

Experiments 3 - Market 4



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 $t=1, \dots, 24$, 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 σ_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're essentially describing a **stochastic bidding optimization model** for a **renewable energy producer** in a **day-ahead electricity market** under **price and generation uncertainty**.

Let's carefully formalize this step by step.

1. Decision Maker and Objective

A *renewable energy producer* (agent) seeks to **maximize expected profit** from selling electricity in a day-ahead market, subject to:

- uncertain renewable generation (weather-driven),
 - uncertain market-clearing prices (competition-driven),
 - imbalance penalties for deviations from committed quantities.
-

2. Time Horizon

- Time is discretized into hourly intervals $t = 1, \dots, 24$ (for the next day).

Each hour is an independent (or weakly correlated) decision stage in a multi-period stochastic setting.

3. Decision Variables

For each hour t :

$b_t \geq 0$: quantity of electricity bid (MWh)

$p_t \geq 0$: minimum acceptable price per MWh

4. Uncertain Variables

For each hour t :

Symbol	Meaning	Distribution / Notes
P_t	Actual renewable generation (MWh)	random variable, distribution derived from weather forecast
c_t	Market-clearing price (€/MWh)	random variable, depends on aggregate bids & demand
D_t	Market demand (MWh)	$D_t \sim \mathcal{N}(\mu_D, \sigma_D^2)$
(b_{it}, p_{it})	Bids from conventional producers $i = 1, 2, 3$	stochastic, known distributions
q_u, q_o	penalty rates for under-/over-delivery	deterministic parameters

5. Market-Clearing Mechanism

For each hour t :

- The market operator determines c_t and accepted bids such that **supply = demand**:

$$\sum_{i \in \mathcal{I}} b_{it} \cdot \mathbf{1}\{p_{it} \leq c_t\} + b_t \cdot \mathbf{1}\{p_t \leq c_t\} = D_t$$

- If the renewable producer's bid is **accepted** ($p_t \leq c_t$), it must deliver b_t .

6. Profit Function per Hour

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

$$\pi_t = c_t b_t - q_u \max(0, b_t - P_t) - q_o \max(0, P_t - b_t)$$

If rejected ($p_t > c_t$):

$$\pi_t = 0$$

Hence the stochastic profit:

$$\pi_t(b_t, p_t, P_t, c_t) = \mathbf{1}\{p_t \leq c_t\} [c_t b_t - q_u(b_t - P_t)^+ - q_o(P_t - b_t)^+]$$

7. Objective Function

The renewable producer maximizes **expected total profit**:

$$\max_{\{b_t, p_t\}_{t=1}^{24}} \mathbb{E} \left[\sum_{t=1}^{24} \pi_t(b_t, p_t, P_t, c_t) \right]$$

Subject to:

- $b_t, p_t \geq 0$
- (optionally) operational constraints such as $b_t \leq \bar{P}_t^{\text{forecast}}$

8. State Variables

If we view this as a **stochastic dynamic decision process**, we can define:

Symbol	Description
s_t	state at time t
$s_t = (\text{forecast distribution of } P_t, \text{price distribution of } c_t, \text{market parameters}, t)$	

These summarize all information available at decision time.

9. State Transition

The **state transition function** updates the beliefs about renewable generation and market prices as time progresses:

$$s_{t+1} = f(s_t, b_t, p_t, \omega_{t+1})$$

where ω_{t+1} represents new stochastic realizations (updated forecasts, weather info, etc.).

If we assume independence across hours (common in day-ahead markets), then each hour can be treated as an independent optimization stage:

$$s_{t+1} = s_0, \text{ independent of } b_t, p_t$$

10. Initial State

At the time of bidding (noon of day D-1), the producer knows:

- forecast distributions of renewable output P_t ,
- estimated distribution of market prices c_t ,
- penalty parameters q_u, q_o .

Hence:

$$s_1 = \{\mathbb{P}(P_t), \mathbb{P}(c_t), q_u, q_o, t = 1, \dots, 24\}$$

11. Compact Model Summary

Decision Problem:

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

Stochastic Elements:

$(P_t, c_t, D_t, b_{it}, p_{it}) \sim$ joint distribution from forecasts and competition.

State Dynamics:

$$s_{t+1} = f(s_t, b_t, p_t, \omega_{t+1})$$

Initial State:

$$s_1 = (\text{forecast info, price distribution, penalties, parameters})$$

12. Interpretation

- This is a **stochastic optimization under uncertainty** (can be solved via stochastic programming, robust optimization, or reinforcement learning).
- If hourly independence is assumed, the problem decomposes into **24 independent stochastic optimization problems**:

$$\max_{b_t, p_t} \mathbb{E} [\pi_t(b_t, p_t, P_t, c_t)]$$

- Otherwise, correlations between hours (e.g., price and wind autocorrelation) would create a **multistage stochastic program**.

Would you like me to express this as a **mathematical programming formulation** (ready for implementation, e.g., in Pyomo or Gurobi) — or as a **Markov decision process (MDP)** specification (states, actions, transitions, rewards, etc.)?



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 merit-order 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 — that's a very helpful clarification.

Let's now **refine the model** to correctly reflect the three key points you raised:

Refined Formal Model for the Renewable Producer's Bidding Problem

1. Overview

We model the renewable energy producer's **day-ahead bidding problem** in a **merit-order electricity market** with the following properties:

- The **renewable generation is deeply uncertain** — its probability distribution is *not known*.
- The **market-clearing price is endogenous**, i.e. it results from the aggregation of bids (supply) and stochastic demand.
- Conventional producers' bids** are stochastic (normal distributions), providing a source of *price uncertainty* from the renewable producer's perspective.

The renewable producer thus faces a **distributionally-robust optimization problem** where the main uncertainty (renewable generation) is *ambiguously distributed*, while prices emerge endogenously from the stochastic market environment.

2. Sets and Indices

- $t \in \mathcal{T} = \{1, \dots, 24\}$: hourly market periods
- $i \in \mathcal{I} = \{1, 2, 3\}$: conventional producers
- R : the renewable producer (decision maker)

3. Decision Variables (for the renewable producer)

For each hour t :

$$b_t \geq 0 \quad (\text{bid quantity, MWh})$$

$$p_t \geq 0 \quad (\text{offer price, €/MWh})$$

4. Exogenous Parameters

Symbol	Description
q_u	penalty rate for under-delivery (€/MWh shortfall)
q_o	penalty rate for over-generation / curtailment (€/MWh surplus)
μ_{ib}, σ_{ib}	mean and standard deviation of bid quantity b_{it} for conventional producer i
μ_{ip}, σ_{ip}	mean and standard deviation of bid price p_{it} for conventional producer i
μ_D, σ_D	mean and standard deviation of market demand D_t

5. Stochastic Quantities

Symbol	Description	Distribution
D_t	total demand in hour t	$\mathcal{N}(\mu_D, \sigma_D^2)$
(b_{it}, p_{it})	conventional producer i 's bid pair	$b_{it} \sim \mathcal{N}(\mu_{ib}, \sigma_{ib}^2), p_{it} \sim \mathcal{N}(\mu_{ip}, \sigma_{ip}^2)$, independent across i, t
P_t	renewable generation in hour t	<i>deeply uncertain</i> , no known distribution; only known support or plausible range $[0, \bar{P}_t]$

6. Market-Clearing Mechanism (Endogenous Price)

For each hour t :

- The system operator collects all bids $(b_{it}, p_{it})_{i \in \mathcal{I}}$ and (b_t, p_t) from the renewable producer.
- The operator constructs the **merit order**: ascending by offer price.
- The **market-clearing price** c_t is the marginal offer price at which total accepted supply equals realized demand D_t :

$$c_t = \min \left\{ p : \sum_{i \in \mathcal{I}} b_{it} \mathbf{1}\{p_{it} \leq p\} + b_t \mathbf{1}\{p_t \leq p\} \geq D_t \right\}$$

Thus, c_t is **endogenous** — determined by the random bids of conventional producers and demand realization, not exogenously drawn.

7. Acceptance Condition

For the renewable producer:

$$\text{Bid accepted in hour } t \iff p_t \leq c_t$$

If not accepted ($p_t > c_t$), the producer earns zero in that hour.

8. Delivery and Penalty Rules

If accepted:

- Committed delivery: b_t
- Realized renewable output: P_t
- **Shortfall:** $(b_t - P_t)^+ = \max(0, b_t - P_t)$
- **Surplus:** $(P_t - b_t)^+ = \max(0, P_t - b_t)$

Profit for hour t :

$$\pi_t(b_t, p_t, P_t, c_t) = \mathbf{1}\{p_t \leq c_t\} [c_t b_t - q_u(b_t - P_t)^+ - q_o(P_t - b_t)^+]$$

9. Deep Uncertainty in Renewable Generation

Since the distribution of P_t is **unknown**, we model the producer's decision under *ambiguity*.

Let \mathcal{P}_t denote the **ambiguity set** — the family of all plausible distributions consistent with known information (e.g., bounds or moments):

$$\mathcal{P}_t = \left\{ \text{all distributions of } P_t \text{ supported on } [0, \bar{P}_t] \text{ and } \mathbb{E}[P_t] \in [\underline{\mu}_t, \bar{\mu}_t] \right\}$$

The renewable producer optimizes against the *worst-case* expected penalty within \mathcal{P}_t .

10. Objective Function

The renewable producer seeks to **maximize robust expected profit**:

$$\max_{\{b_t, p_t\}} \mathbb{E}_{D_t, (b_{it}, p_{it})_{i \in \mathcal{I}}} \left[\min_{\mathbb{P}_t \in \mathcal{P}_t} \mathbb{E}_{P_t \sim \mathbb{P}_t} \left[\sum_{t=1}^{24} \mathbf{1}\{p_t \leq c_t(D_t, \mathbf{b}_{-R,t}, \mathbf{p}_{-R,t}, b_t, p_t)\} (c_t b_t - q_u(b_t - P_t)^+ - q_o(P_t - b_t)^+) \right] \right]$$

where:

- $\mathbf{b}_{-R,t} = (b_{1t}, b_{2t}, b_{3t})$,
- $\mathbf{p}_{-R,t} = (p_{1t}, p_{2t}, p_{3t})$.

Thus, the renewable producer maximizes its **expected profit under worst-case generation outcomes**, given the stochastic market environment.

11. State–Action–Transition Structure

Component	Symbol	Definition
State	$s_t = (\mathcal{P}_t, \mu_D, \sigma_D, \{\mu_{ib}, \sigma_{ib}, \mu_{ip}, \sigma_{ip}\}_{i \in \mathcal{I}}, q_u, q_o)$	Information available at bidding time (beliefs, parameters, and ambiguity set)
Action	$a_t = (b_t, p_t)$	Bid decision for hour t
Uncertainty	$\omega_t = (D_t, (b_{it}, p_{it})_{i \in \mathcal{I}}, P_t)$	Realizations of demand, competitor bids, and renewable generation
Transition	$s_{t+1} = f(s_t, a_t, \omega_t)$	Updates beliefs or ambiguity set if information evolves; if independent hours \rightarrow static s_t
Reward	$r_t(a_t, \omega_t) = \pi_t(b_t, p_t, P_t, c_t(\omega_t))$	Realized profit per hour

12. Initial State

At bidding time (noon, day D-1):

$$s_1 = \{\mathcal{P}_t, (\mu_D, \sigma_D), \{(\mu_{ib}, \sigma_{ib}, \mu_{ip}, \sigma_{ip})\}_{i \in \mathcal{I}}, q_u, q_o\}$$

These summarize all forecast and penalty information available before the day-ahead submission.

13. Problem Type

This formulation defines a **distributionally robust stochastic optimization** (or a **robust MDP**, if hours are dynamically linked):

$$\begin{aligned} & \max_{(b_t, p_t)} \mathbb{E}_{\text{market uncertainty}} \left[\min_{\mathbb{P}_t \in \mathcal{P}_t} \mathbb{E}_{P_t \sim \mathbb{P}_t} [\pi_t(b_t, p_t, P_t, c_t)] \right] \\ & \text{s.t. } b_t, p_t \geq 0, \quad t = 1, \dots, 24 \end{aligned}$$

If hours are independent, it decomposes by hour.

14. Interpretation

- The **renewable producer** chooses bids b_t, p_t considering:
 - Market uncertainty from *conventional competitors* and *demand* (probabilistic).
 - *Deep uncertainty* in its own generation (modeled via an ambiguity set).
- The **clearing price** c_t arises endogenously from the **merit-order dispatch** given all bids and realized demand.
- The **objective** captures **expected profit under worst-case generation** and expected market conditions.

Would you like me to extend this formulation into a **mathematical program** (e.g., an explicit optimization model with constraints that compute c_t) or a **robust MDP** specification ready for computational methods (e.g., reinforcement learning / robust optimization)?



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 — now you're moving toward a **multi-agent modeling framework** for the market, in which the **renewable producer** and the **system regulator** act as distinct decision-makers interacting through a **shared environment** (the electricity market).

Below is a careful and complete specification that keeps all mathematical components consistent with the previous discussion but explicitly separates **shared (environmental) variables** and **agent-specific control, state, transition, and objective structures**.

Common Environment: Day-Ahead Electricity Market

The environment represents the **market system** that evolves as both agents act.

It contains the **global variables**, **shared stochastic processes**, and **rules** linking agents' decisions.

1. Global Sets and Indices

- $t \in \mathcal{T} = \{1, \dots, 24\}$: hourly intervals.
- $i \in \mathcal{I} = \{1, 2, 3\}$: conventional producers.
- R : renewable producer (the decision-making agent).
- $G = \mathcal{I} \cup \{R\}$: all generation units.

2. Global Exogenous Parameters (known to all)

Symbol	Description
μ_D, σ_D	Mean and std. dev. of hourly demand $D_t \sim \mathcal{N}(\mu_D, \sigma_D^2)$
$(\mu_{ib}, \sigma_{ib}), (\mu_{ip}, \sigma_{ip})$	Statistical parameters for conventional producers' bids
q_u, q_o	Under- and over-delivery penalty rates (€/MWh)
\bar{P}_t	Maximum physically possible renewable generation (MWh)

Symbol	Description
Ω_t	Environmental/meteorological uncertainty influencing P_t (deep uncertainty)

3. Global Random Variables (shared)

Symbol	Description	Notes
D_t	Aggregate electricity demand	stochastic, normal
(b_{it}, p_{it})	Conventional producers' bids	normal, independent
P_t	Renewable's actual generation	deeply uncertain, distribution unknown
c_t	Market-clearing price	endogenous equilibrium variable

4. Global Market-Clearing Rule (shared mechanism)

For each hour t , the system operator (within the environment) clears the market:

$$c_t = \min \left\{ p : \sum_{g \in G} b_{gt} \mathbf{1}\{p_{gt} \leq p\} \geq D_t \right\}$$

and determines dispatch commitments:

$$x_{gt} = b_{gt} \mathbf{1}\{p_{gt} \leq c_t\}, \quad \forall g \in G$$

so that

$$\sum_{g \in G} x_{gt} = D_t$$

under ideal market equilibrium.

This mechanism defines how agents' control variables (b_{gt}, p_{gt}) jointly determine the endogenous variable c_t .

Perspective 1: The Renewable Producer

1. Role

The renewable producer chooses its day-ahead bids to **maximize its own profit and/or reliability**, given uncertainty in generation and competitors' stochastic bids.

2. Control Variables (decisions)

Variable	Description
$b_t \geq 0$	Bid quantity for hour t
$p_t \geq 0$	Offer price for hour t

Collectively $a_t^R = (b_t, p_t)$.

3. State Variables

At bidding time:

$$s_t^R = (\text{forecast information on } \Omega_t, \mathcal{P}_t, \text{ beliefs on } D_t, q_u, q_o)$$

where \mathcal{P}_t is the ambiguity set describing possible distributions of P_t .

4. Transition Function

The renewable producer's state evolves as new information (e.g., weather updates) arrives:

$$s_{t+1}^R = f^R(s_t^R, a_t^R, \omega_{t+1}),$$

where ω_{t+1} includes new weather data or forecast updates.

If hours are independent, $s_{t+1}^R = s_t^R$.

5. Objective Functions

The renewable producer can have **multiple objectives**, often expressed separately:

(a) Profit Maximization

$$\max_{(b_t, p_t)} \mathbb{E}_{D_t, (b_{it}, p_{it}), P_t} \left[\sum_{t=1}^{24} \mathbf{1}\{p_t \leq c_t\} (c_t b_t - q_u(b_t - P_t)^+ - q_o(P_t - b_t)^+) \right]$$

subject to $P_t \in \text{supp}(\mathcal{P}_t)$.

(b) Reliability / Risk Minimization

$$\min_{(b_t, p_t)} \mathbb{E}_{P_t \in \mathcal{P}_t} [(b_t - P_t)^+]$$

or equivalently, minimize the expected shortfall or a worst-case penalty:

$$\min_{(b_t, p_t)} \max_{\mathbb{P}_t \in \mathcal{P}_t} \mathbb{E}_{P_t \sim \mathbb{P}_t} [(b_t - P_t)^+]$$

These can be treated as separate objectives (profit vs. reliability).

6. Interaction with Environment

- **Direct influence:** the renewable's (b_t, p_t) affects c_t through the merit order.
- **Indirect feedback:** c_t affects whether the bid is accepted and thus the realized profit.

Perspective 2: The System Regulator

1. Role

The **system regulator (or market operator / policymaker)** sets market rules and penalties to:

- maintain supply–demand balance,
- ensure reliability,
- promote renewable integration and market efficiency.

The regulator does not directly choose bids but may control **policy levers** such as penalty rates or market design parameters.

2. Control Variables (decisions)

Variable	Description
$q_u \geq 0$	penalty rate for under-delivery
$q_o \geq 0$	penalty rate for over-generation (curtailment)
R_t	possible renewable support policy variable (e.g., feed-in premium, cap)

Collectively $a_t^S = (q_u, q_o, R_t)$.

These parameters influence producers' incentives and the endogenous market equilibrium.

3. State Variables

$s_t^S = (\text{aggregate demand forecasts } (\mu_D, \sigma_D), \text{ generation portfolio stats}, \text{market reliability indicators})$

Example reliability indicators:

- Expected imbalance magnitude $E[|D_t - \sum_g x_{gt}|]$
- Renewable curtailment frequency

4. Transition Function

The regulator's environment evolves as a function of market outcomes and policy decisions:

$$s_{t+1}^S = f^S(s_t^S, a_t^S, \omega_t)$$

where ω_t represents observed imbalances, renewable outputs, and price volatility.

5. Objective Functions

The regulator typically has **multiple (possibly conflicting) objectives**.

(a) Market Efficiency

Minimize the expected total system cost (energy payments + imbalance penalties):

$$\min_{a_t^S} \mathbb{E} \left[\sum_{t=1}^{24} \left(\sum_{g \in G} c_t x_{gt} + q_u(b_t - P_t)^+ + q_o(P_t - b_t)^+ \right) \right]$$

(b) Reliability / Security

Minimize the probability or magnitude of system imbalance:

$$\min_{a_t^S} \mathbb{E} \left[|D_t - \sum_{g \in G} x_{gt}| \right]$$

(c) Renewable Integration / Fairness

Maximize the share of renewable generation in total supply or reduce curtailment:

$$\max_{a_t^S} \mathbb{E} \left[\frac{x_{Rt}}{\sum_{g \in G} x_{gt}} \right]$$

Each is a distinct objective; the regulator may treat them via multi-criteria decision analysis rather than a single weighted sum.

6. Interaction with Environment

- The regulator's actions (q_u, q_o) feed into the producer's **profit function** and hence its optimal bidding strategy.
- The producer's actions (b_t, p_t) affect realized imbalances and market reliability metrics, feeding back into s_t^S .
- Both agents influence the shared variable c_t indirectly through their effects on the merit order and system imbalance.

Coupled System Summary

Layer	Controlled by	Variables	Influences
Renewable producer	(b_t, p_t)	bids, accepted quantities, profits	affects market price c_t , imbalance

Layer	Controlled by	Variables	Influences
Conventional producers	stochastic	normal-distributed bids	affect c_t
Regulator	(q_u, q_o, R_t)	penalty parameters, policies	affect renewable incentives & market equilibrium
Environment	shared	$(D_t, c_t, P_t, (b_{it}, p_{it}))$	mediates interactions and evolves over time

Conceptual View

This system can be viewed as a **two-agent stochastic game** (or **bi-level robust optimization**):

- **Lower level (renewable producer):**

$$\max_{(b_t, p_t)} \mathbb{E}_{\text{market}, P_t} [\pi_t(b_t, p_t; q_u, q_o)]$$

- **Upper level (regulator):**

$$\min_{(q_u, q_o)} \mathbb{E}_{\text{market}} [\text{system imbalance or cost}(b_t^*(q_u, q_o), p_t^*(q_u, q_o))]$$

Coupled through the **shared market-clearing mechanism** and the **endogenous clearing price c_t** .

Summary Table of Model Components

Component	Renewable Producer	System Regulator	Shared Environment
Control Variables	b_t, p_t	q_u, q_o, R_t	—
State Variables	\mathcal{P}_t , forecasts	demand, reliability indicators	$D_t, c_t, P_t, (b_{it}, p_{it})$
Transition	$s_{t+1}^R = f^R(s_t^R, a_t^R, \omega_t)$	$s_{t+1}^S = f^S(s_t^S, a_t^S, \omega_t)$	stochastic evolution of D_t, P_t
Objectives	maximize profit; minimize imbalance risk	minimize total cost; minimize imbalance; maximize renewables	—
Interaction	influences c_t via bids	sets penalties influencing producer's incentives	c_t links both sides

Would you like me to **formalize this two-agent framework as an explicit bi-level optimization model** (with equilibrium conditions and KKT constraints for the producer's lower level), or as a **stochastic game / dynamic system diagram** suitable for simulation or RL modeling?



Your specification should only include implementable and quantifiable components. For those conceptual components, please refine them or remove them.

Excellent — that's exactly the right next step. Let's now refine the previous multi-agent model into a **fully implementable, quantifiable specification**, meaning that every component can be operationalized with data or simulation.

We'll keep the **two perspectives** — renewable producer and system regulator — but we will now restrict the model to include only variables, parameters, and transition relations that can be measured, estimated, or computed.

All conceptual placeholders (like “reliability indicators,” “beliefs,” or “fairness objectives”) are replaced with explicitly quantifiable constructs.

Common Quantifiable Environment

The **environment** represents the **electricity market clearing mechanism** and the **stochastic processes** governing demand, conventional bids, and renewable generation.

1. Global Parameters

Symbol	Description	Units / Domain
μ_D, σ_D	Mean and std. deviation of hourly electricity demand	MWh
μ_{ib}, σ_{ib}	Mean and std. deviation of bid quantity for conventional producer i	MWh
μ_{ip}, σ_{ip}	Mean and std. deviation of offer price for conventional producer i	€/MWh
q_u, q_o	Penalty rates for under- and over-delivery (set by regulator)	€/MWh
\bar{P}_t	Maximum renewable generation potential (based on capacity)	MWh
N	Number of Monte Carlo samples (for simulation / estimation)	integer

2. Global Random Variables

Symbol	Distribution	Source / Implementation
$D_t \sim \mathcal{N}(\mu_D, \sigma_D^2)$	Normally distributed hourly demand	historical demand data
$b_{it} \sim \mathcal{N}(\mu_{ib}, \sigma_{ib}^2)$	Bid quantity for each conventional producer i	statistical model of generators
$p_{it} \sim \mathcal{N}(\mu_{ip}, \sigma_{ip}^2)$	Offer price for each conventional producer i	empirical bid data
$P_t \in [0, \bar{P}_t]$	Renewable generation, treated as uncertain scenario variable	scenario samples or forecast bounds

3. Market Clearing Mechanism (Endogenous and Computable)

For each hour t :

1. Collect bids from all producers:

$$\mathcal{B}_t = \{(p_{it}, b_{it})_{i \in \{1,2,3\}}, (p_t, b_t)\}.$$

2. Sort offers by price (ascending).

3. Compute cumulative supply $S_t(p)$:

$$S_t(p) = \sum_{i \in \mathcal{I}} b_{it} \mathbf{1}\{p_{it} \leq p\} + b_t \mathbf{1}\{p_t \leq p\}.$$

4. The **market-clearing price** is:

$$c_t = \min\{p : S_t(p) \geq D_t\}.$$

5. The accepted bid for the renewable is:

$$x_t = b_t \mathbf{1}\{p_t \leq c_t\}.$$

This can be directly implemented in simulation or optimization as a merit-order clearing function.

Perspective 1: Renewable Producer

1. Control Variables (Decision Inputs)

Symbol	Meaning	Type
b_t	Bid quantity (MWh)	continuous, nonnegative
p_t	Offer price (€/MWh)	continuous, nonnegative

Collectively $a_t^R = (b_t, p_t)$.

2. State Variables (Observable Inputs to Decision)

At bidding time for hour t :

$$s_t^R = (\bar{P}_t, \text{forecast range of } P_t, \mu_D, \sigma_D, \{\mu_{ib}, \sigma_{ib}, \mu_{ip}, \sigma_{ip}\}_{i \in \mathcal{I}}).$$

All are **quantifiable**, from forecasts and public statistics.

3. Transition Function

If hourly decisions are independent (standard in day-ahead markets):

$$s_{t+1}^R = s_t^R.$$

Otherwise, forecasts update deterministically or via an empirical update rule based on weather data:

$$s_{t+1}^R = f^R(s_t^R) = s_t^R + \Delta(\text{forecast update}).$$

4. Profit Function per Hour (Quantifiable)

If accepted ($p_t \leq c_t$):

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

If not accepted ($p_t > c_t$), $\pi_t = 0$.

All variables are measurable in €/hour.

5. Objective Functions

Two possible quantifiable objectives (multi-objective setup):

(a) Expected Profit Maximization

$$\max_{b_t, p_t} \mathbb{E}_{D_t, (b_{it}, p_{it}), P_t} [\pi_t(b_t, p_t)].$$

(b) Risk Minimization (Under-delivery Penalty)

$$\min_{b_t, p_t} \mathbb{E}_{P_t} [(b_t - P_t)^+].$$

These can be estimated empirically using Monte Carlo samples of $(D_t, (b_{it}, p_{it}), P_t)$.

Perspective 2: System Regulator

1. Control Variables

Symbol	Meaning	Type
q_u	Under-delivery penalty rate	continuous, ≥ 0
q_o	Over-generation penalty rate	continuous, ≥ 0

Both are **policy levers** chosen by the regulator.

2. State Variables

At time of policy setting:

$$s_t^S = (\mu_D, \sigma_D, \{\mu_{ib}, \sigma_{ib}, \mu_{ip}, \sigma_{ip}\}, \bar{P}_t)$$

These are quantifiable and fixed for a market horizon.

3. Transition Function

The environment evolves stochastically as demand and bids realize:

$$s_{t+1}^S = s_t^S$$

for a static daily market.

(If dynamic learning is included, the transition is based on observed imbalances:

$$s_{t+1}^S = s_t^S + \Delta(\text{observed imbalance statistics}).$$

4. Quantifiable Outcome Metrics

Given (q_u, q_o) and producer bids (b_t, p_t) , the regulator observes:

Metric	Formula	Description
Total system imbalance	$I_t = D_t - \sum_{g \in G} x_{gt}$	
Expected imbalance	$\mathbb{E}[I_t]$	reliability measure
Total system cost	$C_t = \sum_{g \in G} c_t x_{gt} + q_u(b_t - P_t)^+ + q_o(P_t - b_t)^+$	€/hour
Renewable share	$RSH_t = \frac{x_t}{\sum_{g \in G} x_{gt}}$	proportion of renewables in total supply

All are measurable quantities.

5. Objective Functions

Regulator may have **multiple quantifiable objectives**:

(a) System Cost Minimization

$$\min_{q_u, q_o} \mathbb{E}[C_t].$$

(b) Reliability Improvement

$$\min_{q_u, q_o} \mathbb{E}[I_t].$$

(c) Renewable Integration

$$\max_{q_u, q_o} \mathbb{E}[RSH_t].$$

All objectives are estimable by simulation of market outcomes under candidate (q_u, q_o) .

Coupled Market System

The environment connects both stakeholders:

Component	Determined by	Observable / Quantifiable
c_t	Market clearing (function of all bids, demand)	computed value
x_t	Renewable accepted quantity	computed from bids and c_t
P_t	Realized renewable generation	observed (MWh)
π_t	Renewable profit	computed
C_t, I_t, RSH_t	System-level metrics	computed from market data

Interaction Mechanism:

1. Regulator sets (q_u, q_o) .
2. Renewable chooses (b_t, p_t) to optimize its own objectives.
3. Market clears \rightarrow determines c_t, x_t .
4. Outcomes (π_t, C_t, I_t, RSH_t) are realized and recorded.

✓ Compact Implementable Model Summary

Category	Renewable Producer	System Regulator
Decision Variables	b_t, p_t	q_u, q_o
State Variables	$(\bar{P}_t, \mu_D, \sigma_D, \mu_{ib}, \sigma_{ib}, \mu_{ip}, \sigma_{ip})$	$(\mu_D, \sigma_D, \mu_{ib}, \sigma_{ib}, \mu_{ip}, \sigma_{ip}, \bar{P}_t)$
Transition	static or forecast updates	static (policy fixed for horizon)
Objectives	(1) Maximize $\mathbb{E}[\pi_t]$; (2) Minimize $\mathbb{E}[(b_t - P_t)^+]$	(1) Minimize $\mathbb{E}[C_t]$; (2) Minimize $\mathbb{E}[I_t]$; (3) Maximize $\mathbb{E}[RSH_t]$
Shared Variables	$D_t, P_t, (b_{it}, p_{it}), c_t, x_t$	common environment
Market Rule	$c_t = \min\{p : S_t(p) \geq D_t\}$	deterministic and computable

This refined specification is **fully implementable and quantifiable**:

- All stochastic components are explicitly parameterized.
- All objectives and variables are measurable in €/MWh or MWh.
- Interactions between agents are mediated by the market-clearing rule, not by conceptual constructs.

Would you like me to now write this as an **explicit system of equations (bi-level or simulation-ready model)** — e.g., suitable for numerical implementation in Pyomo, GAMS, or Python?



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'll **compose the two perspectives** (renewable producer and system regulator) into **one unified, internally consistent model**, built on a single shared environment and state space.

This will be a **modular multi-agent system model**, designed so that:

- Each agent (renewable producer and regulator) has its **own control variables and objectives**,
- Both operate in a **shared, quantifiable environment**,
- Each agent treats the other's controls as **exogenous** when optimizing,
- The entire system can be simulated or optimized jointly or separately.

Unified Model of the Renewable Producer-Regulator System

1. Common Environment

The **shared environment** defines the physical and market context in which all agents act. It encapsulates the **market-clearing mechanism, stochastic demand and supply processes**, and **global parameters**.

1.1 Sets

- $t \in \mathcal{T} = \{1, \dots, 24\}$: hourly intervals
- $i \in \mathcal{I} = \{1, 2, 3\}$: conventional producers

1.2 Global Parameters

Symbol	Description	Units / Notes
μ_D, σ_D	Mean and standard deviation of hourly demand	MWh
μ_{ib}, σ_{ib}	Mean and standard deviation of bid quantities for conventional producer i	MWh
μ_{ip}, σ_{ip}	Mean and standard deviation of offer prices for conventional producer i	€/MWh
\bar{P}_t	Maximum renewable generation capacity	MWh
N	Number of simulation scenarios	integer

All parameters are measurable or estimable from market and weather data.

1.3 Global Random Variables

Symbol	Description	Distribution / Range
D_t	Market demand	$D_t \sim \mathcal{N}(\mu_D, \sigma_D^2)$
b_{it}, p_{it}	Conventional producer i 's bid quantity and price	$b_{it} \sim \mathcal{N}(\mu_{ib}, \sigma_{ib}^2), p_{it} \sim \mathcal{N}(\mu_{ip}, \sigma_{ip}^2)$

Symbol	Description	Distribution / Range
P_t	Renewable generation realization	unknown distribution, sampled within $[0, \bar{P}_t]$

1.4 Market-Clearing Mechanism (Deterministic Function)

For each hour t :

1. Collect all bids:

$$\mathcal{B}_t = \{(p_{it}, b_{it})_{i \in \mathcal{I}}, (p_t, b_t)\}.$$

2. Sort by price ascending and compute cumulative supply:

$$S_t(p) = \sum_{i \in \mathcal{I}} b_{it} \mathbf{1}\{p_{it} \leq p\} + b_t \mathbf{1}\{p_t \leq p\}.$$

3. Market-clearing price:

$$c_t = \min\{p : S_t(p) \geq D_t\}.$$

4. Accepted renewable quantity:

$$x_t = b_t \mathbf{1}\{p_t \leq c_t\}.$$

These equations define the shared market environment used by both agents.

1.5 Shared State Vector

Define the **common environment state** for each hour t :

$$s_t = (D_t, \{b_{it}, p_{it}\}_{i \in \mathcal{I}}, P_t, c_t, x_t, \mu_D, \sigma_D, \{\mu_{ib}, \sigma_{ib}, \mu_{ip}, \sigma_{ip}\}, \bar{P}_t, q_u, q_o)$$

All components are either stochastic realizations or fixed parameters.

This **common state** is observed by both agents (possibly partially for privacy reasons in applications).

☀️ Agent 1: Renewable Producer Submodel

2.1 Control Variables

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

Bid quantity and price (continuous, nonnegative).

2.2 State Variables

Uses the shared state s_t as input; for implementation, only a subset is required:

$$s_t^R = (\mu_D, \sigma_D, \{\mu_{ib}, \sigma_{ib}, \mu_{ip}, \sigma_{ip}\}, \bar{P}_t, q_u, q_o)$$

2.3 Transition Function

If decisions are independent across hours (typical day-ahead setting):

$$s_{t+1}^R = s_t^R$$

Else, updates via forecast rules if sequential dependency is modeled.

2.4 Profit Function

For each hour t :

$$\pi_t = \begin{cases} c_t b_t - q_u(b_t - P_t)^+ - q_o(P_t - b_t)^+, & \text{if } p_t \leq c_t, \\ 0, & \text{otherwise.} \end{cases}$$

2.5 Objective Functions

Two measurable objectives:

(1) Expected Profit Maximization

$$\max_{b_t, p_t} \mathbb{E}_{D_t, (b_{it}, p_{it}), P_t} [\pi_t].$$

(2) Expected Under-Delivery Minimization

$$\min_{b_t, p_t} \mathbb{E}_{P_t} [(b_t - P_t)^+].$$

2.6 Inputs from Other Perspectives

- q_u, q_o : regulator's control variables (treated as exogenous parameters).
- Market-clearing price c_t : endogenous but depends on exogenous stochastic bids and (b_t, p_t) .

Thus, renewable producer solves:

$$(b_t^*, p_t^*) = \arg \max_{b_t, p_t} \mathbb{E}[\pi_t(b_t, p_t; q_u, q_o)].$$

Agent 2: System Regulator Submodel

3.1 Control Variables

$$a_t^S = (q_u, q_o)$$

Penalty rates, continuous and nonnegative.

3.2 State Variables

Uses the same shared environment state s_t , typically summarized as:

$$s_t^S = (\mu_D, \sigma_D, \{\mu_{ib}, \sigma_{ib}, \mu_{ip}, \sigma_{ip}\}, \bar{P}_t)$$

All quantifiable from data.

3.3 Transition Function

For a static day-ahead horizon:

$$s_{t+1}^S = s_t^S.$$

If regulatory updates occur (e.g., learning over days), transitions depend on observed imbalance statistics.

3.4 System Outcome Metrics (Functions of Shared Environment)

Given all bids, prices, and realizations:

Symbol	Formula	Units
System imbalance	$I_t =$	$D_t - \sum_{g \in G} x_g$
Expected imbalance	$\mathbb{E}[I_t]$	MWh
Total system cost	$C_t = \sum_{g \in G} c_t x_g + q_u(b_t - P_t)^+ + q_o(P_t - b_t)^+$	€
Renewable share	$RSH_t = \frac{x_t}{\sum_{g \in G} x_g}$	dimensionless

All are directly computable from simulation outcomes.

3.5 Objective Functions

Regulator has three quantifiable objectives:

(1) Minimize Expected System Cost

$$\min_{q_u, q_o} \mathbb{E}[C_t].$$

(2) Minimize Expected Imbalance

$$\min_{q_u, q_o} \mathbb{E}[I_t].$$

(3) Maximize Renewable Share

$$\max_{q_u, q_o} \mathbb{E}[RSH_t].$$

3.6 Inputs from Other Perspectives

- b_t, p_t : renewable producer's bid decisions (treated as exogenous from regulator's optimization perspective).

Thus, regulator solves:

$$(q_u^*, q_o^*) = \arg \min_{q_u, q_o} \mathbb{E}[C_t(q_u, q_o; b_t, p_t)].$$

or optimizes any subset of the objectives separately.

4. Composed System Dynamics

4.1 State Evolution

The **joint system state** combines environment and both agents' decisions:

$$S_t = (s_t, a_t^R, a_t^S).$$

Transitions across hours are governed by:

$$S_{t+1} = (f_{\text{env}}(S_t), a_{t+1}^R, a_{t+1}^S),$$

where f_{env} includes new stochastic draws of D_t, b_{it}, p_{it}, P_t , and recomputation of c_t .

4.2 Execution Order per Hour (Operational Sequence)

1. **Regulator sets penalties:** $a_t^S = (q_u, q_o)$.
2. **Renewable producer chooses bid:** $a_t^R = (b_t, p_t)$.
3. **Conventional producers' bids** are realized from their normal distributions.
4. **Market clears:** compute c_t, x_t using merit-order rule.
5. **Renewable generation** P_t is realized.
6. **Profits and system metrics** (π_t, C_t, I_t, RSH_t) are computed.
7. Update expected objectives for both agents.

4.3 Modular Evaluation

- **Renewable module:** Given exogenous q_u, q_o , simulate or optimize b_t, p_t to evaluate $E[\pi_t]$ and $E[(b_t - P_t)^+]$.
- **Regulator module:** Given exogenous b_t, p_t , evaluate $E[C_t], E[I_t]$, and $E[RSH_t]$.
- **Joint simulation:** Iterate between modules to analyze equilibrium or sensitivity.

Each perspective is plug-compatible with the shared environment and can be evaluated separately or together.

✓ 5. Unified Model Summary (Fully Quantifiable)

Layer	Components	Description
Shared Environment	$D_t, P_t, (b_{it}, p_{it}), c_t, x_t$; merit-order rule	Defines stochastic market and equilibrium price
Global Parameters	$\mu_D, \sigma_D, \mu_{ib}, \sigma_{ib}, \mu_{ip}, \sigma_{ip}, \bar{P}_t, q_u, q_o$	Data-driven quantities
Renewable Controls	(b_t, p_t)	Hourly bid decisions
Regulator Controls	(q_u, q_o)	Penalty policy
Renewable Objectives	$\max E[\pi_t], \min E[(b_t - P_t)^+]$	Profit & risk
Regulator Objectives	$\min E[C_t], \min E[I_t], \max E[RSH_t]$	Cost, reliability, renewable integration
Coupling Mechanism	$c_t = f(D_t, (b_{it}, p_{it}), b_t, p_t)$	Market clearing
Transition	$S_{t+1} = f(S_t, \omega_t)$	New random draws and updates

🔧 6. Implementation Notes

- The unified model can be **simulated** hour-by-hour using Monte Carlo draws of D_t, b_{it}, p_{it}, P_t .
- It can be **optimized** from each agent's perspective by treating the other's controls as fixed.
- It can also support **bi-level or iterative equilibrium analysis** if desired:
 $(b_t^*, p_t^*, q_u^*, q_o^*)$ such that each agent is optimal given the other's controls.

In summary, this unified, quantifiable model:

- Shares a **single environment and state space**,
- Embeds both agents' **controls and objectives**,
- Allows **modular, independent evaluation** of each agent,
- Ensures **internal consistency** through a common, implementable market-clearing process.



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 **clean, modular Python implementation** of the **unified renewable producer-regulator model**, organized using classes to represent each perspective and the shared environment.

At the end, a **single interface function** (`simulate_market`) accepts dictionaries of control variables and uncertain parameters, simulates the market clearing and outcome metrics, and returns the **objective function values** for both the renewable producer and the regulator.

python

```

import numpy as np

# =====
# SHARED ENVIRONMENT: MarketClearingEnvironment
# =====

class MarketClearingEnvironment:
    """
    Represents the shared electricity market environment.
    Handles stochastic demand, competitor bids, renewable output,
    and market-clearing price computation.
    """

    def __init__(self, params):
        """
        params: dict of global parameters
        {
            'mu_D', 'sigma_D',
            'mu_ib', 'sigma_ib',      # list of length 3 for conventional producers
            'mu_ip', 'sigma_ip',
            'bar_P',                  # max renewable generation
        }
        """
        self.params = params
        self.num_conv = 3

    def sample_uncertainties(self, uncertain):
        """
        Draw or assign uncertain quantities for a single simulation run.
        uncertain: dict, may override default stochastic draws
        e.g. {'D_t': value, 'P_t': value}
        """
        p = self.params

        # Demand
        D_t = uncertain.get('D_t', np.random.normal(p['mu_D'], p['sigma_D']))

        # Conventional producers' bids
        b_i = [
            np.random.normal(p['mu_ib'][i], p['sigma_ib'][i])
            for i in range(self.num_conv)
        ]
        p_i = [
            np.random.normal(p['mu_ip'][i], p['sigma_ip'][i])
            for i in range(self.num_conv)
        ]

        # Renewable generation realization (within physical range)
        P_t = uncertain.get('P_t', np.random.uniform(0, p['bar_P']))

        return D_t, b_i, p_i, P_t

    def market_clearing(self, D_t, b_i, p_i, b_t, p_t):
        """
        Compute the market-clearing price and accepted quantities using merit order.
        """
        # Combine all bids (3 conventional + 1 renewable)
        bids = [(p_i[j], b_i[j]) for j in range(self.num_conv)] + [(p_t, b_t)]
        # Sort bids by price (ascending)
        bids_sorted = sorted(bids, key=lambda x: x[0])

        supply_accum = 0.0
        c_t = bids_sorted[-1][0] # default if demand exceeds total supply
        for price, qty in bids_sorted:
            supply_accum += qty
            if supply_accum >= D_t:
                c_t = price
                break

        # Determine which bids are accepted

```

```

accepted = [(price, qty) for price, qty in bids_sorted if price <= c_t]
accepted_quantities = [qty for price, qty in accepted]

# Accepted renewable quantity
accepted_renewable = b_t if p_t <= c_t else 0.0

return c_t, accepted_quantities, accepted_renewable

# =====
# AGENT 1: Renewable Producer
# =====

class RenewableProducer:
    """
    Renewable producer agent that chooses (b_t, p_t)
    and computes its expected profit and risk given environment outcomes.
    """

    def __init__(self, q_u, q_o):
        self.q_u = q_u # penalty rates, from regulator

    def compute_profit(self, c_t, b_t, p_t, P_t):
        """
        Compute profit for the renewable producer in one scenario.
        """
        if p_t > c_t:
            return 0.0 # bid not accepted

        q_u, q_o = self.q_u
        shortfall = max(0.0, b_t - P_t)
        surplus = max(0.0, P_t - b_t)
        profit = c_t * b_t - q_u * shortfall - q_o * surplus
        return profit

    def compute_expected_metrics(self, samples):
        """
        Compute expected profit and expected under-delivery over samples.
        samples: list of dicts, each containing {'c_t', 'b_t', 'p_t', 'P_t'}
        """
        profits = []
        underdeliveries = []
        for s in samples:
            c_t, b_t, p_t, P_t = s['c_t'], s['b_t'], s['p_t'], s['P_t']
            profits.append(self.compute_profit(c_t, b_t, p_t, P_t))
            underdeliveries.append(max(0.0, b_t - P_t))
        return {
            'expected_profit': np.mean(profits),
            'expected_under_delivery': np.mean(underdeliveries)
        }

# =====
# AGENT 2: System Regulator
# =====

class SystemRegulator:
    """
    Regulator agent controlling penalty rates (q_u, q_o)
    and evaluating system-level objectives.
    """

    def __init__(self):
        pass

    def compute_system_metrics(self, c_t, D_t, accepted_quantities, b_t, P_t, q_u, q_o):
        """
        Compute quantifiable system outcomes for one scenario.
        """
        total_supply = sum(accepted_quantities)
        imbalance = abs(D_t - total_supply)
        system_cost = sum([c_t * qty for qty in accepted_quantities])
        # Add imbalance penalties
        system_cost += q_u * max(0.0, b_t - P_t) + q_o * max(0.0, P_t - b_t)
        renewable_share = (b_t if p_t <= c_t else 0.0) / max(total_supply, 1e-6)

```

```

        return system_cost, imbalance, renewable_share

    def compute_expected_metrics(self, samples):
        """
        Compute expected system-level objectives across scenarios.
        samples: list of dicts, each containing computed metrics
        """
        cost = np.mean([s['system_cost'] for s in samples])
        imbalance = np.mean([s['imbalance'] for s in samples])
        renewable_share = np.mean([s['renewable_share'] for s in samples])
        return {
            'expected_system_cost': cost,
            'expected_imbalance': imbalance,
            'expected_renewable_share': renewable_share
        }

# =====
# UNIFIED MODEL INTERFACE
# =====

def simulate_market(controls, uncertain_params, n_samples=1000):
    """
    Interface function for unified renewable-regulator model simulation.

    Parameters
    -----
    controls : dict
        {
            'renewable': {'b_t': value, 'p_t': value},
            'regulator': {'q_u': value, 'q_o': value}
        }

    uncertain_params : dict
        Global stochastic and parameter values, e.g.:
        {
            'mu_D', 'sigma_D',
            'mu_ib', 'sigma_ib',
            'mu_ip', 'sigma_ip',
            'bar_P'
        }

    n_samples : int
        Number of Monte Carlo simulations.

    Returns
    -----
    results : dict
        {
            'renewable': {
                'expected_profit': ...,
                'expected_under_delivery': ...
            },
            'regulator': {
                'expected_system_cost': ...,
                'expected_imbalance': ...,
                'expected_renewable_share': ...
            }
        }
    """

# Initialize shared environment
env = MarketClearingEnvironment(uncertain_params)

# Controls
b_t = controls['renewable']['b_t']
p_t = controls['renewable']['p_t']
q_u = controls['regulator']['q_u']
q_o = controls['regulator']['q_o']

# Initialize agents
renewable = RenewableProducer((q_u, q_o))
regulator = SystemRegulator()

# Storage for scenario outcomes

```

```

renewable_samples = []
regulator_samples = []

for _ in range(n_samples):
    # Sample stochastic environment
    D_t, b_i, p_i, P_t = env.sample_uncertainties({})

    # Market clearing
    c_t, accepted_quantities, accepted_renewable = env.market_clearing(D_t, b_i, p_i, b_t,
p_t)

    # Renewable outcomes
    renewable_samples.append({
        'c_t': c_t,
        'b_t': b_t,
        'p_t': p_t,
        'P_t': P_t
    })

    # Regulator outcomes
    system_cost, imbalance, renewable_share = regulator.compute_system_metrics(
        c_t, D_t, accepted_quantities, b_t, P_t, q_u, q_o
    )
    regulator_samples.append({
        'system_cost': system_cost,
        'imbalance': imbalance,
        'renewable_share': renewable_share
    })

# Compute expected objectives
renewable_results = renewable.compute_expected_metrics(renewable_samples)
regulator_results = regulator.compute_expected_metrics(regulator_samples)

return {
    'renewable': renewable_results,
    'regulator': regulator_results
}

```

✓ Explanation

- **MarketClearingEnvironment**

Defines shared stochastic processes and market-clearing logic (demand, bids, price determination).

- **RenewableProducer**

Takes the regulator's penalty rates as input and computes its profit and under-delivery risk over Monte Carlo scenarios.

- **SystemRegulator**

Evaluates system-wide outcomes (cost, imbalance, renewable share) given all agents' actions and random realizations.

- **simulate_market()**

Acts as the **interface function**:

- Inputs: dictionaries for **controls** and **uncertain parameters**.
- Simulates the joint market environment across many samples.
- Returns the expected values of **each agent's objectives**.

This implementation is **ready for simulation or extension**:

- Each module is fully quantifiable (no conceptual placeholders).

- Each perspective's controls are independent.
- The shared environment ensures internal consistency and coupling.



`mu_D, sigma_D, mu_ib, sigma_ib, mu_ip, sigma_ip` are constants instead of uncertain parameters, so the implementation should not take them as inputs. Please refine the implementation.

Excellent — here's the **refined, fully consistent Python implementation** of the unified renewable-regulator market model.

In this version:

- The **global constants** (`mu_D, sigma_D, mu_ib, sigma_ib, mu_ip, sigma_ip`) are fixed within the model (not user inputs).
- The **interface function** `simulate_market()` now only accepts:
 - A dictionary of **control variables** (from the renewable producer and regulator), and
 - A dictionary of **uncertain values** (`D_t, P_t`, etc.) if the user wants to override stochastic draws.

All other constants are embedded as internal model parameters.

python

```
import numpy as np

# =====
# SHARED ENVIRONMENT
# =====

class MarketClearingEnvironment:
    """
    Shared electricity market environment.
    Contains constants for demand and conventional producers,
    handles stochastic sampling and market clearing.
    """

    def __init__(self, bar_P):
        """
        bar_P : float
            Maximum renewable generation (MWh)
        """
        # Fixed (global) constants for this implementation
        self.num_conv = 3
        self.mu_D = 1000.0
        self.sigma_D = 100.0

        # Conventional producers' bid parameters (example realistic values)
        self.mu_ib = [300.0, 350.0, 400.0]    # MWh
        self.sigma_ib = [30.0, 35.0, 40.0]
        self.mu_ip = [40.0, 50.0, 60.0]        # €/MWh
        self.sigma_ip = [4.0, 5.0, 6.0]

        self.bar_P = bar_P

    def sample_uncertainties(self, uncertain):
        """
        Draw stochastic realizations for one simulation.
        uncertain: dict to override random draws, e.g. {'D_t': value, 'P_t': value}
        """

```

```

# Demand
D_t = uncertain.get('D_t', np.random.normal(self.mu_D, self.sigma_D))

# Conventional producers' bids
b_i = [
    np.random.normal(self.mu_ib[i], self.sigma_ib[i])
    for i in range(self.num_conv)
]
p_i = [
    np.random.normal(self.mu_ip[i], self.sigma_ip[i])
    for i in range(self.num_conv)
]

# Renewable generation realization
P_t = uncertain.get('P_t', np.random.uniform(0, self.bar_P))

return D_t, b_i, p_i, P_t

def market_clearing(self, D_t, b_i, p_i, b_t, p_t):
    """
    Compute market-clearing price and accepted quantities (merit-order).
    """
    bids = [(p_i[j], b_i[j]) for j in range(self.num_conv)] + [(p_t, b_t)]
    bids_sorted = sorted(bids, key=lambda x: x[0])

    supply_accum = 0.0
    c_t = bids_sorted[-1][0] # fallback: last offer
    for price, qty in bids_sorted:
        supply_accum += qty
        if supply_accum >= D_t:
            c_t = price
            break

    accepted = [(price, qty) for price, qty in bids_sorted if price <= c_t]
    accepted_quantities = [qty for price, qty in accepted]

    accepted_renewable = b_t if p_t <= c_t else 0.0
    return c_t, accepted_quantities, accepted_renewable

# =====
# RENEWABLE PRODUCER
# =====

class RenewableProducer:
    """
    Renewable producer choosing (b_t, p_t) and evaluated under penalties.
    """

    def __init__(self, q_u, q_o):
        self.q_u = q_u
        self.q_o = q_o

    def profit(self, c_t, b_t, p_t, P_t):
        """
        Profit for one realization.
        """
        if p_t > c_t:
            return 0.0 # bid not accepted
        shortfall = max(0.0, b_t - P_t)
        surplus = max(0.0, P_t - b_t)
        return c_t * b_t - self.q_u * shortfall - self.q_o * surplus

    def expected_metrics(self, samples):
        """
        Compute expected profit and expected under-delivery.
        """
        profits = []
        underdeliveries = []
        for s in samples:
            profits.append(self.profit(s['c_t'], s['b_t'], s['p_t'], s['P_t']))
            underdeliveries.append(max(0.0, s['b_t'] - s['P_t']))
        return {
            'expected_profit': np.mean(profits),
            'expected_under_delivery': np.mean(underdeliveries)
}

```

```

}

# -----
# SYSTEM REGULATOR
# -----
class SystemRegulator:
    """
    Regulator controlling penalties (q_u, q_o),
    evaluating system-level cost, imbalance, and renewable share.
    """

    def __init__(self):
        pass

    def metrics(self, c_t, D_t, accepted_quantities, b_t, p_t, P_t, q_u, q_o):
        """
        Compute quantifiable system outcomes for one simulation.
        """
        total_supply = sum(accepted_quantities)
        imbalance = abs(D_t - total_supply)
        system_cost = sum([c_t * qty for qty in accepted_quantities])
        system_cost += q_u * max(0.0, b_t - P_t) + q_o * max(0.0, P_t - b_t)
        renewable_share = (b_t if p_t <= c_t else 0.0) / max(total_supply, 1e-6)
        return system_cost, imbalance, renewable_share

    def expected_metrics(self, samples):
        """
        Compute expected values of system metrics.
        """
        cost = np.mean([s['system_cost'] for s in samples])
        imbalance = np.mean([s['imbalance'] for s in samples])
        renewable_share = np.mean([s['renewable_share'] for s in samples])
        return {
            'expected_system_cost': cost,
            'expected_imbalance': imbalance,
            'expected_renewable_share': renewable_share
        }

# -----
# UNIFIED MODEL INTERFACE
# -----
def simulate_market(controls, uncertain_inputs, n_samples=1000):
    """
    Unified interface for simulating the renewable-regulator market model.

    Parameters
    -----
    controls : dict
        {
            'renewable': {'b_t': float, 'p_t': float},
            'regulator': {'q_u': float, 'q_o': float}
        }

    uncertain_inputs : dict
        May include overrides for uncertain variables, e.g.:
        {'D_t': 950.0, 'P_t': 420.0}
        (Otherwise, random sampling is used.)

    n_samples : int
        Number of Monte Carlo simulation runs.

    Returns
    -----
    results : dict
        {
            'renewable': {
                'expected_profit': float,
                'expected_under_delivery': float
            },
            'regulator': {
                'expected_system_cost': float,
    """

```

```

        'expected_imbalance': float,
        'expected_renewable_share': float
    }
}
"""

# Extract control variables
b_t = controls['renewable']['b_t']
p_t = controls['renewable']['p_t']
q_u = controls['regulator']['q_u']
q_o = controls['regulator']['q_o']

# Initialize shared environment and agents
env = MarketClearingEnvironment(bar_P=500.0)
renewable = RenewableProducer(q_u=q_u, q_o=q_o)
regulator = SystemRegulator()

# Scenario results
renewable_samples = []
regulator_samples = []

for _ in range(n_samples):
    # Draw random scenario (or use overrides)
    D_t, b_i, p_i, P_t = env.sample_uncertainties(uncertain_inputs)

    # Market clearing
    c_t, accepted_quantities, accepted_renewable = env.market_clearing(
        D_t, b_i, p_i, b_t, p_t
    )

    # Renewable metrics
    renewable_samples.append({
        'c_t': c_t,
        'b_t': b_t,
        'p_t': p_t,
        'P_t': P_t
    })

    # Regulator metrics
    system_cost, imbalance, renewable_share = regulator.metrics(
        c_t, D_t, accepted_quantities, b_t, p_t, P_t, q_u, q_o
    )
    regulator_samples.append({
        'system_cost': system_cost,
        'imbalance': imbalance,
        'renewable_share': renewable_share
    })

# Aggregate expected results
renewable_results = renewable.expected_metrics(renewable_samples)
regulator_results = regulator.expected_metrics(regulator_samples)

return {
    'renewable': renewable_results,
    'regulator': regulator_results
}

```

✓ Summary of Refinements

Aspect	Change
Global constants	<code>mu_D, sigma_D, mu_ib, sigma_ib, mu_ip, sigma_ip</code> are embedded inside the environment class.
Interface inputs	The interface now only requires control variables (<code>b_t, p_t, q_u, q_o</code>) and optional uncertain overrides (<code>D_t, P_t</code>).
Output	Returns expected objective values for both agents: renewable (<code>profit, under_delivery</code>) and regulator (<code>system_cost, imbalance, renewable_share</code>).

Aspect	Change
Fully quantifiable	All variables and parameters are numerical and computable.

✓ Usage Example (not executed):

```
python

controls = {
    'renewable': {'b_t': 400.0, 'p_t': 45.0},
    'regulator': {'q_u': 15.0, 'q_o': 5.0}
}

uncertain_inputs = {} # use stochastic draws
results = simulate_market(controls, uncertain_inputs, n_samples=500)
```

This code is **self-contained, modular, and internally consistent** — ready for numerical testing, optimization, or policy evaluation.



bar_P is the uncertain parameter, so the implementation should take that as input. Please refine the implementation.

Perfect — here's the **final refined and internally consistent Python implementation** of the unified model.

In this version:

- The **maximum renewable capacity bar_P** is now correctly treated as an **uncertain parameter** — it's passed through the interface.
- All other constants (`mu_D, sigma_D, mu_ib, sigma_ib, mu_ip, sigma_ip`) remain fixed inside the environment.
- The unified model remains modular, fully quantifiable, and simulation-ready.

```
python

import numpy as np

# =====#
# SHARED ENVIRONMENT
# =====#

class MarketClearingEnvironment:
    """
    Shared electricity market environment.
    Contains constants for demand and conventional producers,
    handles stochastic sampling and market clearing.
    """

    def __init__(self, bar_P):
        """
        bar_P : float
            Maximum renewable generation (MWh), treated as uncertain parameter.
        """
        # Fixed (global) constants
```

```

self.num_conv = 3
self.mu_D = 1000.0
self.sigma_D = 100.0

# Conventional producers' bid parameters
self.mu_ib = [300.0, 350.0, 400.0]    # MWh
self.sigma_ib = [30.0, 35.0, 40.0]
self.mu_ip = [40.0, 50.0, 60.0]        # €/MWh
self.sigma_ip = [4.0, 5.0, 6.0]

self.bar_P = bar_P

def sample_uncertainties(self, uncertain):
    """
    Draw stochastic realizations for one simulation.
    uncertain: dict to override random draws, e.g. {'D_t': value, 'P_t': value}
    """
    # Demand
    D_t = uncertain.get('D_t', np.random.normal(self.mu_D, self.sigma_D))

    # Conventional producers' bids
    b_i = [
        np.random.normal(self.mu_ib[i], self.sigma_ib[i])
        for i in range(self.num_conv)
    ]
    p_i = [
        np.random.normal(self.mu_ip[i], self.sigma_ip[i])
        for i in range(self.num_conv)
    ]

    # Renewable generation realization (bounded by uncertain bar_P)
    P_t = uncertain.get('P_t', np.random.uniform(0, self.bar_P))

    return D_t, b_i, p_i, P_t

def market_clearing(self, D_t, b_i, p_i, b_t, p_t):
    """
    Compute market-clearing price and accepted quantities (merit-order).
    """
    bids = [(p_i[j], b_i[j]) for j in range(self.num_conv)] + [(p_t, b_t)]
    bids_sorted = sorted(bids, key=lambda x: x[0])

    supply_accum = 0.0
    c_t = bids_sorted[-1][0]  # fallback: last offer
    for price, qty in bids_sorted:
        supply_accum += qty
        if supply_accum >= D_t:
            c_t = price
            break

    accepted = [(price, qty) for price, qty in bids_sorted if price <= c_t]
    accepted_quantities = [qty for price, qty in accepted]

    accepted_renewable = b_t if p_t <= c_t else 0.0
    return c_t, accepted_quantities, accepted_renewable

# =====
# RENEWABLE PRODUCER
# =====

class RenewableProducer:
    """
    Renewable producer choosing (b_t, p_t) and evaluated under penalties.
    """

    def __init__(self, q_u, q_o):
        self.q_u = q_u
        self.q_o = q_o

    def profit(self, c_t, b_t, p_t, P_t):
        """
        Profit for one realization.
        """
        if p_t > c_t:

```

```

        return 0.0 # bid not accepted
    shortfall = max(0.0, b_t - P_t)
    surplus = max(0.0, P_t - b_t)
    return c_t * b_t - self.q_u * shortfall - self.q_o * surplus

def expected_metrics(self, samples):
    """
    Compute expected profit and expected under-delivery.
    """
    profits = []
    underdeliveries = []
    for s in samples:
        profits.append(self.profit(s['c_t'], s['b_t'], s['p_t'], s['P_t']))
        underdeliveries.append(max(0.0, s['b_t'] - s['P_t']))
    return {
        'expected_profit': np.mean(profits),
        'expected_under_delivery': np.mean(underdeliveries)
    }

# =====
# SYSTEM REGULATOR
# =====

class SystemRegulator:
    """
    Regulator controlling penalties (q_u, q_o),
    evaluating system-level cost, imbalance, and renewable share.
    """

    def __init__(self):
        pass

    def metrics(self, c_t, D_t, accepted_quantities, b_t, p_t, P_t, q_u, q_o):
        """
        Compute quantifiable system outcomes for one simulation.
        """
        total_supply = sum(accepted_quantities)
        imbalance = abs(D_t - total_supply)
        system_cost = sum([c_t * qty for qty in accepted_quantities])
        system_cost += q_u * max(0.0, b_t - P_t) + q_o * max(0.0, P_t - b_t)
        renewable_share = (b_t if p_t <= c_t else 0.0) / max(total_supply, 1e-6)
        return system_cost, imbalance, renewable_share

    def expected_metrics(self, samples):
        """
        Compute expected values of system metrics.
        """
        cost = np.mean([s['system_cost'] for s in samples])
        imbalance = np.mean([s['imbalance'] for s in samples])
        renewable_share = np.mean([s['renewable_share'] for s in samples])
        return {
            'expected_system_cost': cost,
            'expected_imbalance': imbalance,
            'expected_renewable_share': renewable_share
        }

# =====
# UNIFIED MODEL INTERFACE
# =====

def simulate_market(controls, uncertain_inputs, n_samples=1000):
    """
    Unified interface for simulating the renewable-regulator market model.

    Parameters
    -----
    controls : dict
        {
            'renewable': {'b_t': float, 'p_t': float},
            'regulator': {'q_u': float, 'q_o': float}
        }
    uncertain_inputs : dict
    """

```

```

Must include the uncertain renewable capacity:
{'bar_P': float, ...}
Optional: may override other uncertain variables, e.g. {'D_t': value, 'P_t': value}.

n_samples : int
    Number of Monte Carlo simulation runs.

Returns
-----
results : dict
{
    'renewable': {
        'expected_profit': float,
        'expected_under_delivery': float
    },
    'regulator': {
        'expected_system_cost': float,
        'expected_imbalance': float,
        'expected_renewable_share': float
    }
}
"""

# Extract controls
b_t = controls['renewable']['b_t']
p_t = controls['renewable']['p_t']
q_u = controls['regulator']['q_u']
q_o = controls['regulator']['q_o']

# Extract uncertain parameter: bar_P
if 'bar_P' not in uncertain_inputs:
    raise ValueError("uncertain_inputs must include 'bar_P' as an uncertain parameter.")
bar_P = uncertain_inputs['bar_P']

# Initialize shared environment and agents
env = MarketClearingEnvironment(bar_P=bar_P)
renewable = RenewableProducer(q_u=q_u, q_o=q_o)
regulator = SystemRegulator()

# Scenario results
renewable_samples = []
regulator_samples = []

for _ in range(n_samples):
    # Sample stochastic elements
    D_t, b_i, p_i, P_t = env.sample_uncertainties(uncertain_inputs)

    # Market clearing
    c_t, accepted_quantities, accepted_renewable = env.market_clearing(
        D_t, b_i, p_i, b_t, p_t
    )

    # Renewable results
    renewable_samples.append({
        'c_t': c_t,
        'b_t': b_t,
        'p_t': p_t,
        'P_t': P_t
    })

    # Regulator results
    system_cost, imbalance, renewable_share = regulator.metrics(
        c_t, D_t, accepted_quantities, b_t, p_t, P_t, q_u, q_o
    )
    regulator_samples.append({
        'system_cost': system_cost,
        'imbalance': imbalance,
        'renewable_share': renewable_share
    })

# Expected outcomes
renewable_results = renewable.expected_metrics(renewable_samples)
regulator_results = regulator.expected_metrics(regulator_samples)

return {

```

```
'renewable': renewable_results,
'regulator': regulator_results
}
```

✓ Key Refinements

Aspect	Update
bar_P treated as uncertain	Passed via uncertain_inputs and used in MarketClearingEnvironment .
Constants fixed internally	mu_D, sigma_D, mu_ib, sigma_ib, mu_ip, sigma_ip remain model constants (not inputs).
Interface validation	The function checks that bar_P is provided.
Still modular and quantifiable	Each class encapsulates one perspective; all quantities are numeric and simulation-ready.

✓ Example (not executed)

```
python

controls = {
    'renewable': {'b_t': 400.0, 'p_t': 45.0},
    'regulator': {'q_u': 15.0, 'q_o': 5.0}
}

uncertain_inputs = {
    'bar_P': 500.0 # uncertain renewable generation capacity
}

results = simulate_market(controls, uncertain_inputs, n_samples=500)
```

This version is **complete, modular, and fully consistent** — ready for use in simulations, optimization, or policy evaluation studies.