OpenAI

Experiments 3 - Market 3

In this problem, four energy producers participate in a day-ahead merit-order electricity market, including three conventional ones and a renewable one. These producers submit hourly bids that specify a delivery quantity of electricity and an offer price for each hour of the next day. The market operator uses these bids to determine the market-clearing price and dispatch schedule. If a bid is accepted, the producer is obligated to deliver the specified quantity of electricity at the clearing price. Otherwise, the producer earns zero profit for that hour.

However, renewable generation is deeply uncertain, as it depends on weather conditions that cannot be predicted. If the producer under-delivers relative to its bid (due to overestimation of production), or over-delivers (due to underestimation), penalties or imbalance costs would apply. These costs are imposed by the system operator to maintain grid stability and to ensure supply-demand balance in real time. Meanwhile, the clearing price is unknown to the producer in advance, so they need to compete in this market.

The renewable energy producer’s goal is to maximize its expected profit by strategically selecting the quantity of electricity to bid for each hour of the next day, accounting for the uncertainty in both renewable generation and clearing prices.

Market structure and timing:
1. The day-ahead market is cleared once per day (e.g., at noon) for 24 hourly intervals of the following day.
2. Each producer submits a bid (b_t, p_t) for each hour $t=1, ..., 24$, where:
2.1. b_t: the quantity of energy (in MWh) the producer is willing to supply.
2.2. p_t: the minimum acceptable price per MWh.
3. The market operator computes the market-clearing price c_t for each hour.
4. If $p_t \leq c_t$, the bid is accepted, and the producer is committed to delivering b_t MWh in hour t.
5. The total energy demand in the market is D_t, which follows a normal distribution with constant mean μ_D and variance ${\sigma_D}^2$.

Penalty structure:
1. If actual generation $P_t < b_t$, the shortfall $b_t - P_t$ incurs a penalty of q_u per MWh.
2. If actual generation $P_t \geq b_t$, the surplus $P_t - b_t$ incurs a curtailment penalty of q_o per MWh.

Conventional producers:
1. For each conventional producer i, its bid (b_{it}, p_{it}) for each hour t follows normal distributions with constant means and variances.

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

Excellent — you’ve given a clear description of a **stochastic, single-period decision problem** repeated across 24 hours, where the renewable producer strategically bids into a day-ahead market under two layers of uncertainty: market clearing price and actual generation.

Let's formalize it as a **stochastic optimization model**, including all key structural components (decision variables, states, uncertainties, transition functions, and objective).

---

1. Model Overview

We consider a **renewable producer** participating in a **day-ahead electricity market** for 24 hourly intervals.
The producer chooses bids (b_t, p_t) to maximize **expected profit** under **uncertain clearing prices** and **uncertain generation**.

---

2. Sets and Indices

- $t \in \mathcal{T} = \{1, 2, \ldots, 24\}$: hourly intervals of the next day.

---

3. Decision Variables

For each hour t:

- $b_t \in \mathbb{R}_+$: bid quantity (MWh).
- $p_t \in \mathbb{R}_+$: bid price ($/MWh).

These are **chosen ex ante** (before uncertainties are realized).

---

4. Random Variables (Uncertainties)

(a) Renewable Generation

- P_t: actual renewable generation (MWh) in hour t.
  - Distribution: $P_t \sim f_P(\cdot)$, e.g. normal or truncated normal.
  - Mean $\mu_P(t)$, variance $\sigma_P^2(t)$.

(b) Market Clearing Price

- C_t: market clearing price ($/MWh) in hour t.
  - Random variable determined by the market operator based on all bids and realized demand.
  - Distribution: $C_t \sim f_C(\cdot)$ (uncertain to the producer).

(c) Market Demand

- $D_t \sim \mathcal{N}(\mu_D, \sigma_D^2)$ (affects clearing price indirectly).

---

5. Acceptance Condition

Bid (b_t, p_t) is **accepted** if

$$p_t \le C_t.$$

Let indicator variable:

$$I_t = \begin{cases} 1, & \text{if } p_t \le C_t, \\ 0, & \text{otherwise.} \end{cases}$$

---

6. Penalty Structure (Given Acceptance)

If accepted ($I_t = 1$):

- **Delivered amount:** must deliver b_t MWh.
- **Revenue:** $C_t \cdot b_t$.
- **Penalty:**
  - Underproduction (shortfall): $(b_t - P_t)^+ q_u$
  - Overproduction (surplus): $(P_t - b_t)^+ q_o$

Total hourly profit:

$$\Pi_t = I_t \big[C_t b_t - q_u (b_t - P_t)^+ - q_o (P_t - b_t)^+\big].$$

If not accepted ($I_t = 0$): $\Pi_t = 0$.

---

7. Objective Function

Maximize **expected total profit** over the 24-hour horizon:

$$\max_{\{b_t, p_t\}_{t=1}^{24}} \mathbb{E}\left[\sum_{t=1}^{24} \Pi_t\right] = \sum_{t=1}^{24} \mathbb{E}\left[I_t \left(C_t b_t - q_u (b_t - P_t)^+ - q_o (P_t - b_t)^+\right)\right].$$

Equivalently, using probability notation:

$$\mathbb{E}[\Pi_t] = \Pr(p_t \le C_t) \cdot \mathbb{E}\big[C_t b_t - q_u (b_t - P_t)^+ - q_o (P_t - b_t)^+ \,\big|\, p_t \le C_t\big].$$

---

8. State Variables and Dynamics

Although each hour can be treated independently (since bids are hourly and the day-ahead market clears once), we can formalize this as a **finite-horizon stochastic dynamic program**:

State Variables

At the beginning of hour t:

- $S_t = (\mu_P(t), \sigma_P^2(t), \mu_D, \sigma_D^2, \text{forecast info up to } t)$.

Decision Variables

- $a_t = (b_t, p_t)$.

Uncertainties

- $\omega_t = (P_t, C_t)$.

Transition Function

- $S_{t+1} = f(S_t, a_t, \omega_t)$:
  typically updates forecasts for future hours, e.g. via weather prediction or demand update models. If hours are independent, S_{t+1} = deterministic known value (no state evolution).

Reward Function

$$r(S_t, a_t, \omega_t) = I_t \left[C_t b_t - q_u (b_t - P_t)^+ - q_o (P_t - b_t)^+ \right].$$

Value Function Recursion

$$V_t(S_t) = \max_{a_t} \mathbb{E}\left[r(S_t, a_t, \omega_t) + V_{t+1}(S_{t+1}) \mid S_t, a_t \right],$$

with terminal condition $V_{25}(\cdot) = 0$.

If each hour is independent:

$$V_t(S_t) = \max_{a_t} \mathbb{E}\left[r(S_t, a_t, \omega_t) \right].$$

9. Initial State

At bidding time (day-ahead):

- $S_1 = \{\text{forecasts for } P_t, D_t, \text{ and price distributions } f_C(\cdot) \text{ for all } t \in \mathcal{T}\}$.

This captures all available information when forming bids.

10. Summary of Key Components

Component	Symbol	Description
Decision variables	b_t, p_t	Bid quantity and price per hour
Random variables	P_t, C_t, D_t	Generation, market price, and demand
State variable	S_t	Information set or forecast state
Transition	$S_{t+1} = f(S_t, a_t, \omega_t)$	Update forecasts or remain static

Component	Symbol	Description
Reward	$r(S_t, a_t, \omega_t)$	Profit net of penalties
Objective	$\max \mathbb{E}[\sum_t r(S_t, a_t, \omega_t)]$	Expected total profit
Constraints	$b_t \geq 0,\ p_t \geq 0$	Physical and market constraints

11. Extensions (optional)

- **Correlated uncertainties:** P_t and C_t may be correlated due to weather–price linkage.
- **Inter-temporal constraints:** e.g., ramping limits or storage carryover could link hours.
- **Distributionally robust or risk-averse formulations:** replace expectation with CVaR or min–max operator.

Compact Mathematical Formulation

$$\begin{aligned} \max_{\{b_t, p_t\}_{t=1}^{24}} \quad & \sum_{t=1}^{24} \mathbb{E}\left[\mathbf{1}_{\{p_t \leq C_t\}} \left(C_t b_t - q_u (b_t - P_t)^+ - q_o (P_t - b_t)^+\right)\right] \\ \text{s.t.} \quad & b_t \geq 0,\ p_t \geq 0, \quad \forall t. \end{aligned}$$

Would you like me to **extend this into a stochastic dynamic programming (SDP) representation** with explicit Bellman recursion and transition distributions (e.g., weather forecast updating between hours)?

There are three areas where you can improve the specification: (1) the renewable energy generation is deeply uncertain, so its probability distribution cannot be determined; (2) the market-clearing price is not random but endogenous, determined by the equilibrium of bids; and (3) for each conventional producer i, its bid (b_{it}, p_{it}) for each hour t follows normal distributions with constant means and variances. Please refine your specification.

Excellent — those are crucial clarifications.
Let's now **refine the model** to explicitly account for:

1. **Deep uncertainty** in renewable generation (no known probability distribution).
2. **Endogenous market-clearing price**, determined by market equilibrium.
3. **Stochastic (but parametric) behavior of conventional producers**, whose bids follow normal distributions.

This leads us naturally toward a **hybrid stochastic–equilibrium–robust model**, combining **game-theoretic** and **decision-theoretic** components.

1. Model Structure Overview

We model a **day-ahead electricity market** with four producers:
three **conventional** producers $i \in \mathcal{I}_C = \{1, 2, 3\}$ and one **renewable** producer (the decision-maker).

The market operates hourly ($t = 1, \ldots, 24$).

---

2. Decision Variables

For the renewable producer:

- $b_t \geq 0$: bid quantity (MWh).
- $p_t \geq 0$: bid price ($/MWh).

These are chosen **ex ante** (before any uncertainties are realized).

---

3. Market Participants and Bid Structure

(a) Conventional Producers

Each conventional producer $i \in \mathcal{I}_C$ submits for each hour t:

$$(b_{it}, p_{it}),$$

where both are **random variables** reflecting uncertain bidding behavior:

$$b_{it} \sim \mathcal{N}(\mu_{b_i}, \sigma_{b_i}^2), \qquad p_{it} \sim \mathcal{N}(\mu_{p_i}, \sigma_{p_i}^2).$$

Assume independence across hours but possibly correlation across producers.

Thus, the **supply side** of the market depends on:

$$\mathcal{B}_t = \{(b_t, p_t), (b_{1t}, p_{1t}), (b_{2t}, p_{2t}), (b_{3t}, p_{3t})\}.$$

---

4. Market Clearing Mechanism (Endogenous Price)

(a) Market Demand

For each hour t:

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

a known probabilistic demand.

(b) Market-Clearing Condition

The market operator determines the **market-clearing price** C_t and accepted bids via:

$$\sum_{j \in \{r,1,2,3\}} b_{jt} \cdot \mathbf{1}_{\{p_{jt} \leq C_t\}} = D_t,$$

where r denotes the renewable producer.
This equilibrium implicitly defines C_t as a **function** of all bids and demand:

$$C_t = g_t(p_t, b_t; \{(p_{it}, b_{it})_{i \in \mathcal{I}_C}\}, D_t).$$

Hence, C_t is **endogenous**, not random exogenously—it arises from **market equilibrium** given all bids.

---

5. Renewable Generation under Deep Uncertainty

Let the actual renewable generation in hour t be P_t.

- $P_t \in \mathcal{P}_t$, where $\mathcal{P}_t$ is an **uncertainty set**, not a random variable with a known distribution.
- The uncertainty set may be derived from forecasts:

  $$\mathcal{P}_t = [\underline{P}_t, \overline{P}_t],$$

  or more generally, $\mathcal{P}_t = \{\tilde{P}_t : |\tilde{P}_t - \hat{P}_t| \leq \delta_t\}$, where $\hat{P}_t$ is a point forecast and δ_t quantifies uncertainty.

The renewable producer adopts a **robust** (or **distributionally robust**) approach, guarding against unfavorable realizations within $\mathcal{P}_t$.

---

6. Profit and Penalties

If the renewable producer's bid is **accepted** ($p_t \leq C_t$):

- **Revenue:** $C_t \cdot b_t$
- **Penalty (imbalances):**
  - Underproduction: $q_u(b_t - P_t)^+$
  - Overproduction: $q_o(P_t - b_t)^+$

Profit realization:

$$\Pi_t(b_t, p_t, P_t, C_t) = \mathbf{1}_{\{p_t \leq C_t\}} \left[C_t b_t - q_u(b_t - P_t)^+ - q_o(P_t - b_t)^+ \right].$$

---

7. Renewable Producer's Optimization Problem

Since generation is deeply uncertain and market prices are endogenous (driven by stochastic rival bids and demand), the renewable's problem is a **hybrid robust–stochastic optimization** problem:

$$\max_{\{b_t, p_t\}_{t=1}^{24}} \mathbb{E}_{(b_{it}, p_{it}), D_t} \left[\min_{P_t \in \mathcal{P}_t} \sum_{t=1}^{24} \Pi_t(b_t, p_t, P_t, C_t(b_t, p_t, \{b_{it}, p_{it}\}, D_t)) \right].$$

That is:

- **Expectation** over stochastic competitor bids and demand.

- **Minimization** over generation uncertainty (worst-case realization).

---

8. State Variables and Transitions

(a) State Variables

At the start of each hour t:

$$S_t = (\hat{P}_t, \mathcal{P}_t, \mu_D, \sigma_D^2, \{\mu_{b_i}, \sigma_{b_i}, \mu_{p_i}, \sigma_{p_i}\}_{i=1}^3),$$

representing:

- Forecast of renewable generation and its uncertainty set.
- Demand parameters.
- Competitor bid parameters.

(b) Decision Variables

$$a_t = (b_t, p_t).$$

(c) Uncertainties

- Conventional bids: $\omega_{Ct} = \{(b_{it}, p_{it})_{i \in \mathcal{I}_C}\}$
- Demand: D_t
- Generation: $P_t \in \mathcal{P}_t$

(d) Transition Function

$$S_{t+1} = f(S_t, a_t, \omega_{Ct}, D_t),$$

e.g. updating forecasts for later hours based on weather and demand signals.
If hours are independent, then $S_{t+1} = S_t$.

---

9. Value Function (Dynamic Form)

The dynamic programming representation:

$$V_t(S_t) = \max_{a_t=(b_t,p_t)} \mathbb{E}_{\omega_{Ct},D_t} \left[\min_{P_t \in \mathcal{P}_t} \{r(S_t, a_t, \omega_{Ct}, D_t, P_t) + V_{t+1}(S_{t+1})\} \right],$$

where the **instantaneous reward** is:

$$r(S_t, a_t, \omega_{Ct}, D_t, P_t) = \mathbf{1}_{\{p_t \le C_t(b_t, p_t, \omega_{Ct}, D_t)\}} \left[C_t b_t - q_u(b_t - P_t)^+ - q_o(P_t - b_t)^+ \right].$$

Terminal condition: $V_{25}(S_{25}) = 0$.

If hours are independent, then:

$$V_t(S_t) = \max_{a_t} \mathbb{E}_{\omega_{Ct}, D_t} \left[\min_{P_t \in \mathcal{P}_t} r(S_t, a_t, \omega_{Ct}, D_t, P_t) \right].$$

10. Interpretation of the Three Improvements

Issue	Previous	Refined Treatment
1. Renewable generation "deep uncertainty"	Treated as random variable with known distribution	Treated as **uncertain within a set** (robust or distributionally robust). The producer maximizes expected profit under worst-case generation.
2. Market-clearing price endogenous	Exogenous random variable C_t	Endogenously determined by equilibrium condition $C_t = g_t(p_t, b_t, \{b_{it}, p_{it}\}, D_t)$.
3. Conventional producers' bids stochastic	Not explicitly modeled	Each (b_{it}, p_{it}) follows normal distributions with given parameters, introducing stochasticity in market equilibrium and thus in C_t.

11. Final Compact Formulation

$$\boxed{\begin{aligned} \max_{\{b_t, p_t\}_{t=1}^{24}} \quad & \mathbb{E}_{(b_{it}, p_{it}), D_t} \left[\min_{P_t \in \mathcal{P}_t} \sum_{t=1}^{24} \mathbf{1}_{\{p_t \le C_t(b_t, p_t, \{b_{it}, p_{it}\}, D_t)\}} \left(C_t b_t - q_u (b_t - P_t)^+ - q_o (P_t - b_t)^+ \right) \right] \\ \text{s.t.} \quad & C_t = g_t(p_t, b_t, \{b_{it}, p_{it}\}, D_t), \quad \forall t, \\ & b_t \ge 0, \; p_t \ge 0, \quad \forall t. \end{aligned}}$$

Would you like me to take the next step and **derive the equilibrium-clearing function** $g_t(\cdot)$ explicitly (e.g., under merit-order sorting of bids) to show how C_t is computed from the bid distributions?

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

Excellent — this is now a **multi-agent stochastic–robust equilibrium system**, where:

- The **renewable producer** acts strategically to maximize its own expected (robust) profit under uncertainty and market rules.
- The **system regulator (market operator)** designs and enforces rules — clearing mechanisms, penalties, and reliability constraints — to ensure efficient, stable, and fair operation of the market.

They share a **common environment**: the physical power system and the day-ahead market, where their actions **mutually influence** one another through the market-clearing mechanism, prices, and system balance.

Let's carefully formalize both perspectives within one coherent shared structure.

---

1. Common Environment (Shared System)

Sets

- $t \in \mathcal{T} = \{1, 2, \ldots, 24\}$: hourly intervals of the next day.
- $\mathcal{I}_C = \{1, 2, 3\}$: set of conventional producers.
- r: renewable producer.

Global Exogenous Parameters

Symbol	Meaning
μ_D, σ_D^2	Mean and variance of hourly demand.
q_u, q_o	Penalty rates for under- and over-delivery (set by regulator).
μ_{b_i}, σ_{b_i}	Mean and variance of conventional producer i's bid quantity.
μ_{p_i}, σ_{p_i}	Mean and variance of conventional producer i's bid price.
$\hat{P}_t$	Forecast renewable generation for hour t.
$\mathcal{P}_t = [\underline{P}_t, \overline{P}_t]$	Deep uncertainty set for renewable generation.

Global Random Variables

Variable	Description	Source
$D_t \sim \mathcal{N}(\mu_D, \sigma_D^2)$	Total system demand	External
$(b_{it}, p_{it}) \sim \mathcal{N}((\mu_{b_i}, \mu_{p_i}), \mathrm{diag}(\sigma_{b_i}^2, \sigma_{p_i}^2))$	Conventional producers' bids	Other producers
$P_t \in \mathcal{P}_t$	Actual renewable generation	Deep uncertainty

---

Endogenous Market Variables

These are determined jointly by the regulator's clearing mechanism and producers' bids.

Variable	Description
C_t	Market-clearing price
$x_{jt} \in \{0, 1\}$	Acceptance indicator for each producer $j \in \{r\} \cup \mathcal{I}_C$
$\sum_j b_{jt} x_{jt} = D_t$	Market-clearing condition (supply = demand)

Thus, C_t and x_{jt} are **shared global variables** linking the stakeholders.

---

2. Perspective 1: Renewable Producer

State Variables

At the time of bidding (before day-ahead market clears):

$$S_t^r = (\hat{P}_t, \mathcal{P}_t, \mu_D, \sigma_D^2, \{\mu_{b_i}, \sigma_{b_i}, \mu_{p_i}, \sigma_{p_i}\}_{i \in \mathcal{I}_C}).$$

This state encodes the renewable's knowledge of forecasts, demand statistics, and competitor behavior distributions.

Control Variables

$$a_t^r = (b_t, p_t),$$

where:

- b_t: bid quantity (MWh).
- p_t: bid price ($/MWh).

Uncertainties

- Competitor bids $(b_{it}, p_{it})_{i \in \mathcal{I}_C}$,
- Demand D_t,
- Realized renewable output $P_t \in \mathcal{P}_t$.

Transition Function

If we model intertemporal dependencies (e.g., updated weather forecasts):

$$S_{t+1}^r = f^r(S_t^r, a_t^r, D_t, P_t),$$

otherwise $S_{t+1}^r = S_t^r$ if hours are independent.

Market Interaction

The renewable's bid affects the **clearing price** through the market equilibrium function:

$$C_t = g_t(a_t^r, \{a_t^C\}, D_t),$$

where $\{a_t^C\}$ are stochastic conventional bids.

Profit Function

If the bid is accepted ($p_t \leq C_t$):

$$\pi_t^r = C_t b_t - q_u(b_t - P_t)^+ - q_o(P_t - b_t)^+.$$

If not accepted, $\pi_t^r = 0$.

Objectives of the Renewable Producer

The renewable producer faces *deep uncertainty* in P_t but *stochastic* uncertainty in D_t and competitors' bids.
Hence, its objectives can be expressed separately as:

1. **Expected Profit Maximization (over market uncertainty):**

$$J_1^r = \mathbb{E}_{(b_{it}, p_{it}), D_t}\left[\sum_{t=1}^{24} \mathbf{1}_{\{p_t \le C_t\}} C_t b_t\right].$$

2. **Worst-Case Penalty Minimization (over generation uncertainty):**

$$J_2^r = \min_{P_t \in \mathcal{P}_t} \sum_{t=1}^{24} \mathbf{1}_{\{p_t \le C_t\}}\left[q_u(b_t - P_t)^+ + q_o(P_t - b_t)^+\right].$$

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

$$\boxed{\begin{aligned} \max_{\{b_t, p_t\}_{t=1}^{24}} \quad & (J_1^r, -J_2^r) \\ \text{s.t. } & C_t = g_t(a_t^r, \{a_t^C\}, D_t), \\ & b_t, p_t \ge 0. \end{aligned}}$$

---

3. Perspective 2: System Regulator (Market Operator)

The regulator is a **centralized actor** managing the same shared environment, focusing on system-level performance, efficiency, and reliability.
It does not control individual bids, but it **sets rules and parameters** that influence agents' behavior.

State Variables

$$S_t^{\text{reg}} = (\{a_t^r, a_{it}\}_{i \in \mathcal{I}_C}, D_t, P_t).$$

This is the realized market environment for hour t.

Control Variables

The regulator's policy levers (control variables) include:

- q_u, q_o: penalty coefficients for imbalances.

- Market-clearing mechanism $g_t(\cdot)$: defines how bids map to prices and dispatch (e.g., merit-order, uniform pricing).
- Possibly, market participation rules (e.g., caps on bids, reserve requirements).

We denote:

$$a_t^{\text{reg}} = (q_u, q_o, g_t(\cdot)).$$

Transition Function

$$S_{t+1}^{\text{reg}} = f^{\text{reg}}(S_t^{\text{reg}}, a_t^{\text{reg}}, \text{system responses}),$$

where responses depend on how producers adjust bids in reaction to market rules.

Objectives of the Regulator

The regulator has **multiple system-level objectives** reflecting social welfare, reliability, and fairness:

1. **Market Efficiency (social welfare maximization):**

$$J_1^{\text{reg}} = \mathbb{E}_{D_t}\left[\sum_{t=1}^{24} \left(U_t(D_t) - C_t D_t\right)\right],$$

   where $U_t(D_t)$ is the consumer utility of demand D_t.

2. **System Reliability (penalty or deviation minimization):**

$$J_2^{\text{reg}} = -\mathbb{E}\left[\sum_{t=1}^{24} |D_t - \sum_j P_{jt}^{\text{delivered}}|\right],$$

   encouraging stable supply–demand balance.

3. **Renewable Integration and Fairness:**

$$J_3^{\text{reg}} = -\operatorname{Var}_t[\text{accepted renewable share}],$$

   promoting predictable renewable participation.

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

$$\boxed{\begin{aligned} \max_{a_t^{\text{reg}}} \quad & (J_1^{\text{reg}}, J_2^{\text{reg}}, J_3^{\text{reg}}) \\ \text{s.t.} \quad & C_t, x_{jt} = \arg\max_{C_t, x_{jt}} \left\{ \sum_j p_{jt} x_{jt} \mid \sum_j b_{jt} x_{jt} = D_t \right\}, \\ & a_t^r, a_t^C \text{ follow equilibrium responses to } a_t^{\text{reg}}. \end{aligned}}$$

---

4. Shared Equilibrium and Interaction

- The **renewable producer** chooses bids (b_t, p_t) optimizing its private objectives $(J_1^r, -J_2^r)$, taking the regulator's market rules and penalties as given.
- The **regulator** sets market rules (q_u, q_o, g_t) optimizing its social objectives $(J_1^{\text{reg}}, J_2^{\text{reg}}, J_3^{\text{reg}})$, anticipating producers' equilibrium responses.
- Both interact via the **shared environment** (market-clearing price C_t, accepted bids, realized generation).

Mathematically, the system forms a **Stackelberg game** (leader–follower structure):

- Regulator = leader.
- Producers (including the renewable) = followers.
- Market clearing acts as the coupling equilibrium constraint.

5. Summary Table of Components

Aspect	Common Environment	Renewable Producer	Regulator
State	Forecasts, distributions, demand, bids, uncertainty sets	S_t^r (forecasts + competitor stats)	S_t^{reg} (bids, demand, generation)
Controls	—	$a_t^r = (b_t, p_t)$	$a_t^{\text{reg}} = (q_u, q_o, g_t)$
Uncertainty	$D_t, (b_{it}, p_{it}), P_t \in \mathcal{P}_t$	Competitor bids, demand, deep uncertainty in generation	Producer responses, demand
Transition	System evolves via $S_{t+1} = f(S_t, a_t^r, a_t^{\text{reg}})$	$f^r(S_t^r, \cdot)$	$f^{\text{reg}}(S_t^{\text{reg}}, \cdot)$
Objectives	—	J_1^r (expected profit), J_2^r (worst-case penalty)	J_1^{reg} (efficiency), J_2^{reg} (reliability), J_3^{reg} (renewable fairness)
Interaction	Market-clearing price C_t, acceptance x_{jt}	Follows market rule $g_t(\cdot)$	Defines and enforces $g_t(\cdot)$

Interpretation

- The **shared environment** ensures consistent physical and market constraints.
- Each stakeholder acts **independently but interdependently**, influencing equilibrium outcomes indirectly.
- The model captures both **strategic bidding** (renewable's viewpoint) and **policy design** (regulator's viewpoint) within a unified stochastic–robust–game-theoretic framework.

Would you like me to extend this into an **explicit bilevel mathematical program** (regulator's upper-level policy design and renewable's lower-level optimal bidding), with full equilibrium conditions for the market-clearing price C_t?

Your specification should only include implementable and quantifiable components. For those conceptual components, please refine them or remove them. Furthermore, the market-clearing mechanism follows merit-order as described in the problem description.

Perfect — this is an excellent clarification.
We'll now **refine the model specification** so that all components are **implementable, measurable, and directly quantifiable**, while removing or grounding any conceptual or abstract elements.

We'll retain the two perspectives:

1. **Renewable producer**, who strategically decides bids.
2. **System regulator**, who operates the **merit-order market** and sets observable system parameters (e.g., penalty rates).

They share a **common environment** containing all exogenous inputs and endogenous market variables that link their actions.

---

1. Common Environment (Shared System)

The common environment describes the data, exogenous uncertainties, and market rules observed by both stakeholders.

1.1 Sets and Indices

- $t \in \mathcal{T} = \{1, 2, \ldots, 24\}$: hourly intervals for the next day.
- $\mathcal{I}_C = \{1, 2, 3\}$: conventional producers.
- r: renewable producer.

1.2 Global Exogenous Parameters

Symbol	Description	Units / Type
μ_D, σ_D	Mean and standard deviation of hourly demand	MWh
μ_{b_i}, σ_{b_i}	Mean and std. dev. of bid quantity for conventional producer i	MWh
μ_{p_i}, σ_{p_i}	Mean and std. dev. of bid price for conventional producer i	\$/MWh
$\hat{P}_t$	Point forecast of renewable generation	MWh
$\underline{P}_t, \overline{P}_t$	Lower and upper bounds of renewable generation uncertainty set	MWh
q_u, q_o	Penalty rates for under- and over-delivery	\$/MWh

All parameters are numerical and measurable (historical data or forecasts).

1.3 Random Variables

Symbol	Distribution	Description
$D_t \sim \mathcal{N}(\mu_D, \sigma_D^2)$	Hourly system demand	

Symbol	Distribution	Description
$b_{it} \sim \mathcal{N}(\mu_{b_i}, \sigma_{b_i}^2)$	Conventional bid quantity	
$p_{it} \sim \mathcal{N}(\mu_{p_i}, \sigma_{p_i}^2)$	Conventional bid price	
$P_t \in [\underline{P}_t, \overline{P}_t]$	Renewable generation (uncertain but bounded)	

All quantities are observable or simulatable.

1.4 Market-Clearing Mechanism (Merit-Order Rule)

For each hour t:

1. The market operator collects bids (b_{jt}, p_{jt}) for all producers $j \in \mathcal{I}_C \cup \{r\}$.
2. Sort bids by ascending price p_{jt}.
3. Accept bids sequentially until total accepted quantity equals demand D_t.
4. The **market-clearing price** C_t equals the marginal accepted bid price:

$$C_t = \min\{p_{jt} : \sum_{k:p_{kt} \le p_{jt}} b_{kt} \ge D_t\}.$$

5. Each accepted producer j is dispatched for its bid quantity b_{jt} and paid C_t per MWh.

Thus, C_t and the acceptance indicator x_{jt} are **endogenous** but computable.

$$x_{jt} = \begin{cases} 1, & \text{if } p_{jt} \le C_t, \\ 0, & \text{otherwise.} \end{cases}$$

1.5 Shared Variables (Endogenous)

Symbol	Description	Derived from
C_t	Market-clearing price	Merit-order algorithm
x_{jt}	Acceptance indicator (0/1)	Merit-order algorithm
Π_t^r	Renewable producer profit	Renewable model
Reliability_t	Realized imbalance magnitude	Regulator model

These link the two perspectives.

2. Renewable Producer's Model

2.1 State Variables

At bidding time (before market clearing):

$$S_t^r = (\hat{P}_t, \underline{P}_t, \overline{P}_t, \mu_D, \sigma_D, \{\mu_{b_i}, \sigma_{b_i}, \mu_{p_i}, \sigma_{p_i}\}_{i=1}^3).$$

All components are **quantifiable forecasts or statistics** available to the renewable producer.

2.2 Control Variables

$$a_t^r = (b_t, p_t),$$

the bid quantity and price for hour t.

2.3 Random Variables Observed After Bidding

- Demand D_t
- Competitor bids (b_{it}, p_{it})
- Actual renewable generation $P_t \in [\underline{P}_t, \overline{P}_t]$

2.4 Market Outcome

Given all bids and demand, the market-clearing price C_t is computed via the merit-order rule.
Acceptance indicator for renewable producer:

$$x_t^r = \mathbf{1}_{\{p_t \le C_t\}}.$$

2.5 Profit Calculation

If accepted:

$$\Pi_t^r = C_t b_t - q_u (b_t - P_t)^+ - q_o (P_t - b_t)^+.$$

Otherwise $\Pi_t^r = 0$.

2.6 Objective Functions

The renewable producer faces two measurable objectives:

1. **Expected Profit Maximization (over stochastic market outcomes):**

$$J_1^r = \mathbb{E}_{D_t,(b_{it},p_{it})} \left[\sum_{t=1}^{24} x_t^r C_t b_t \right].$$

2. **Worst-Case Penalty Minimization (over bounded generation uncertainty):**

$$J_2^r = - \max_{P_t \in [\underline{P}_t, \overline{P}_t]} \sum_{t=1}^{24} x_t^r \left[q_u (b_t - P_t)^+ + q_o (P_t - b_t)^+ \right].$$

2.7 Transition Function

If each hour is independent (no carry-over constraints):

$$S_{t+1}^r = S_t^r.$$

Otherwise, forecasts for P_{t+1} can be updated based on observed P_t:

$$S_{t+1}^r = f^r(S_t^r, P_t).$$

This can be quantified by a forecast-updating model.

2.8 Renewable Producer Model Summary

$$\boxed{\begin{aligned} \max_{\{b_t,p_t\}_{t=1}^{24}} \quad & (J_1^r, J_2^r) \\ \text{s.t.} \quad & C_t = \text{MeritOrder}(\{(b_t,p_t),(b_{it},p_{it})_{i\in\mathcal{I}_C}\}, D_t), \\ & P_t \in [\underline{P}_t, \overline{P}_t], \\ & b_t, p_t \geq 0. \end{aligned}}$$

All components are computable from data or simulations.

---

3. System Regulator's Model

3.1 State Variables

$$S_t^{\text{reg}} = (\{(b_{it},p_{it})_{i\in\mathcal{I}_C}\},(b_t,p_t),D_t,P_t).$$

Everything in S_t^{reg} is observable to the system operator after bids are submitted.

3.2 Control Variables

The regulator sets **market parameters** and applies the **merit-order clearing rule**:

$$a_t^{\text{reg}} = (q_u, q_o),$$

since the merit-order algorithm itself is fixed by market design.

3.3 Transition Function

For each hour t:

$$S_{t+1}^{\text{reg}} = S_t^{\text{reg}},$$

(hours are cleared independently).

3.4 Quantifiable System Metrics

After clearing and dispatch:

- **System imbalance (MWh):**

$$\Delta_t = D_t - \sum_j P_{jt}^{\text{delivered}}.$$

- **Renewable participation share:**

$$\rho_t = \frac{x_t^r b_t}{D_t}.$$

3.5 Objective Functions

The regulator has measurable performance goals:

1. **Price Stability:**

$$J_1^{\text{reg}} = -\operatorname{Var}(C_t), \quad t = 1, \ldots, 24.$$

2. **System Reliability (imbalance minimization):**

$$J_2^{\text{reg}} = -\mathbb{E}\left[\sum_{t=1}^{24} |\Delta_t|\right].$$

3. **Renewable Integration (share target):**

$$J_3^{\text{reg}} = -\sum_{t=1}^{24} |\rho_t - \rho^*|,$$

where ρ^* is a policy target fraction (e.g., 30%).

Each of these metrics is measurable from market data.

3.6 Regulator Model Summary

$$\boxed{\begin{aligned} \max_{q_u, q_o} \quad & (J_1^{\text{reg}}, J_2^{\text{reg}}, J_3^{\text{reg}}) \\ \text{s.t.} \quad & C_t, x_{jt} = \text{MeritOrder}(\{(b_{jt}, p_{jt})\}, D_t), \\ & \Delta_t = D_t - \sum_j P_{jt}^{\text{delivered}}, \\ & \rho_t = \tfrac{x_t^r b_t}{D_t}, \\ & q_u, q_o \geq 0. \end{aligned}}$$

---

4. Interaction and Coupling

Element	Controlled by	Observed by	Quantifiable effect
q_u, q_o	Regulator	Renewable (inputs)	Affects penalties and bid strategy
b_t, p_t	Renewable	Regulator (as bid data)	Influences market price C_t
C_t	Shared outcome	Both	Observable clearing price

Element	Controlled by	Observed by	Quantifiable effect
Π_t^r, Δ_t	Computed from market outcome	Both	Quantitative performance metrics

Both stakeholders interact **indirectly through the market-clearing process** but optimize distinct, measurable objectives.

---

5. Overall Model Summary

Shared Market Environment

$$\mathrm{MeritOrder}(\{(b_{jt}, p_{jt})\}, D_t) \Rightarrow (C_t, x_{jt}), \quad t = 1, \dots, 24.$$

Renewable Producer

$$\begin{aligned}
&\max_{b_t, p_t} \quad (J_1^r, J_2^r) \\
&J_1^r = \mathbb{E}_{D_t, (b_{it}, p_{it})}[x_t^r C_t b_t], \\
&J_2^r = -\max_{P_t \in [\underline{P}_t, \overline{P}_t]} x_t^r [q_u(b_t - P_t)^+ + q_o(P_t - b_t)^+].
\end{aligned}$$

System Regulator

$$\begin{aligned}
&\max_{q_u, q_o} \quad (J_1^{\mathrm{reg}}, J_2^{\mathrm{reg}}, J_3^{\mathrm{reg}}) \\
&J_1^{\mathrm{reg}} = -\mathrm{Var}(C_t), \\
&J_2^{\mathrm{reg}} = -\mathbb{E}[|\Delta_t|], \\
&J_3^{\mathrm{reg}} = -\sum_t |\rho_t - \rho^*|.
\end{aligned}$$

All variables and objectives are **implementable**, **quantifiable**, and based on measurable quantities (bids, prices, quantities, demand, and generation).
The two models operate within the **same market environment** and influence one another **only through data-driven, computable quantities**.

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

Excellent — now we will **compose** the renewable producer's model and the system regulator's model into a **single unified, implementable, and internally consistent framework**.

This **composed model** maintains:

1. A **shared environment** (states, exogenous parameters, and random variables).

2. **Perspective-specific control and objective modules** (renewable vs. regulator).
3. A consistent interaction mechanism (merit-order market).
4. The ability to **evaluate each perspective independently** by treating the other's controls as fixed inputs.

All elements are **quantifiable and implementable**.

Unified Model Specification

---

1. Common Environment

1.1 Time and Agents

- Hours: $t \in \mathcal{T} = \{1, 2, \ldots, 24\}$
- Producers:
  - Renewable producer r
  - Conventional producers $\mathcal{I}_C = \{1, 2, 3\}$
- Regulator / Market Operator: M

1.2 Global Parameters

Symbol	Description	Units
μ_D, σ_D	Mean and std. dev. of hourly demand	MWh
μ_{b_i}, σ_{b_i}	Mean and std. dev. of bid quantity for conventional producer i	MWh
μ_{p_i}, σ_{p_i}	Mean and std. dev. of bid price for conventional producer i	\$/MWh
$\hat{P}_t$	Point forecast of renewable generation	MWh
$\underline{P}_t, \overline{P}_t$	Lower and upper bounds of renewable generation	MWh
q_u, q_o	Penalty rates for imbalance (set by regulator)	\$/MWh
ρ^*	Target renewable participation fraction	—

All quantities are measurable or set by policy.

1.3 Random Variables

Symbol	Distribution	Description
$D_t \sim \mathcal{N}(\mu_D, \sigma_D^2)$	System demand	
$b_{it} \sim \mathcal{N}(\mu_{b_i}, \sigma_{b_i}^2)$	Conventional bid quantity	
$p_{it} \sim \mathcal{N}(\mu_{p_i}, \sigma_{p_i}^2)$	Conventional bid price	
$P_t \in [\underline{P}_t, \overline{P}_t]$	Renewable generation (bounded uncertainty)	

1.4 Shared State Vector

At hour t:

$$S_t = (\hat{P}_t, \underline{P}_t, \overline{P}_t, D_t, \{b_{it}, p_{it}\}_{i \in \mathcal{I}_C}, q_u, q_o).$$

All stakeholders observe or forecast S_t when making decisions.

1.5 Market Clearing Mechanism (Merit-Order)

Given all submitted bids (b_{jt}, p_{jt}) from producers $j \in \mathcal{I}_C \cup \{r\}$ and realized demand D_t:

1. Sort all bids in **ascending order of** p_{jt}.
2. Accept bids sequentially until total accepted supply meets or exceeds D_t.
3. Set **clearing price**:

$$C_t = \min\{p_{jt} : \sum_{k: p_{kt} \le p_{jt}} b_{kt} \ge D_t\}.$$

4. Acceptance indicator:

$$x_{jt} = \begin{cases} 1, & p_{jt} \le C_t, \\ 0, & \text{otherwise.} \end{cases}$$

5. Each accepted producer is dispatched for b_{jt} and paid C_t per MWh.

Thus, $C_t = G_t(\{b_{jt}, p_{jt}\}, D_t)$ and $x_{jt} = X_t(\{b_{jt}, p_{jt}\}, D_t)$.

These mappings are fully computable.

---

2. Perspective Modules

Each stakeholder is embedded in the shared environment but has its own controls, objectives, and evaluation criteria.

2.1 Renewable Producer Module

(a) Inputs

- State S_t (forecasts, competitor bid distributions, demand parameters, penalties q_u, q_o).
- Exogenous variables $(b_{it}, p_{it}), D_t$.
- Controls of other perspectives (here, q_u, q_o) treated as fixed inputs.

(b) Control Variables

$$a_t^r = (b_t, p_t), \quad b_t \ge 0,\, p_t \ge 0.$$

(c) Market Outcome

Given all bids, compute:

$$C_t = G_t((b_t, p_t), \{b_{it}, p_{it}\}, D_t), \quad x_t^r = X_t((b_t, p_t), \{b_{it}, p_{it}\}, D_t).$$

(d) Profit Realization

If accepted:

$$\Pi_t^r = C_t b_t - q_u(b_t - P_t)^+ - q_o(P_t - b_t)^+.$$

Otherwise, $\Pi_t^r = 0$.

(e) Objectives

Two measurable objectives:

1. **Expected Market Profit:**

$$J_1^r = \mathbb{E}_{D_t,(b_{it},p_{it})}\left[\sum_{t=1}^{24} x_t^r \, C_t b_t\right].$$

2. **Worst-Case Penalty Minimization:**

$$J_2^r = -\max_{P_t \in [\underline{P}_t, \overline{P}_t]} \sum_{t=1}^{24} x_t^r \left[q_u(b_t - P_t)^+ + q_o(P_t - b_t)^+\right].$$

(f) Transition

If hours are independent:

$$S_{t+1} = S_t.$$

If forecasts update with observations, use:

$$S_{t+1} = f^r(S_t, P_t).$$

2.2 Regulator Module

(a) Inputs

- State S_t (bids, demand, generation, etc.).
- Renewable producer's bids (b_t, p_t) and conventional bids (b_{it}, p_{it}) are treated as exogenous inputs.

(b) Control Variables

$$a_t^M = (q_u, q_o), \quad q_u, q_o \geq 0.$$

(The merit-order clearing mechanism G_t, X_t is fixed and non-decision-based.)

(c) Market Outcomes

From the same merit-order rule:

$$C_t = G_t(\{b_{jt}, p_{jt}\}, D_t), \quad x_{jt} = X_t(\{b_{jt}, p_{jt}\}, D_t).$$

(d) Derived Quantities

- **Supply-demand imbalance:**

$$\Delta_t = D_t - \sum_j P_{jt}^{\text{delivered}},$$

  where $P_{jt}^{\text{delivered}} = \min(b_{jt}, P_t)$ for accepted producers.

- **Renewable participation fraction:**

$$\rho_t = \frac{x_t^r b_t}{D_t}.$$

All quantities are measurable after market clearing.

(e) Objectives**

1. **Price Stability:**

$$J_1^M = -\operatorname{Var}(C_t).$$

2. **System Reliability:**

$$J_2^M = -\mathbb{E}\left[\sum_{t=1}^{24} |\Delta_t|\right].$$

3. **Renewable Integration:**

$$J_3^M = -\sum_{t=1}^{24} |\rho_t - \rho^*|.$$

(f) Transition**

$$S_{t+1} = S_t,$$

since hourly clearings are independent.

---

3. Unified System Dynamics

All agents evolve within the same state S_t and market-clearing rule.

At each hour t:

1. **Bidding stage:**
    - Renewable selects $a_t^r = (b_t, p_t)$.
    - Conventional producers submit $(b_{it}, p_{it}) \sim \mathcal{N}(\mu_{b_i}, \sigma_{b_i}^2) \times \mathcal{N}(\mu_{p_i}, \sigma_{p_i}^2)$.
    - Regulator's penalty rates $a_t^M = (q_u, q_o)$ are in force.
2. **Market clearing (shared computation):**

$$(C_t, \{x_{jt}\}) = \mathrm{MeritOrder}(\{(b_{jt}, p_{jt})\}, D_t).$$

3. **Settlement stage:**
    - Renewable realizes profit Π_t^r.
    - Regulator observes market outcomes and computes reliability and integration metrics.
4. **Update stage (optional):**

$$S_{t+1} = f(S_t, P_t, D_t),$$

to reflect new forecasts or statistics.

---

4. Unified Objective Structure

Renewable Producer Objective Module

$$\max_{(b_t, p_t)} (J_1^r, J_2^r) \quad \text{s.t. shared market and state dynamics.}$$

Regulator Objective Module

$$\max_{(q_u, q_o)} (J_1^M, J_2^M, J_3^M) \quad \text{s.t. shared market and state dynamics.}$$

Each module treats the other's controls as **given inputs**.

---

5. Unified Mathematical Representation

$$\textbf{Common Environment: } S_t,\ (C_t, x_{jt}) = \mathrm{MeritOrder}(\{(b_{jt}, p_{jt})\}, D_t),\ D_t \sim \mathcal{N}(\mu_D, \sigma_D^2).$$

$$\textbf{Renewable Control: } a_t^r = (b_t, p_t), \quad \text{given } (q_u, q_o).$$

$$\textbf{Regulator Control: } a_t^M = (q_u, q_o), \quad \text{given } (b_t, p_t).$$

$$\textbf{Renewable Objectives: } \begin{cases} J_1^r = \mathbb{E}_{D_t,(b_{it},p_{it})}[x_t^r C_t b_t], \\ J_2^r = -\max_{P_t \in [\underline{P}_t, \overline{P}_t]} x_t^r [q_u(b_t - P_t)^+ + q_o(P_t - b_t)^+]. \end{cases}$$

$$\textbf{Regulator Objectives: } \begin{cases} J_1^M = -\mathrm{Var}(C_t), \\ J_2^M = -\mathbb{E}[|\Delta_t|], \\ J_3^M = -\sum_t |\rho_t - \rho^*|. \end{cases}$$

$$\textbf{Transitions: } S_{t+1} = f(S_t, P_t, D_t) \text{ or } S_{t+1} = S_t.$$

6. Modularity and Evaluation

- The **shared state** S_t and **market-clearing mechanism** ensure internal consistency.
- Each **perspective module** (renewable, regulator) can be simulated or optimized separately:
  - To analyze renewable bidding strategy: fix $q_u, q_o \rightarrow$ optimize b_t, p_t.
  - To analyze regulation policy: fix $b_t, p_t \rightarrow$ optimize q_u, q_o.
- The shared structure guarantees that outcomes (prices, profits, imbalances) are computed consistently across both evaluations.

✅ Summary of the Unified Model

Layer	Component	Description	Shared/Private
Environment	S_t	Forecasts, bids, demand, penalties	Shared
Mechanism	Merit-order (G_t, X_t)	Computes C_t, x_{jt}	Shared
Renewable Control	$a_t^r = (b_t, p_t)$	Hourly bid decision	Private (Renewable)
Regulator Control	$a_t^M = (q_u, q_o)$	Penalty parameters	Private (Regulator)
Renewable Objectives	J_1^r, J_2^r	Profit and penalty risk	Private
Regulator Objectives	J_1^M, J_2^M, J_3^M	Stability, reliability, renewables	Private
Coupling Variables	$C_t, x_{jt}, \Pi_t^r, \Delta_t, \rho_t$	Determined jointly	Shared

Interpretation

This unified model:

- Is **internally consistent** — all variables exist once, not per perspective.
- Contains **only implementable components** — every parameter, state, and function is computable or measurable.

- Allows **modular analysis** — each perspective can be optimized or simulated separately by holding the other's controls fixed.
- Embeds both perspectives' objectives in a single, consistent market environment governed by the merit-order mechanism.

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

Below is a **modular Python implementation** of the unified, implementable model.
It uses **object-oriented design** to represent each stakeholder's perspective (Renewable Producer and Regulator), sharing a **common environment** and **market-clearing mechanism**.

The code is organized for **simulation and evaluation**, not optimization.
At the end, a function `simulate_market()` serves as the unified interface.
It takes two dictionaries:

- `controls` → control variables (`b_t, p_t` for renewable; `q_u, q_o` for regulator)
- `uncertainty` → uncertain inputs (`D_t`, conventional bids, renewable generation bounds, etc.)

It returns **computed objective values** for both perspectives.

```python
python

import numpy as np
from dataclasses import dataclass, field
from typing import List, Dict, Tuple


# ==============================================================
# 1. COMMON ENVIRONMENT
# ==============================================================

@dataclass
class MarketEnvironment:
    """Shared environment parameters and data."""
    mu_D: float
    sigma_D: float
    mu_bi: List[float]
    sigma_bi: List[float]
    mu_pi: List[float]
    sigma_pi: List[float]
    P_forecast: List[float]
    P_lower: List[float]
    P_upper: List[float]
    rho_target: float
    hours: int = 24

    def sample_demand(self) -> np.ndarray:
        return np.random.normal(self.mu_D, self.sigma_D, self.hours)

    def sample_conventional_bids(self) -> Tuple[np.ndarray, np.ndarray]:
        """Return (bids quantities, bid prices) for each conventional producer and hour."""
        n_conv = len(self.mu_bi)
        b = np.zeros((n_conv, self.hours))
```

```
        p = np.zeros((n_conv, self.hours))
        for i in range(n_conv):
            b[i, :] = np.random.normal(self.mu_bi[i], self.sigma_bi[i], self.hours)
            p[i, :] = np.random.normal(self.mu_pi[i], self.sigma_pi[i], self.hours)
        return b, p


# ================================================================
# 2. MARKET CLEARING (MERIT ORDER)
# ================================================================

def merit_order_clearing(bids: List[Tuple[float, float]], demand: float) -> Tuple[float,
List[int]]:
    """
    Compute market clearing price and acceptance indicators using merit order.
    :param bids: list of (quantity, price)
    :param demand: total demand
    :return: (clearing_price, acceptance_indicators)
    """
    # Sort by price ascending
    sorted_indices = np.argsort([p for (_, p) in bids])
    cumulative = 0.0
    accepted = [0] * len(bids)
    clearing_price = 0.0

    for idx in sorted_indices:
        b, p = bids[idx]
        if cumulative + b < demand:
            accepted[idx] = 1
            cumulative += b
        else:
            accepted[idx] = 1
            clearing_price = p
            break

    # If demand is exactly matched or exceeded
    if clearing_price == 0.0:
        clearing_price = bids[sorted_indices[-1]][1]

    return clearing_price, accepted


# ================================================================
# 3. RENEWABLE PRODUCER MODULE
# ================================================================

@dataclass
class RenewableProducer:
    """Renewable producer perspective."""
    env: MarketEnvironment
    b_t: List[float]
    p_t: List[float]
    q_u: float
    q_o: float

    def compute_profit(self, C_t, x_t, P_t) -> float:
        """Compute realized profit for one hour."""
        if x_t == 0:
            return 0.0
        under = max(self.b_t - P_t, 0)
        over = max(P_t - self.b_t, 0)
        return C_t * self.b_t - self.q_u * under - self.q_o * over

    def expected_profit(self, D_t, conv_bids_b, conv_bids_p) -> float:
        """Expected profit over random outcomes of demand and conventional bids."""
        hours = self.env.hours
        profits = np.zeros(hours)

        for t in range(hours):
            bids = [(self.b_t[t], self.p_t[t])]
            for i in range(len(conv_bids_b)):
                bids.append((conv_bids_b[i, t], conv_bids_p[i, t]))

            C_t, x_vec = merit_order_clearing(bids, D_t[t])
            x_r = x_vec[0]  # renewable's acceptance
```

```
            P_t = np.random.uniform(self.env.P_lower[t], self.env.P_upper[t])
            profits[t] = self.compute_profit(C_t, x_r, P_t)
        return np.mean(profits)

    def worst_case_penalty(self, D_t, conv_bids_b, conv_bids_p) -> float:
        """Compute worst-case penalty across generation uncertainty."""
        hours = self.env.hours
        penalty_total = 0.0

        for t in range(hours):
            bids = [(self.b_t[t], self.p_t[t])]
            for i in range(len(conv_bids_b)):
                bids.append((conv_bids_b[i, t], conv_bids_p[i, t]))

            C_t, x_vec = merit_order_clearing(bids, D_t[t])
            x_r = x_vec[0]
            if x_r == 1:
                # worst case: choose generation that maximizes penalty
                penalties = []
                for P_t in [self.env.P_lower[t], self.env.P_upper[t]]:
                    under = max(self.b_t[t] - P_t, 0)
                    over = max(P_t - self.b_t[t], 0)
                    penalties.append(self.q_u * under + self.q_o * over)
                penalty_total += max(penalties)
        return penalty_total


# ==============================================================
# 4. REGULATOR MODULE
# ==============================================================

@dataclass
class Regulator:
    """System regulator perspective."""
    env: MarketEnvironment
    q_u: float
    q_o: float
    rho_target: float

    def compute_system_metrics(
        self, D_t, conv_bids_b, conv_bids_p, ren_b_t, ren_p_t
    ) -> Dict[str, float]:
        """Compute system-level performance metrics."""
        hours = self.env.hours
        prices = np.zeros(hours)
        imbalances = np.zeros(hours)
        rho = np.zeros(hours)

        for t in range(hours):
            bids = [(ren_b_t[t], ren_p_t[t])]
            for i in range(len(conv_bids_b)):
                bids.append((conv_bids_b[i, t], conv_bids_p[i, t]))

            C_t, x_vec = merit_order_clearing(bids, D_t[t])
            prices[t] = C_t
            x_r = x_vec[0]
            P_t = np.random.uniform(self.env.P_lower[t], self.env.P_upper[t])

            # Delivered energy (approx.)
            delivered_total = 0.0
            if x_r == 1:
                delivered_total += min(ren_b_t[t], P_t)
            for i in range(1, len(bids)):
                if x_vec[i] == 1:
                    delivered_total += bids[i][0]

            imbalances[t] = abs(D_t[t] - delivered_total)
            rho[t] = x_r * ren_b_t[t] / max(D_t[t], 1e-6)

        # Objectives (quantifiable)
        J1_price_stability = -np.var(prices)
        J2_reliability = -np.mean(imbalances)
        J3_renewable_share = -np.sum(np.abs(rho - self.rho_target))

        return {
```

```
            "price_stability": J1_price_stability,
            "reliability": J2_reliability,
            "renewable_integration": J3_renewable_share,
        }


# ==============================================================
# 5. UNIFIED SIMULATION INTERFACE
# ==============================================================

def simulate_market(
    controls: Dict[str, List[float]],
    uncertainty: Dict[str, np.ndarray],
) -> Dict[str, float]:
    """
    Unified interface for simulating the market.
    Inputs:
        controls: dictionary of control variables
            {
                "b_t": [...],       # renewable bid quantities
                "p_t": [...],       # renewable bid prices
                "q_u": float,       # underdelivery penalty
                "q_o": float        # overdelivery penalty
            }
        uncertainty: dictionary of uncertain variables
            {
                "D_t": np.array([...]),           # hourly demands
                "conv_bids_b": np.array([...]),    # conventional bid quantities
                "conv_bids_p": np.array([...]),    # conventional bid prices
                "P_lower": [...],
                "P_upper": [...]
            }

    Output:
        Dictionary of objective values for both perspectives.
    """

    # Environment construction
    env = MarketEnvironment(
        mu_D=0, sigma_D=0,  # not used in simulation mode
        mu_bi=[0, 0, 0], sigma_bi=[0, 0, 0],
        mu_pi=[0, 0, 0], sigma_pi=[0, 0, 0],
        P_forecast=list((np.array(uncertainty["P_lower"]) + np.array(uncertainty["P_upper"])) /
2),
        P_lower=uncertainty["P_lower"],
        P_upper=uncertainty["P_upper"],
        rho_target=0.3,
        hours=len(uncertainty["D_t"])
    )

    # Initialize modules
    renewable = RenewableProducer(
        env=env,
        b_t=controls["b_t"],
        p_t=controls["p_t"],
        q_u=controls["q_u"],
        q_o=controls["q_o"],
    )

    regulator = Regulator(
        env=env,
        q_u=controls["q_u"],
        q_o=controls["q_o"],
        rho_target=env.rho_target,
    )

    # Run simulation
    D_t = uncertainty["D_t"]
    conv_bids_b = uncertainty["conv_bids_b"]
    conv_bids_p = uncertainty["conv_bids_p"]

    # Renewable objectives
    J1_r = renewable.expected_profit(D_t, conv_bids_b, conv_bids_p)
    J2_r = -renewable.worst_case_penalty(D_t, conv_bids_b, conv_bids_p)
```

```
    # Regulator objectives
    regulator_metrics = regulator.compute_system_metrics(
        D_t, conv_bids_b, conv_bids_p, controls["b_t"], controls["p_t"]
    )

    return {
        "renewable_expected_profit": J1_r,
        "renewable_worst_case_penalty": J2_r,
        "regulator_price_stability": regulator_metrics["price_stability"],
        "regulator_reliability": regulator_metrics["reliability"],
        "regulator_renewable_integration": regulator_metrics["renewable_integration"],
    }


# ==============================================================
# End of unified model implementation
# ==============================================================
```

✅ Explanation

- **`MarketEnvironment`**: shared parameters and sampling functions.
- **`merit_order_clearing`**: implements the merit-order market rule deterministically.
- **`RenewableProducer`**: handles bidding, profit calculation, and objective evaluation.
- **`Regulator`**: evaluates system-level outcomes and computes quantifiable objectives.
- **`simulate_market()`**: unified interface for simulation — accepts control & uncertainty dictionaries and outputs objective values for both perspectives.

Example (conceptual, not to execute here)

```python
controls = {
    "b_t": [50]*24,
    "p_t": [40]*24,
    "q_u": 100,
    "q_o": 20
}

uncertainty = {
    "D_t": np.random.normal(200, 20, 24),
    "conv_bids_b": np.random.normal(60, 5, (3, 24)),
    "conv_bids_p": np.random.normal(35, 5, (3, 24)),
    "P_lower": [40]*24,
    "P_upper": [60]*24
}

results = simulate_market(controls, uncertainty)
print(results)
```

This unified structure is ready for **simulation, calibration, and optimization**, and all modules are internally consistent and quantifiable.

There is an error in your Python implementation. The system electricity demand D_t and the bids of the conventional producers (b_{jt}, p_{jt}) should be sampled from the normal distributions with constant means and variances. The program should not take them as inputs.

Excellent — you're absolutely right.
In the unified model, D_t and the conventional producers' bids (b_{jt}, p_{jt}) are **random variables drawn from known normal distributions** with fixed means and variances.
They **should not be external inputs**; instead, the simulation should **sample them internally** based on parameters in the shared environment.

Below is the **corrected and internally consistent Python implementation**, where:

- Demand and conventional bids are **sampled** using the environment's distributional parameters.
- The interface only takes control variables (from both perspectives) and renewable generation uncertainty bounds.

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


# ============================================================
# 1. COMMON ENVIRONMENT
# ============================================================

@dataclass
class MarketEnvironment:
    """Shared environment parameters and stochastic sampling."""
    mu_D: float
    sigma_D: float
    mu_bi: List[float]
    sigma_bi: List[float]
    mu_pi: List[float]
    sigma_pi: List[float]
    P_forecast: List[float]
    P_lower: List[float]
    P_upper: List[float]
    rho_target: float
    hours: int = 24

    def sample_demand(self) -> np.ndarray:
        """Sample hourly electricity demand."""
        return np.random.normal(self.mu_D, self.sigma_D, self.hours)

    def sample_conventional_bids(self) -> Tuple[np.ndarray, np.ndarray]:
        """Sample bids (quantities, prices) for conventional producers."""
        n_conv = len(self.mu_bi)
        b = np.zeros((n_conv, self.hours))
        p = np.zeros((n_conv, self.hours))
        for i in range(n_conv):
            b[i, :] = np.random.normal(self.mu_bi[i], self.sigma_bi[i], self.hours)
            p[i, :] = np.random.normal(self.mu_pi[i], self.sigma_pi[i], self.hours)
        return b, p


# ============================================================
# 2. MARKET CLEARING (MERIT ORDER)
# ============================================================

def merit_order_clearing(bids: List[Tuple[float, float]], demand: float) -> Tuple[float,
List[int]]:
    """
    Compute market clearing price and acceptance indicators using merit-order.
    :param bids: list of (quantity, price)
    :param demand: total demand
```

```
    :return: (clearing_price, acceptance_indicators)
    """
    sorted_indices = np.argsort([p for (_, p) in bids])
    cumulative = 0.0
    accepted = [0] * len(bids)
    clearing_price = 0.0

    for idx in sorted_indices:
        b, p = bids[idx]
        cumulative += b
        accepted[idx] = 1
        if cumulative >= demand:
            clearing_price = p
            break

    # If demand never reached, price = highest accepted bid
    if clearing_price == 0.0:
        clearing_price = bids[sorted_indices[-1]][1]

    return clearing_price, accepted


# ==============================================================
# 3. RENEWABLE PRODUCER MODULE
# ==============================================================

@dataclass
class RenewableProducer:
    """Renewable producer's decision and objectives."""
    env: MarketEnvironment
    b_t: List[float]
    p_t: List[float]
    q_u: float
    q_o: float

    def compute_profit(self, C_t, x_t, P_t, b_t):
        """Compute realized profit for one hour."""
        if x_t == 0:
            return 0.0
        under = max(b_t - P_t, 0)
        over = max(P_t - b_t, 0)
        return C_t * b_t - self.q_u * under - self.q_o * over

    def expected_profit(self, D_t, conv_bids_b, conv_bids_p) -> float:
        """Expected profit over market and generation uncertainty."""
        profits = []
        for t in range(self.env.hours):
            bids = [(self.b_t[t], self.p_t[t])]
            for i in range(len(conv_bids_b)):
                bids.append((conv_bids_b[i, t], conv_bids_p[i, t]))
            C_t, x_vec = merit_order_clearing(bids, D_t[t])
            x_r = x_vec[0]
            P_t = np.random.uniform(self.env.P_lower[t], self.env.P_upper[t])
            profits.append(self.compute_profit(C_t, x_r, P_t, self.b_t[t]))
        return np.mean(profits)

    def worst_case_penalty(self, D_t, conv_bids_b, conv_bids_p) -> float:
        """Compute worst-case penalty across renewable generation uncertainty."""
        total_penalty = 0.0
        for t in range(self.env.hours):
            bids = [(self.b_t[t], self.p_t[t])]
            for i in range(len(conv_bids_b)):
                bids.append((conv_bids_b[i, t], conv_bids_p[i, t]))
            C_t, x_vec = merit_order_clearing(bids, D_t[t])
            if x_vec[0] == 1:
                # Compute penalties for extreme generation values
                penalties = []
                for P_t in [self.env.P_lower[t], self.env.P_upper[t]]:
                    under = max(self.b_t[t] - P_t, 0)
                    over = max(P_t - self.b_t[t], 0)
                    penalties.append(self.q_u * under + self.q_o * over)
                total_penalty += max(penalties)
        return total_penalty
```

```python
# ==============================================================
# 4. REGULATOR MODULE
# ==============================================================

@dataclass
class Regulator:
    """System regulator perspective and objectives."""
    env: MarketEnvironment
    q_u: float
    q_o: float

    def compute_system_metrics(
        self, D_t, conv_bids_b, conv_bids_p, ren_b_t, ren_p_t
    ) -> Dict[str, float]:
        """Compute regulator's measurable objectives."""
        prices = np.zeros(self.env.hours)
        imbalances = np.zeros(self.env.hours)
        renewable_shares = np.zeros(self.env.hours)

        for t in range(self.env.hours):
            bids = [(ren_b_t[t], ren_p_t[t])]
            for i in range(len(conv_bids_b)):
                bids.append((conv_bids_b[i, t], conv_bids_p[i, t]))
            C_t, x_vec = merit_order_clearing(bids, D_t[t])
            prices[t] = C_t
            x_r = x_vec[0]
            P_t = np.random.uniform(self.env.P_lower[t], self.env.P_upper[t])

            # Delivered supply
            delivered = 0.0
            if x_r == 1:
                delivered += min(ren_b_t[t], P_t)
            for i in range(1, len(bids)):
                if x_vec[i] == 1:
                    delivered += bids[i][0]

            imbalances[t] = abs(D_t[t] - delivered)
            renewable_shares[t] = x_r * ren_b_t[t] / max(D_t[t], 1e-6)

        # Quantifiable objectives
        J1_price_stability = -np.var(prices)
        J2_reliability = -np.mean(imbalances)
        J3_renewable_integration = -np.sum(np.abs(renewable_shares - self.env.rho_target))

        return {
            "price_stability": J1_price_stability,
            "reliability": J2_reliability,
            "renewable_integration": J3_renewable_integration,
        }


# ==============================================================
# 5. UNIFIED SIMULATION INTERFACE
# ==============================================================

def simulate_market(controls: Dict[str, List[float]], params: Dict[str, float]) -> Dict[str,
float]:
    """
    Unified interface for simulating the unified electricity market model.
    ------------------------------------------------------------------
    Inputs:
        controls: dict of control variables
            {
                "b_t": list of renewable bid quantities (len=24),
                "p_t": list of renewable bid prices (len=24),
                "q_u": underdelivery penalty,
                "q_o": overdelivery penalty
            }

        params: dict of exogenous parameter values
            {
                "mu_D", "sigma_D",
                "mu_bi", "sigma_bi",
                "mu_pi", "sigma_pi",
                "P_lower", "P_upper",
```

```
                "rho_target"
            }

    Output:
        Dict containing objective values for both perspectives.
    """

    # === Construct common environment ===
    env = MarketEnvironment(
        mu_D=params["mu_D"],
        sigma_D=params["sigma_D"],
        mu_bi=params["mu_bi"],
        sigma_bi=params["sigma_bi"],
        mu_pi=params["mu_pi"],
        sigma_pi=params["sigma_pi"],
        P_forecast=list((np.array(params["P_lower"]) + np.array(params["P_upper"])) / 2),
        P_lower=params["P_lower"],
        P_upper=params["P_upper"],
        rho_target=params["rho_target"],
        hours=len(params["P_lower"]),
    )

    # === Sample uncertainties ===
    D_t = env.sample_demand()
    conv_bids_b, conv_bids_p = env.sample_conventional_bids()

    # === Initialize agents ===
    renewable = RenewableProducer(
        env=env,
        b_t=controls["b_t"],
        p_t=controls["p_t"],
        q_u=controls["q_u"],
        q_o=controls["q_o"],
    )
    regulator = Regulator(env=env, q_u=controls["q_u"], q_o=controls["q_o"])

    # === Compute renewable objectives ===
    J1_r = renewable.expected_profit(D_t, conv_bids_b, conv_bids_p)
    J2_r = -renewable.worst_case_penalty(D_t, conv_bids_b, conv_bids_p)

    # === Compute regulator objectives ===
    regulator_metrics = regulator.compute_system_metrics(
        D_t, conv_bids_b, conv_bids_p, controls["b_t"], controls["p_t"]
    )

    # === Return unified results ===
    return {
        "renewable_expected_profit": J1_r,
        "renewable_worst_case_penalty": J2_r,
        "regulator_price_stability": regulator_metrics["price_stability"],
        "regulator_reliability": regulator_metrics["reliability"],
        "regulator_renewable_integration": regulator_metrics["renewable_integration"],
    }


# ================================================================
# End of unified model implementation
# ================================================================
```

✅ Summary of Fixes

- The system demand D_t and conventional producers' bids (b_{jt}, p_{jt}) are now **sampled internally** using their respective normal distributions (**`MarketEnvironment.sample_demand()`** and **`.sample_conventional_bids()`**).
- The simulation interface no longer requires them as inputs.

- Only **control variables** (bids and penalty rates) and **known parameters** (means, variances, renewable generation bounds) are user inputs.

Example (not to execute here)

```python
controls = {
    "b_t": [50.0]*24,
    "p_t": [40.0]*24,
    "q_u": 100.0,
    "q_o": 20.0
}

params = {
    "mu_D": 200.0,
    "sigma_D": 20.0,
    "mu_bi": [60.0, 70.0, 80.0],
    "sigma_bi": [5.0, 5.0, 5.0],
    "mu_pi": [30.0, 35.0, 40.0],
    "sigma_pi": [3.0, 3.0, 3.0],
    "P_lower": [40.0]*24,
    "P_upper": [60.0]*24,
    "rho_target": 0.3
}

results = simulate_market(controls, params)
print(results)
```

This version is now **internally self-contained**, **quantitatively consistent**, and **directly runnable** for simulation or analysis.

mu_D, sigma_D, mu_bi, sigma_bi, mu_pi, sigma_pi are constants instead of uncertain parameters, so the implementation should not take them as inputs. In contrast, P_lower and P_upper are uncertain, so they should be taken as inputs. Please refine your implementation.

Perfect — this final refinement makes the unified model fully consistent with the problem specification:

- The constants $\mu_D, \sigma_D, \mu_{b_i}, \sigma_{b_i}, \mu_{p_i}, \sigma_{p_i}$ are **fixed constants** (embedded in the code, not passed as inputs).
- The **renewable generation bounds** P_{lower} and P_{upper} remain **uncertain inputs** because they describe the deep uncertainty in renewable generation.

Below is the **refined, final, modular, and internally consistent Python implementation**.
Everything is quantifiable, self-contained, and consistent with the shared environment and merit-order market-clearing mechanism.

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

```python
# ==============================================================
# 1. COMMON ENVIRONMENT
# ==============================================================

@dataclass
class MarketEnvironment:
    """Shared environment with fixed constants and stochastic sampling."""
    # --- Fixed constants ---
    mu_D: float = 200.0
    sigma_D: float = 20.0
    mu_bi: List[float] = None
    sigma_bi: List[float] = None
    mu_pi: List[float] = None
    sigma_pi: List[float] = None
    rho_target: float = 0.3
    hours: int = 24

    # --- Uncertain renewable generation bounds ---
    P_lower: List[float] = None
    P_upper: List[float] = None

    def __post_init__(self):
        if self.mu_bi is None:
            self.mu_bi = [60.0, 70.0, 80.0]
        if self.sigma_bi is None:
            self.sigma_bi = [5.0, 5.0, 5.0]
        if self.mu_pi is None:
            self.mu_pi = [30.0, 35.0, 40.0]
        if self.sigma_pi is None:
            self.sigma_pi = [3.0, 3.0, 3.0]

    # --- Sampling methods for stochastic variables ---
    def sample_demand(self) -> np.ndarray:
        """Sample hourly electricity demand."""
        return np.random.normal(self.mu_D, self.sigma_D, self.hours)

    def sample_conventional_bids(self) -> Tuple[np.ndarray, np.ndarray]:
        """Sample bids (quantities, prices) for conventional producers."""
        n_conv = len(self.mu_bi)
        b = np.zeros((n_conv, self.hours))
        p = np.zeros((n_conv, self.hours))
        for i in range(n_conv):
            b[i, :] = np.random.normal(self.mu_bi[i], self.sigma_bi[i], self.hours)
            p[i, :] = np.random.normal(self.mu_pi[i], self.sigma_pi[i], self.hours)
        return b, p


# ==============================================================
# 2. MARKET CLEARING (MERIT ORDER)
# ==============================================================

def merit_order_clearing(bids: List[Tuple[float, float]], demand: float) -> Tuple[float,
List[int]]:
    """
    Compute market clearing price and acceptance indicators using merit order.
    :param bids: list of (quantity, price)
    :param demand: total demand
    :return: (clearing_price, acceptance_indicators)
    """
    sorted_indices = np.argsort([p for (_, p) in bids])
    cumulative = 0.0
    accepted = [0] * len(bids)
    clearing_price = 0.0

    for idx in sorted_indices:
        b, p = bids[idx]
        cumulative += b
        accepted[idx] = 1
        if cumulative >= demand:
            clearing_price = p
            break

    if clearing_price == 0.0:
        clearing_price = bids[sorted_indices[-1]][1]
```

```
    return clearing_price, accepted


# ==============================================================
# 3. RENEWABLE PRODUCER MODULE
# ==============================================================

@dataclass
class RenewableProducer:
    """Renewable producer's decision and objectives."""
    env: MarketEnvironment
    b_t: List[float]
    p_t: List[float]
    q_u: float
    q_o: float

    def compute_profit(self, C_t, x_t, P_t, b_t):
        """Compute realized profit for one hour."""
        if x_t == 0:
            return 0.0
        under = max(b_t - P_t, 0)
        over = max(P_t - b_t, 0)
        return C_t * b_t - self.q_u * under - self.q_o * over

    def expected_profit(self, D_t, conv_bids_b, conv_bids_p) -> float:
        """Expected profit over market and generation uncertainty."""
        profits = []
        for t in range(self.env.hours):
            bids = [(self.b_t[t], self.p_t[t])]
            for i in range(len(conv_bids_b)):
                bids.append((conv_bids_b[i, t], conv_bids_p[i, t]))
            C_t, x_vec = merit_order_clearing(bids, D_t[t])
            x_r = x_vec[0]
            P_t = np.random.uniform(self.env.P_lower[t], self.env.P_upper[t])
            profits.append(self.compute_profit(C_t, x_r, P_t, self.b_t[t]))
        return np.mean(profits)

    def worst_case_penalty(self, D_t, conv_bids_b, conv_bids_p) -> float:
        """Compute worst-case penalty across renewable generation uncertainty."""
        total_penalty = 0.0
        for t in range(self.env.hours):
            bids = [(self.b_t[t], self.p_t[t])]
            for i in range(len(conv_bids_b)):
                bids.append((conv_bids_b[i, t], conv_bids_p[i, t]))
            C_t, x_vec = merit_order_clearing(bids, D_t[t])
            if x_vec[0] == 1:
                # Penalties for extreme generation values
                penalties = []
                for P_t in [self.env.P_lower[t], self.env.P_upper[t]]:
                    under = max(self.b_t[t] - P_t, 0)
                    over = max(P_t - self.b_t[t], 0)
                    penalties.append(self.q_u * under + self.q_o * over)
                total_penalty += max(penalties)
        return total_penalty


# ==============================================================
# 4. REGULATOR MODULE
# ==============================================================

@dataclass
class Regulator:
    """System regulator perspective and objectives."""
    env: MarketEnvironment
    q_u: float
    q_o: float

    def compute_system_metrics(
        self, D_t, conv_bids_b, conv_bids_p, ren_b_t, ren_p_t
    ) -> Dict[str, float]:
        """Compute regulator's measurable objectives."""
        prices = np.zeros(self.env.hours)
        imbalances = np.zeros(self.env.hours)
        renewable_shares = np.zeros(self.env.hours)
```

```
        for t in range(self.env.hours):
            bids = [(ren_b_t[t], ren_p_t[t])]
            for i in range(len(conv_bids_b)):
                bids.append((conv_bids_b[i, t], conv_bids_p[i, t]))
            C_t, x_vec = merit_order_clearing(bids, D_t[t])
            prices[t] = C_t
            x_r = x_vec[0]
            P_t = np.random.uniform(self.env.P_lower[t], self.env.P_upper[t])

            # Delivered supply
            delivered = 0.0
            if x_r == 1:
                delivered += min(ren_b_t[t], P_t)
            for i in range(1, len(bids)):
                if x_vec[i] == 1:
                    delivered += bids[i][0]

            imbalances[t] = abs(D_t[t] - delivered)
            renewable_shares[t] = x_r * ren_b_t[t] / max(D_t[t], 1e-6)

        # Quantifiable objectives
        J1_price_stability = -np.var(prices)
        J2_reliability = -np.mean(imbalances)
        J3_renewable_integration = -np.sum(np.abs(renewable_shares - self.env.rho_target))

        return {
            "price_stability": J1_price_stability,
            "reliability": J2_reliability,
            "renewable_integration": J3_renewable_integration,
        }


# ==============================================================
# 5. UNIFIED SIMULATION INTERFACE
# ==============================================================

def simulate_market(controls: Dict[str, List[float]], uncertainty: Dict[str, List[float]]) ->
Dict[str, float]:
    """
    Unified interface for simulating the unified electricity market model.
    --------------------------------------------------------------------
    Inputs:
        controls: dict of control variables
            {
                "b_t": list of renewable bid quantities (len=24),
                "p_t": list of renewable bid prices (len=24),
                "q_u": underdelivery penalty (float),
                "q_o": overdelivery penalty (float)
            }

        uncertainty: dict of uncertain renewable generation bounds
            {
                "P_lower": [...],
                "P_upper": [...]
            }

    Output:
        Dict containing objective values for both perspectives.
    """

    # === Construct environment (constants embedded, uncertain inputs provided) ===
    env = MarketEnvironment(
        P_lower=uncertainty["P_lower"],
        P_upper=uncertainty["P_upper"],
    )

    # === Sample endogenous uncertainties ===
    D_t = env.sample_demand()
    conv_bids_b, conv_bids_p = env.sample_conventional_bids()

    # === Initialize agents ===
    renewable = RenewableProducer(
        env=env,
        b_t=controls["b_t"],
        p_t=controls["p_t"],
```

```
        q_u=controls["q_u"],
        q_o=controls["q_o"],
    )

    regulator = Regulator(env=env, q_u=controls["q_u"], q_o=controls["q_o"])

    # === Compute objectives ===
    J1_r = renewable.expected_profit(D_t, conv_bids_b, conv_bids_p)
    J2_r = -renewable.worst_case_penalty(D_t, conv_bids_b, conv_bids_p)
    regulator_metrics = regulator.compute_system_metrics(
        D_t, conv_bids_b, conv_bids_p, controls["b_t"], controls["p_t"]
    )

    # === Return unified results ===
    return {
        "renewable_expected_profit": J1_r,
        "renewable_worst_case_penalty": J2_r,
        "regulator_price_stability": regulator_metrics["price_stability"],
        "regulator_reliability": regulator_metrics["reliability"],
        "regulator_renewable_integration": regulator_metrics["renewable_integration"],
    }


# ================================================================
# End of unified model implementation
# ================================================================
```

✅ Key Corrections

Component	Change	Reason
`mu_D`, **`sigma_D`**, **`mu_bi`**, **`sigma_bi`**, **`mu_pi`**, **`sigma_pi`**	Hardcoded constants in **`MarketEnvironment`**	They are **fixed constants**, not uncertain inputs
`P_lower`, **`P_upper`**	Passed as arguments to the simulation	They represent **deep uncertainty** in renewable generation
`simulate_market()`	Now only takes control variables and renewable generation bounds	Consistent with problem statement
Market sampling	Demand and conventional bids are sampled internally	Reflects stochastic behavior based on fixed parameters

Example Usage (conceptual)

```python
controls = {
    "b_t": [50.0]*24,
    "p_t": [40.0]*24,
    "q_u": 100.0,
    "q_o": 20.0
}

uncertainty = {
    "P_lower": [40.0]*24,
    "P_upper": [60.0]*24
}

results = simulate_market(controls, uncertainty)
print(results)
```

✅ Summary

This final version is:

- **Internally consistent** across all perspectives.
- **Quantifiable and implementable** — no conceptual placeholders.
- **Self-contained** — constants are embedded, uncertain parameters (generation bounds) are inputs.
- **Modular** — each class (RenewableProducer, Regulator) can be tested or extended independently.