

- D_t: realized electricity demand (MWh)
- $\tilde{W}_t$: realized wind production (MWh)
- Q_t^{solar}: realized solar bid quantity (MWh)
- $P_{-w,t}$: realized bid prices of coal and solar producers

State variables are revealed **after day-ahead bids are submitted**.

---

5. Stochastic Variables

5.1 Market Demand (Fixed Distribution)

$$D_t \sim \mathcal{N}(\mu_D, \sigma_D^2), \qquad D_t \geq 0$$

- i.i.d. across hours and days
- μ_D, σ_D are **known constants**

5.2 Coal Producers

For each coal producer i:

- **Bid quantity (deterministic)**:

$$Q_{i,t} = \bar{Q}_i$$

- **Bid price (stochastic)**:

$$P_{i,t} \sim \mathcal{N}(\mu_i, \sigma_i^2)$$

with μ_i, σ_i^2 treated as **behavioral parameters**.

5.3 Solar Producer

- **Bid quantity (deterministic diurnal profile)**:

$$Q_t^{\text{solar}} = \max\left(0, a + b\cos\left(\frac{2\pi t}{24}\right)\right)$$

- **Bid price (stochastic)**:

$$P_t^{\text{solar}} \sim \mathcal{N}(\mu_{\text{solar}}, \sigma_{\text{solar}}^2)$$

5.4 Wind Production (Fixed Distribution)

- **Actual wind generation**:

$$\tilde{W}_t \sim \mathcal{N}(\mu_W, \sigma_W^2), \qquad \tilde{W}_t \geq 0$$

- i.i.d. across hours and days
- μ_W, σ_W are **known constants**

---

6. Decision Variables (Wind Producer)

For each hour t, the wind producer submits:

$$x_t = (q_t^w, p_t^w)$$

Where:

- q_t^w: committed dispatch quantity (MWh)
- p_t^w: bid price (€/MWh)

Constraints:

$$0 \leq q_t^w \leq \bar{W}, \qquad \underline{p} \leq p_t^w \leq \bar{p}$$

Decisions are made **day-ahead**, before uncertainty is realized.

---

7. Market-Clearing Transition Function

For each hour t, the market operator applies merit-order dispatch:

$$(P_t^*, \delta_{i,t}) = \mathcal{M}\left(D_t,\ \{(Q_{i,t}, P_{i,t})\}_{i \in \mathcal{I}}\right)$$

Where:

- P_t^*: market-clearing price
- $\delta_{i,t} \in \{0, 1\}$: acceptance indicator

Wind producer is dispatched if:

$$\delta_{w,t} = \mathbb{I}(p_t^w \leq P_t^*)$$

---

8. Delivery, Imbalance, and Penalty Mechanism

8.1 Actual Delivered Energy

$$q_t^{\text{del}} = \delta_{w,t} \cdot \min(q_t^w, \tilde{W}_t)$$

8.2 Under-Delivery (Shortfall)

$$s_t = \delta_{w,t} \cdot \max(0, q_t^w - \tilde{W}_t)$$

8.3 Penalty Cost

$$C_t^{\text{pen}} = \pi \cdot s_t$$

Where:

- π: constant penalty per MWh of shortfall (€/MWh)

---

9. Revenue Function

9.1 Gross Revenue (Day-Ahead Settlement)

$$R_t^{\text{gross}} = \delta_{w,t} \cdot q_t^w \cdot P_t^*$$

9.2 Net Revenue

$$R_t = R_t^{\text{gross}} - C_t^{\text{pen}}$$

9.3 Daily Revenue

$$R = \sum_{t=1}^{24} R_t$$

---

10. Objective Function

The wind producer maximizes expected daily net revenue:

$$\max_{\{q_t^w, p_t^w\}_{t=1}^{24}} \mathbb{E}\left[\sum_{t=1}^{24} \left(\delta_{w,t} \cdot q_t^w \cdot P_t^* - \pi \cdot \delta_{w,t} \cdot \max(0, q_t^w - \tilde{W}_t)\right)\right]$$

Expectation is taken over:

$$D_t,\ P_{i,t},\ P_t^{\text{solar}},\ \tilde{W}_t$$

---

11. Transition Dynamics

Because all stochastic processes are i.i.d.:

$$\mathbb{P}(S_{t+1} \mid S_t, x_t) = \mathbb{P}(S_{t+1})$$

- No intertemporal coupling
- Model is **stage-wise independent**

---

12. Constant Parameters (Final)

Parameter	Description
μ_D, σ_D	Demand distribution (fixed)
μ_W, σ_W	Wind production distribution (fixed)
$\bar{Q}_i$	Coal bid quantities
a, b	Solar diurnal parameters
π	Imbalance penalty
$\bar{W}$	Installed wind capacity
$\underline{p}, \overline{p}$	Price bounds

---

13. Deep Uncertainties (Remaining)

Deep uncertainty is **restricted to behavioral and institutional dimensions**:

Producer Bidding Behavior

- μ_i, σ_i^2: coal bid prices
- $\mu_{\text{solar}}, \sigma^2_{\text{solar}}$: solar bid prices

Market Design

- Penalty level π
- Price caps or scarcity pricing rules

These are explored through **scenario ensembles**, not probability distributions.

---

14. Key Implications

- Wind bidding faces a **risk–reward trade-off** due to penalties
- Over-commitment increases expected revenue but raises imbalance costs
- Model supports:
  - Expected revenue estimation
  - Downside-risk analysis
  - Investment viability assessment

If you want, I can now:

- Derive **analytical acceptance probabilities**
- Add **risk-averse objectives (e.g., CVaR)**
- Convert this into **executable simulation code**

- Extend the framework to **storage or balancing markets**

Just tell me how you'd like to proceed.

Various stakeholders approach this problem from different perspectives. Please refine the model specification from the perspectives of (1) the wind-power producer; and (2) the system 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 **refined, multi-stakeholder model specification** that separates **objectives and controls** while embedding both stakeholders in a **shared market environment**. The wind-power producer and the system regulator act **independently**, each controlling its own decision variables, but interacting through the same stochastic electricity market.

I. Common Environment (Shared by All Stakeholders)

This layer defines the **physical market, stochastic processes, and clearing rules**. It is exogenous to individual stakeholders' objectives.

I.1 Time Structure

- Dispatch intervals:

$$t \in \{1, \dots, 24\}$$

- Each hour is cleared independently.
- Simulation repeated over many synthetic days.

I.2 Market Participants

$$\mathcal{I} = \{\mathrm{Coal}_1, \mathrm{Coal}_2, \mathrm{Coal}_3, \mathrm{Solar}, \mathrm{Wind}\}$$

I.3 Global State Variables

At hour t, the realized environment state is:

$$S_t = \left(D_t,\ \tilde{W}_t,\ Q_t^{\mathrm{solar}},\ P_{-w,t}\right)$$

Where:

- D_t: realized electricity demand (MWh)

- $\tilde{W}_t$: realized wind production (MWh)
- Q_t^{solar}: solar bid quantity (MWh)
- $P_{-w,t}$: realized bid prices of coal and solar producers

These variables are **not controlled** by either stakeholder.

I.4 Stochastic Processes (Fixed Distributions)

Demand

$$D_t \sim \mathcal{N}(\mu_D, \sigma_D^2), \quad D_t \geq 0$$

Wind production

$$\tilde{W}_t \sim \mathcal{N}(\mu_W, \sigma_W^2), \quad \tilde{W}_t \geq 0$$

Coal producers (for each i)

$$Q_{i,t} = \bar{Q}_i, \qquad P_{i,t} \sim \mathcal{N}(\mu_i, \sigma_i^2)$$

Solar producer

$$Q_t^{\text{solar}} = \max\left(0, a + b\cos\frac{2\pi t}{24}\right), \qquad P_t^{\text{solar}} \sim \mathcal{N}(\mu_{\text{solar}}, \sigma_{\text{solar}}^2)$$

I.5 Market-Clearing Transition Function

For each hour t:

$$(P_t^*, \delta_{i,t}) = \mathcal{M}\big(D_t, \{(Q_{i,t}, P_{i,t})\}_{i\in\mathcal{I}}\big)$$

Where:

- P_t^*: market-clearing price
- $\delta_{i,t} \in \{0, 1\}$: dispatch indicator

This transition function is **common and immutable**.

I.6 Global Parameters

Parameter	Description
μ_D, σ_D	Demand distribution
μ_W, σ_W	Wind production distribution
$\bar{Q}_i$	Coal quantities
a, b	Solar profile

Parameter	Description
$\underline{p}, \overline{p}$	Price bounds
$\bar{W}$	Wind capacity

II. Perspective 1: Wind-Power Producer

The wind producer is a **profit-maximizing agent** exposed to imbalance penalties.

II.1 Decision Variables

For each hour t:

$$x_t^w = (q_t^w, p_t^w)$$

Subject to:

$$0 \le q_t^w \le \bar{W}, \qquad \underline{p} \le p_t^w \le \overline{p}$$

Decisions are made **day-ahead**, before uncertainty is realized.

II.2 Producer-Specific Transition Components

Dispatch indicator

$$\delta_{w,t} = \mathbb{I}(p_t^w \le P_t^*)$$

Delivered energy

$$q_t^{\text{del}} = \delta_{w,t} \min(q_t^w, \tilde{W}_t)$$

Shortfall

$$s_t = \delta_{w,t} \max(0, q_t^w - \tilde{W}_t)$$

II.3 Revenue and Cost Functions

Gross revenue

$$R_t^{\text{gross}} = \delta_{w,t}\, q_t^w\, P_t^*$$

Imbalance penalty

$$C_t^{\text{pen}} = \pi\, s_t$$

Net revenue

$$R_t^w = R_t^{\text{gross}} - C_t^{\text{pen}}$$

II.4 Objective Function (Wind Producer)

Objective W1: Expected profit maximization

$$\max_{\{q_t^w, p_t^w\}} \mathbb{E}\left[\sum_{t=1}^{24} R_t^w\right]$$

(Optional extensions such as risk constraints can be added, but are not imposed here.)

III. Perspective 2: System Regulator

The regulator does **not bid energy**. It influences outcomes through **market design parameters** and evaluates system-level performance.

III.1 Decision Variables (Regulatory Controls)

The regulator selects parameters that define market incentives:

$$x^{\text{reg}} = (\pi, \overline{p})$$

Where:

- π: imbalance penalty per MWh
- $\overline{p}$: price cap (optional)

These are **set ex ante** and apply uniformly.

III.2 Regulator-Relevant System Metrics

Total system cost

$$C_t^{\text{sys}} = \sum_{i \in \mathcal{I}} \delta_{i,t} \, Q_{i,t} \, P_t^*$$

Wind shortfall

$$s_t$$

Renewable utilization

$$U_t^{\text{wind}} = \frac{q_t^{\text{del}}}{\tilde{W}_t}$$

III.3 Objective Functions (Multiple, Non-Weighted)

The regulator evaluates **multiple objectives simultaneously**:

Objective R1: Minimize expected imbalance

$$\min_{\pi,\overline{p}} \mathbb{E}\left[\sum_{t=1}^{24} s_t\right]$$

Objective R2: Minimize expected system cost

$$\min_{\pi,\overline{p}} \mathbb{E}\left[\sum_{t=1}^{24} C_t^{\text{sys}}\right]$$

Objective R3: Maximize renewable utilization

$$\max_{\pi,\overline{p}} \mathbb{E}\left[\sum_{t=1}^{24} U_t^{\text{wind}}\right]$$

(No scalarization is imposed; trade-offs are analyzed via Pareto frontiers.)

IV. Interaction Structure

- The **environment** maps:

$$(x_t^w, x^{\text{reg}}, S_t) \longrightarrow (P_t^*, \delta_{i,t}, R_t^w, C_t^{\text{sys}})$$

- The wind producer responds optimally to regulatory parameters.
- The regulator evaluates system outcomes induced by producer behavior.

This defines a **hierarchical but non-cooperative simulation**:

- Regulator: market designer / evaluator
- Wind producer: strategic bidder within given rules

V. Summary

Aspect	Wind Producer	Regulator
Control	q_t^w, p_t^w	$\pi, \overline{p}$
Goal	Maximize expected profit	Cost, reliability, renewables
Risk	Price + imbalance	System-level performance
Interaction	Via market clearing	Via incentive design

If you want, I can next:

- Formulate this as a **bi-level optimization problem**
- Show **policy–response maps** for π
- Implement a **multi-objective simulation experiment**
- Extend the regulator's view to **emissions or reliability metrics**

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 model** that **composes the wind-producer and regulator perspectives into one shared environment**, while preserving **modularity, independence of objectives, and exogeneity of other actors' decisions** for each perspective.

The model is written as a **multi-agent stochastic environment** with **perspective-specific controls and objectives** layered on top of a common state, dynamics, and market-clearing mechanism.

Unified Multi-Perspective Simulation Model

(Day-Ahead Electricity Market with Wind Imbalance Penalties)

---

1. Common Environment (Authoritative Core)

This layer is **shared**, immutable, and used identically by all perspectives.

1.1 Time Structure

$$t \in \{1, \ldots, 24\}$$

- Each hour is cleared independently.
- The day is the fundamental simulation episode.
- Episodes are repeated across Monte Carlo samples.

1.2 Agents

$$\mathcal{A} = \{\text{Wind Producer}, \ \text{System Regulator}\}$$

Other generators (coal, solar) are modeled as **environmental processes**, not decision-making agents.

1.3 Global State Variables

At each hour t, the environment state is:

$$S_t = \left(D_t, \ \tilde{W}_t, \ Q_t^{\text{solar}}, \ P_{-w,t}\right)$$

Where:

- D_t: realized electricity demand (MWh)
- $\tilde{W}_t$: realized wind production (MWh)
- Q_t^{solar}: solar bid quantity (MWh)
- $P_{-w,t}$: realized bid prices of coal and solar producers

These states are **not controlled by any perspective**.

1.4 Stochastic Processes (Fixed)

All stochastic variables are i.i.d. across hours and days.

Demand

$$D_t \sim \mathcal{N}(\mu_D, \sigma_D^2), \quad D_t \geq 0$$

Wind production

$$\tilde{W}_t \sim \mathcal{N}(\mu_W, \sigma_W^2), \quad \tilde{W}_t \geq 0$$

Coal producers (for each $i = 1, 2, 3$)

$$Q_{i,t} = \bar{Q}_i, \qquad P_{i,t} \sim \mathcal{N}(\mu_i, \sigma_i^2)$$

Solar producer

$$Q_t^{\text{solar}} = \max\left(0, a + b\cos\frac{2\pi t}{24}\right), \qquad P_t^{\text{solar}} \sim \mathcal{N}(\mu_{\text{solar}}, \sigma_{\text{solar}}^2)$$

1.5 Market-Clearing Transition Function

For each hour t:

$$(P_t^*, \delta_{i,t}) = \mathcal{M}\left(D_t, \{(Q_{i,t}, P_{i,t})\}_{i \in \mathcal{I}}\right)$$

Where:

- P_t^*: market-clearing price
- $\delta_{i,t} \in \{0, 1\}$: dispatch indicator

This transition function is **global and fixed**.

1.6 Global Parameters

$$\Theta = (\mu_D, \sigma_D, \mu_W, \sigma_W, \bar{Q}_i, a, b, \bar{W}, \underline{p})$$

2. Perspective-Specific Embedding

Each perspective is defined by:

1. Its **decision variables**
2. Its **induced state augmentations**
3. Its **objective functions**
4. **Exogenous treatment** of other perspectives' decisions

2A. Perspective 1 — Wind-Power Producer

2A.1 Decision Variables

For each hour t:

$$x_t^w = (q_t^w, p_t^w)$$

Subject to:

$$0 \leq q_t^w \leq \bar{W}, \qquad \underline{p} \leq p_t^w \leq \overline{p}$$

Exogenous inputs:

- Regulator decisions x^{reg}
- Market prices P_t^*

2A.2 Producer-Specific State Augmentation

From the common environment:

$$\delta_{w,t} = \mathbb{I}(p_t^w \leq P_t^*)$$

$$s_t = \delta_{w,t} \max(0, q_t^w - \tilde{W}_t)$$

2A.3 Producer Revenue Function

$$R_t^w = \delta_{w,t}\, q_t^w\, P_t^* - \pi\, s_t$$

2A.4 Objective (Wind Producer)

W1: Expected profit maximization

$$\max_{\{q_t^w, p_t^w\}} \mathbb{E}\left[\sum_{t=1}^{24} R_t^w\right]$$

Other perspectives' decisions are treated as **fixed parameters**.

2B. Perspective 2 — System Regulator

2B.1 Decision Variables

$$x^{\text{reg}} = (\pi, \overline{p})$$

Where:

- π: imbalance penalty (€/MWh)
- $\overline{p}$: price cap

Exogenous inputs:

- Wind producer bids x_t^w

2B.2 Regulator-Relevant Metrics

Derived from the common environment:

System cost

$$C_t^{\text{sys}} = \sum_{i \in \mathcal{I}} \delta_{i,t} \, Q_{i,t} \, P_t^*$$

Wind shortfall

$$s_t$$

Wind utilization

$$U_t^{\text{wind}} = \frac{\delta_{w,t} \min(q_t^w, \tilde{W}_t)}{\tilde{W}_t}$$

2B.3 Objectives (Regulator)

The regulator evaluates **multiple non-aggregated objectives**:

R1: Minimize expected imbalance

$$\min_{\pi, \overline{p}} \ \mathbb{E}\left[\sum_{t=1}^{24} s_t\right]$$

R2: Minimize expected system cost

$$\min_{\pi, \overline{p}} \ \mathbb{E}\left[\sum_{t=1}^{24} C_t^{\text{sys}}\right]$$

R3: Maximize renewable utilization

$$\max_{\pi,\overline{p}} \mathbb{E}\left[\sum_{t=1}^{24} U_t^{\text{wind}}\right]$$

Wind-producer decisions are treated as **exogenous**.

---

3. Composition Structure

The unified model is a **tuple**:

$$\mathcal{M} = \left(\mathcal{S},\ \mathcal{X}^{w},\ \mathcal{X}^{\text{reg}},\ \mathcal{T},\ \mathcal{O}^{w},\ \mathcal{O}^{\text{reg}}\right)$$

Where:

- $\mathcal{S}$: shared state space
- $\mathcal{X}^{w}$: wind-producer decision space
- $\mathcal{X}^{\text{reg}}$: regulator decision space
- $\mathcal{T}$: common stochastic transition + market clearing
- $\mathcal{O}^{w}$: wind-producer objectives
- $\mathcal{O}^{\text{reg}}$: regulator objectives

---

4. Key Properties of the Composed Model

✔️ **Single shared environment and state**
✔️ **Perspective-specific decision control**
✔️ **Other perspectives treated as exogenous inputs**
✔️ **Multiple objectives preserved without scalarization**
✔️ **Modular evaluation possible for each stakeholder**

---

5. Interpretation

- The wind producer solves a **stochastic profit-maximization problem** inside a fixed market design.
- The regulator evaluates **policy–outcome mappings** across multiple system objectives.
- Together, they form a **multi-agent stochastic environment**, suitable for:
  - bilevel analysis,
  - policy stress testing,
  - robustness and trade-off exploration.

If you'd like, I can next:

- Reformulate this as a **bilevel optimization problem**

- Provide **pseudo-code for a modular simulator**
- Show how to compute **Pareto fronts for regulator objectives**
- Extend the model to **storage, reserves, or emissions**

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 Python implementation** of the **unified multi-perspective model**.
It is structured so that:

- The **environment** is shared and authoritative
- Each **perspective** (wind producer, regulator) is encapsulated in its own class
- Each perspective treats the other's decisions as **exogenous inputs**
- Objectives are evaluated **modularly**
- A single **interface function** runs the simulation and returns all objectives

The code is written for clarity and internal consistency rather than computational efficiency.

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


# ============================================================
# Common Environment
# ============================================================

class ElectricityMarketEnvironment:
    """
    Shared stochastic environment and market-clearing mechanism.
    """

    def __init__(self, constants: Dict[str, Any], uncertainties: Dict[str, Any]):
        self.constants = constants
        self.uncertainties = uncertainties

        self.T = constants["T"]
        self.rng = np.random.default_rng(uncertainties.get("seed", None))

    # ------------------------
    # Stochastic processes
    # ------------------------

    def sample_demand(self):
        mu, sigma = self.constants["mu_D"], self.constants["sigma_D"]
        return max(0.0, self.rng.normal(mu, sigma))

    def sample_wind(self):
        mu, sigma = self.constants["mu_W"], self.constants["sigma_W"]
        return max(0.0, self.rng.normal(mu, sigma))

    def sample_coal_prices(self):
        prices = []
        for mu, sigma in self.uncertainties["coal_price_params"]:
            prices.append(self.rng.normal(mu, sigma))
        return prices
```

```python
    def sample_solar_price(self):
        mu, sigma = self.uncertainties["solar_price_params"]
        return self.rng.normal(mu, sigma)

    def solar_quantity(self, t):
        a, b = self.constants["solar_a"], self.constants["solar_b"]
        return max(0.0, a + b * np.cos(2 * np.pi * t / 24))

    # ------------------------
    # Market clearing
    # ------------------------

    def clear_market(self, demand, bids):
        """
        bids: list of dicts with keys {agent, quantity, price}
        Returns clearing_price, dispatch_dict
        """
        bids_sorted = sorted(bids, key=lambda x: x["price"])
        dispatched = {}
        remaining = demand
        clearing_price = 0.0

        for bid in bids_sorted:
            if remaining <= 0:
                dispatched[bid["agent"]] = 0.0
                continue

            accepted = min(bid["quantity"], remaining)
            dispatched[bid["agent"]] = accepted
            remaining -= accepted
            clearing_price = bid["price"]

        return clearing_price, dispatched


# ============================================================
# Wind Producer Perspective
# ============================================================

class WindProducer:
    """
    Wind producer decision logic and objective evaluation.
    """

    def __init__(self, decisions: Dict[str, Any], constants: Dict[str, Any]):
        self.q = decisions["wind_quantity"]      # list of length T
        self.p = decisions["wind_price"]         # list of length T
        self.capacity = constants["wind_capacity"]

    def bid(self, t):
        return {
            "agent": "wind",
            "quantity": min(self.q[t], self.capacity),
            "price": self.p[t],
        }

    def evaluate_objective(self, prices, dispatch, wind_realizations, penalty):
        revenue = 0.0
        for t in range(len(prices)):
            dispatched = dispatch[t].get("wind", 0.0)
            shortfall = max(0.0, dispatched - wind_realizations[t])
            revenue += dispatched * prices[t] - penalty * shortfall
        return revenue


# ============================================================
# Regulator Perspective
# ============================================================

class SystemRegulator:
    """
    Regulator policy parameters and multi-objective evaluation.
    """
```

```
    def __init__(self, decisions: Dict[str, Any]):
        self.penalty = decisions["penalty"]
        self.price_cap = decisions.get("price_cap", np.inf)

    def apply_price_cap(self, price):
        return min(price, self.price_cap)

    def evaluate_objectives(self, dispatch, prices, wind_realizations):
        imbalance = 0.0
        system_cost = 0.0
        wind_utilization = 0.0

        for t in range(len(prices)):
            for agent, qty in dispatch[t].items():
                system_cost += qty * prices[t]

            wind_dispatch = dispatch[t].get("wind", 0.0)
            shortfall = max(0.0, wind_dispatch - wind_realizations[t])
            imbalance += shortfall

            if wind_realizations[t] > 0:
                wind_utilization += min(wind_dispatch, wind_realizations[t]) /
wind_realizations[t]

        return {
            "expected_imbalance": imbalance,
            "expected_system_cost": system_cost,
            "expected_wind_utilization": wind_utilization,
        }


# ============================================================
# Unified Simulation Interface
# ============================================================

def simulate_unified_model(
    decision_vars: Dict[str, Dict[str, Any]],
    uncertain_params: Dict[str, Any],
    constant_params: Dict[str, Any],
):
    """
    Interface function.

    Inputs
    ------
    decision_vars:
        {
            "wind": {
                "wind_quantity": [...],
                "wind_price": [...]
            },
            "regulator": {
                "penalty": float,
                "price_cap": float (optional)
            }
        }

    uncertain_params:
        {
            "coal_price_params": [(mu1, sigma1), (mu2, sigma2), (mu3, sigma3)],
            "solar_price_params": (mu, sigma),
            "seed": int (optional)
        }

    constant_params:
        {
            "T": 24,
            "mu_D": float,
            "sigma_D": float,
            "mu_W": float,
            "sigma_W": float,
            "coal_quantities": [Q1, Q2, Q3],
            "solar_a": float,
            "solar_b": float,
            "wind_capacity": float
```

```
    }

    Returns
    -------
    Dictionary with objective values for each perspective.
    """

    # Initialize components
    env = ElectricityMarketEnvironment(constant_params, uncertain_params)
    wind = WindProducer(decision_vars["wind"], constant_params)
    regulator = SystemRegulator(decision_vars["regulator"])

    prices = []
    dispatches = []
    wind_realizations = []

    # Simulation loop
    for t in range(constant_params["T"]):
        demand = env.sample_demand()
        wind_real = env.sample_wind()
        wind_realizations.append(wind_real)

        coal_prices = env.sample_coal_prices()
        solar_price = env.sample_solar_price()
        solar_qty = env.solar_quantity(t)

        bids = []

        # Coal bids
        for i, (q, p) in enumerate(zip(constant_params["coal_quantities"], coal_prices)):
            bids.append({"agent": f"coal_{i}", "quantity": q, "price": p})

        # Solar bid
        bids.append({"agent": "solar", "quantity": solar_qty, "price": solar_price})

        # Wind bid
        bids.append(wind.bid(t))

        price, dispatch = env.clear_market(demand, bids)
        price = regulator.apply_price_cap(price)

        prices.append(price)
        dispatches.append(dispatch)

    # Objective evaluation
    wind_profit = wind.evaluate_objective(
        prices, dispatches, wind_realizations, regulator.penalty
    )

    regulator_objectives = regulator.evaluate_objectives(
        dispatches, prices, wind_realizations
    )

    return {
        "wind_expected_profit": wind_profit,
        **regulator_objectives,
    }
```

Notes on Design

- **Single shared environment** (**`ElectricityMarketEnvironment`**)
- **Perspective-specific classes** with independent objectives
- Each perspective treats the other’s decisions as **exogenous**
- The interface function:
    - accepts decisions, uncertainties, and constants

- simulates the shared dynamics
- returns **all objective values separately**

This structure is directly extensible to:

- bilevel optimization,
- multi-objective policy analysis,
- scenario ensembles,
- additional agents (storage, balancing market, emissions).

If you'd like, I can now:

- add **CVaR objectives**
- convert this into a **bilevel solver skeleton**
- provide **unit-test stubs**
- or refactor it into a **reinforcement-learning environment**

Please refine your implementation so that any bid with a price less than or equal to the clearing price is accepted, even if the total market demand has already been met.

Good catch — this is an important **market-clearing consistency fix**.

Below is a **refined implementation** that ensures:

> **Any bid with a price ≤ the clearing price is accepted**,
> **even if total demand has already been met**.

This corresponds to a **uniform-price market with full acceptance at the marginal price**, rather than stopping acceptance once demand is satisfied.

Key Conceptual Change

We split market clearing into **two phases**:

1. **Price discovery**
    - Sort bids by price
    - Accumulate quantities until demand is met
    - Identify the **clearing price** P^*
2. **Dispatch determination**
    - **Accept all bids with price ≤** P^*
    - Dispatch their **full bid quantities**
    - (Over-supply is allowed; no rationing is applied)

This exactly implements your requirement.

Refined `clear_market` Implementation

🔁 Replace the existing `clear_market` method with the following:

```python
def clear_market(self, demand, bids):
    """
    Merit-order clearing with full acceptance at clearing price.

    bids: list of dicts with keys {agent, quantity, price}

    Returns
    -------
    clearing_price : float
    dispatch : dict {agent: dispatched_quantity}
    """
```

```python
    # Sort bids by ascending price
    bids_sorted = sorted(bids, key=lambda x: x["price"])

    # ------------------------------------------------
    # Phase 1: Determine clearing price
    # ------------------------------------------------
    cumulative_quantity = 0.0
    clearing_price = None

    for bid in bids_sorted:
        cumulative_quantity += bid["quantity"]
        if cumulative_quantity >= demand:
            clearing_price = bid["price"]
            break

    # If demand is never met, clearing price is highest bid
    if clearing_price is None:
        clearing_price = bids_sorted[-1]["price"]

    # ------------------------------------------------
    # Phase 2: Dispatch all bids priced ≤ clearing price
    # ------------------------------------------------
    dispatch = {}

    for bid in bids_sorted:
        if bid["price"] <= clearing_price:
            dispatch[bid["agent"]] = bid["quantity"]
        else:
            dispatch[bid["agent"]] = 0.0

    return clearing_price, dispatch
```

What This Fix Achieves

✔️ All bids with $p_i \leq P^*$ are accepted
✔️ Dispatch is **independent of remaining demand**
✔️ Uniform-price settlement is respected
✔️ Wind, solar, and coal are treated consistently
✔️ Compatible with imbalance penalties and oversupply

Implications for the Wind Producer

- If the wind bid price is ≤ clearing price:
  - **Entire bid quantity is dispatched**
  - **Penalty applies** if actual production is lower
- Over-commitment now carries **real risk**, as intended
- This strengthens the realism of the bidding trade-off

Model Consistency Check

This refinement preserves:

- ✅ single shared environment

- ✅ modular stakeholder perspectives
- ✅ exogenous treatment of other decisions
- ✅ unchanged objective functions

Only the **transition function** is refined — exactly where it should be.

If you'd like, the next natural steps would be:

- adding **pro-rata rationing** as an alternative market rule,
- introducing **balancing-market buy-back** instead of penalties,
- or validating the clearing logic with **edge-case tests**.