# Experiments 3 - Market_narrative 2

In this problem, a renewable energy producer participates in a day-ahead merit-order electricity market competing against three other conventional producers. The producer submits hourly bids that specify a delivery quantity of electricity and an offer price for each hour of the next day. 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. Meanwhile, the hourly clearing price depends on the bids of all producers and is unknown in advance. The renewable energy producer's goal is to maximize its expected profit by strategically selecting the quantity of electricity \$b_t\$ and offer price \$p_t\$ to bid for each hour of the next day, accounting for the uncertainty in both renewable generation and clearing prices.

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

Excellent — this is a rich stochastic decision-making problem under uncertainty, typical in electricity market bidding for renewables. Let's formalize it carefully and systematically.

---

## 🧮 1. Problem Overview

A renewable energy producer decides **hourly bids** $(b_t, p_t)$ for each hour $t \in \{1, 2, \ldots, T\}$ of the next day (so typically $T = 24$).
Each bid specifies:

- $b_t$: **bid quantity** (MWh)
- $p_t$: **offer price** (€/MWh)

The market then clears, determining the **clearing price** $\pi_t$ (€/MWh).
The renewable producer also realizes **available generation** $g_t$ (MWh) due to weather (wind, solar, etc.).
Both $\pi_t$ and $g_t$ are **stochastic** and unknown when bidding.

If the producer's offer price is **below or equal to** the clearing price ($p_t \leq \pi_t$), the bid is **accepted**, and the producer must deliver $b_t$. If actual generation $g_t$ is less than $b_t$, the producer incurs a shortfall penalty.

---

## ⚙️ 2. Decision Variables

$$b_t \geq 0 \quad \text{and} \quad p_t \geq 0, \quad \text{for each hour } t = 1, \ldots, T$$

---

## 💰 3. Stochastic Elements

- **Renewable generation**: $g_t \sim \text{distribution } G_t$ (depends on weather)
- **Market clearing price**: $\pi_t \sim \text{distribution } \Pi_t$ (depends on competitors' bids and demand)

These are random variables known only through their distributions or scenarios.

---

## 📊 4. Profit Function

For a given realization $(g_t, \pi_t)$, the **profit** in hour $t$ is:

$$\Pi_t^{\text{profit}}(b_t, p_t; g_t, \pi_t) = \begin{cases} \pi_t\, b_t - C_t(b_t, g_t), & \text{if } p_t \le \pi_t \\ 0, & \text{if } p_t > \pi_t \end{cases}$$

where $C_t(b_t, g_t)$ is the **cost or penalty** due to generation shortfall (or imbalance).

Typical form for the penalty:

$$C_t(b_t, g_t) = c_s\,[b_t - g_t]_+,$$

where $[x]_+ = \max(x, 0)$ and $c_s$ is the shortfall penalty (€/MWh).

---

## 🎯 5. Objective Function

The producer's goal is to **maximize expected profit**:

$$\max_{b_t, p_t\ \forall t}\ \mathbb{E}\left[\sum_{t=1}^{T} \Pi_t^{\text{profit}}(b_t, p_t; g_t, \pi_t)\right]$$

subject to possible operational or regulatory constraints (e.g., $0 \le b_t \le \bar{b}_t$).

---

## 🧩 6. Model Components

| Component | Description | Type |
|---|---|---|
| $t$ | Hour index | Discrete time |
| $b_t$ | Bid quantity (decision) | Control variable |
| $p_t$ | Offer price (decision) | Control variable |
| $g_t$ | Renewable generation | Stochastic variable |
| $\pi_t$ | Market clearing price | Stochastic variable |
| $c_s$ | Shortfall penalty rate | Constant parameter |
| $\bar{b}_t$ | Maximum available bid | Constant parameter |
| $r_t$ | Renewable resource state (e.g., wind speed forecast) | State variable |
| $\omega_t$ | Random disturbance (weather, market uncertainty) | Noise variable |

## 🔄 7. State and Transition Functions

If generation depends on a weather-related **state variable** $r_t$ (e.g., wind speed, solar irradiance forecast), the dynamics might be modeled as:

$$r_{t+1} = f(r_t, \omega_t)$$

and available generation $g_t = h(r_t)$.

If each hour is independent, transitions can be omitted, but in a sequential planning setting, the transition captures temporal correlation in weather or market prices.

---

## 🧠 8. Stochastic Programming Formulation

In compact form, this can be written as a **two-stage stochastic optimization problem** for each hour (or a multi-stage extension):

$$\begin{aligned} \max_{\{b_t, p_t\}} \quad & \mathbb{E}_{\pi_t, g_t}\left[\sum_{t=1}^{T} \mathbf{1}_{\{p_t \le \pi_t\}}(\pi_t b_t - c_s[b_t - g_t]_+)\right] \\ \text{s.t.} \quad & 0 \le b_t \le \bar{b}_t, \quad \forall t \\ & p_t \ge 0, \quad \forall t \end{aligned}$$

where $\mathbf{1}_{\{p_t \le \pi_t\}}$ is an indicator of bid acceptance.

---

## 🧱 9. Extensions (optional)

- **Risk aversion:** include Conditional Value at Risk (CVaR) or variance penalty on profit:

$$\max \; \mathbb{E}[\text{profit}] - \lambda \, \text{CVaR}_\alpha(\text{loss})$$

- **Coupled hours:** account for ramping or energy limits over the day:

$$\sum_t b_t \le E_{\max}$$

- **Scenario-based representation:** approximate distributions with scenarios $\{s\}$ and probabilities $p_s$ .

---

✅ **In summary**, the renewable producer's day-ahead bidding problem is a **stochastic optimization problem** with:

- **Decision variables:** $b_t, p_t$
- **Random variables:** $g_t, \pi_t$
- **Objective:** maximize expected profit given uncertainty in generation and clearing prices
- **State dynamics (optional):** $r_{t+1} = f(r_t, \omega_t)$

- **Key constraints:** physical limits, market acceptance condition, shortfall penalties.

---

Would you like me to express this model as a **stochastic dynamic programming (Bellman) formulation** next (to explicitly show the recursive structure)?

> First, the day-ahead market is cleared once per day for 24 hourly intervals of the following day, so you can assume the problem only considers a 24-hour period. Second, penalties would apply if the renewable producer under-delivers or over-delivers relative to its bid. Please refine the specification.

Perfect — that clarification changes the structure slightly but importantly:

- The optimization is **static** (a single decision stage for the next day), not multi-stage, since all 24 hourly bids are submitted **simultaneously** before the market clears.
- There are **two-sided penalties**: one for under-delivery (shortfall) and one for over-delivery (excess generation).

Let's refine the formal model accordingly.

---

## ⚙️ 1. Problem Setting

A renewable energy producer participates in a **day-ahead electricity market** that clears once per day, setting 24 hourly prices for the following day.

The producer submits bids:

$$(b_t, p_t), \quad t = 1, \ldots, 24$$

where

- $b_t$ = bid quantity for hour $t$ (MWh)
- $p_t$ = offer price (€/MWh)

After all producers submit bids, the market operator determines for each hour $t$ the **market clearing price** $\pi_t$ (€/MWh).
The bid is **accepted** if $p_t \leq \pi_t$, in which case the producer must deliver $b_t$ MWh in that hour.

Actual renewable generation $g_t$ (MWh) is uncertain and realized the next day. If $g_t \neq b_t$, imbalance penalties apply.

---

## 💡 2. Model Components

| Symbol | Description | Type |
|---|---|---|
| $t = 1, \ldots, 24$ | Hour of the day | Index |

| Symbol | Description | Type |
|---|---|---|
| $b_t$ | Bid quantity (MWh) | Decision variable |
| $p_t$ | Offer price (€/MWh) | Decision variable |
| $g_t$ | Realized renewable generation (MWh) | Stochastic variable |
| $\pi_t$ | Market clearing price (€/MWh) | Stochastic variable |
| $c_t^{\text{under}}$ | Penalty for under-delivery (€/MWh short) | Parameter |
| $c_t^{\text{over}}$ | Penalty for over-delivery (€/MWh excess) | Parameter |
| $\bar{b}_t$ | Max feasible bid (MWh) | Parameter |
| $f_{G_t,\Pi_t}$ | Joint probability distribution of $(g_t, \pi_t)$ | Data/uncertainty |

All 24 hours are coupled **only through the expectation in the objective** (no inter-hour dynamics, since bids are day-ahead and independent across hours).

## 🎯 3. Profit Function per Hour

For a given realization $(g_t, \pi_t)$:

$$\Pi_t^{\text{profit}}(b_t, p_t; g_t, \pi_t) = \begin{cases} \pi_t b_t - C_t(g_t, b_t), & \text{if } p_t \leq \pi_t \quad (\text{accepted}) \\ 0, & \text{if } p_t > \pi_t \quad (\text{rejected}) \end{cases}$$

The **imbalance penalty** $C_t(g_t, b_t)$ accounts for both over- and under-delivery:

$$C_t(g_t, b_t) = c_t^{\text{under}}[b_t - g_t]_+ + c_t^{\text{over}}[g_t - b_t]_+,$$

where $[x]_+ = \max(x, 0)$.

## 💰 4. Expected Total Profit (Objective)

The producer's expected profit over all 24 hours is:

$$\max_{\{b_t,p_t\}_{t=1}^{24}} \mathbb{E}_{\pi_t,g_t}\left[\sum_{t=1}^{24} \mathbf{1}_{\{p_t \leq \pi_t\}} \left(\pi_t b_t - C_t(g_t, b_t)\right)\right]$$

subject to:

$$0 \leq b_t \leq \bar{b}_t, \quad p_t \geq 0, \quad \forall t = 1, \dots, 24.$$

Here, $\mathbf{1}_{\{p_t \leq \pi_t\}}$ is an indicator function representing whether the bid clears in hour $t$.

## 📊 5. Randomness and Uncertainty

- The **renewable generation** $g_t$ is random due to weather (wind, solar irradiance).
  → Distribution: $g_t \sim G_t(r_t)$ based on forecast information.

- The **market clearing price** $\pi_t$ depends on aggregate market bids and demand.
  → Distribution: $\pi_t \sim \Pi_t$, potentially correlated with $g_t$ (e.g., low prices when renewable output is high).

If represented with scenarios $s \in \mathcal{S}$, with probabilities $p_s$:

$$\max_{b_t, p_t} \sum_{s \in \mathcal{S}} p_s \sum_{t=1}^{24} \mathbf{1}_{\{p_t \le \pi_t^s\}} (\pi_t^s b_t - C_t(g_t^s, b_t)).$$

---

## 🧩 6. States and Structure

Since the day-ahead market is **single-stage**:

- **State variables:** none evolve between hours (static day-ahead problem).
  Optionally, you can treat weather forecast information at bidding time as an **initial state** $r_0$ summarizing uncertainty distributions for $(g_t, \pi_t)$.
- **Transition function:** not applicable (no sequential re-bidding within the day).

---

## 🧱 7. Compact Formulation

$$\begin{aligned} \max_{\{b_t, p_t\}} \quad & \mathbb{E}_{\pi_t, g_t} \left[ \sum_{t=1}^{24} \mathbf{1}_{\{p_t \le \pi_t\}} \left( \pi_t b_t - c_t^{\text{under}} [b_t - g_t]_+ - c_t^{\text{over}} [g_t - b_t]_+ \right) \right] \\ \text{s.t.} \quad & 0 \le b_t \le \bar{b}_t, \quad p_t \ge 0, \quad t = 1, \dots, 24. \end{aligned}$$

---

## 🔍 8. Interpretation

- The problem is a **stochastic optimization under uncertainty**, with the two main uncertainties being $g_t$ (generation) and $\pi_t$ (price).
- Because all decisions $(b_t, p_t)$ are taken before realizations occur, this is a **single-stage expected-profit maximization problem**.
- Each hour can be solved independently **if** the uncertainties are independent across hours; otherwise, joint optimization across hours captures correlations.

---

Would you like me to extend this model to include **risk aversion** (e.g., CVaR-based objective) or **joint probability modeling** of $(g_t, \pi_t)$?

There are still several areas where the model specification can be improved: (1) the renewable energy generation is deeply uncertain, so its probability distribution cannot be determined; and (2) the market-clearing price is not random but endogenous, determined by merit-order of bids. Please refine your specification.

Excellent — that's an important refinement.
Once we acknowledge that (a) renewable generation is **deeply uncertain** (no reliable probability distribution) and (b) the **clearing price is endogenous**, determined by **all producers' bids**, then the problem changes fundamentally:

We now have a **strategic optimization under endogenous market clearing** and **deep uncertainty**, not a simple stochastic program.

Let's rebuild the specification rigorously and explicitly reflect those features.

---

## ⚙️ 1. Problem Context

We consider a **day-ahead electricity market** that clears once per day for 24 hourly intervals.

- There are $N = 4$ producers:
  $i = 1, 2, 3$ are **conventional** generators, and $i = R$ is the **renewable** producer.
- Each producer $i$ submits hourly offers $(b_{i,t}, p_{i,t})$ for $t = 1, \ldots, 24$,
  where:

$$b_{i,t} \geq 0 \text{ (MWh)}, \quad p_{i,t} \geq 0 \text{ (€/MWh)}.$$

---

## 🧩 2. Market Clearing Mechanism (Endogenous Price)

For each hour $t$:

1. The market operator sorts all offers by **ascending offer price** $p_{i,t}$.
2. The **clearing price** $\pi_t$ is the lowest price at which cumulative offered quantity meets demand $D_t$.

Formally,

$$\pi_t = \min \left\{ p \mid \sum_{i: p_{i,t} \leq p} b_{i,t} \geq D_t \right\}.$$

Let $\mathcal{A}_t = \{ i \mid p_{i,t} \leq \pi_t \}$ denote the **accepted producers** for hour $t$.

Each accepted producer $i \in \mathcal{A}_t$ must deliver its bid quantity $b_{i,t}$ at the market price $\pi_t$.

---

## ⚡ 3. Renewable Producer's Physical Uncertainty (Deep Uncertainty)

The renewable producer $R$ faces **deep uncertainty** in its actual generation $g_t$, which depends on weather conditions that cannot be represented by a precise probability distribution.

We instead assume a **bounded uncertainty set**:

$$g_t \in \mathcal{G}_t \subseteq [0, \bar{g}_t],$$

which represents all plausible realizations (e.g., derived from physical forecasts, scenario envelopes, or expert ranges).

## 💰 4. Profit Function (with Two-Sided Imbalance Penalties)

If the renewable producer's bid $(b_{R,t}, p_{R,t})$ is **accepted**, it must deliver $b_{R,t}$.
Actual generation $g_t$ may differ from $b_{R,t}$, incurring penalties for both under- and over-delivery:

$$C_t(g_t, b_{R,t}) = c_t^{\text{under}}[b_{R,t} - g_t]_+ + c_t^{\text{over}}[g_t - b_{R,t}]_+,$$

where $c_t^{\text{under}}$, $c_t^{\text{over}}$ are penalty rates.

Then, for hour $t$:

$$\Pi_t^R(b_{R,t}, p_{R,t}; g_t, \pi_t) = \begin{cases} \pi_t b_{R,t} - C_t(g_t, b_{R,t}), & \text{if } p_{R,t} \le \pi_t, \\ 0, & \text{otherwise.} \end{cases}$$

## 🧮 5. Renewable Producer's Optimization Problem

The renewable producer strategically selects bids $(b_{R,t}, p_{R,t})$ for all hours $t = 1, \dots, 24$ **before** market clearing.

Because the clearing price $\pi_t$ depends on all producers' bids, the renewable producer faces an **equilibrium problem**.

Formally:

$$\begin{aligned} \max_{\{b_{R,t}, p_{R,t}\}_{t=1}^{24}} \sum_{t=1}^{24} \min_{g_t \in \mathcal{G}_t} \Pi_t^R(b_{R,t}, p_{R,t}; g_t, \pi_t(\mathbf{b}_t, \mathbf{p}_t)) \\ \text{s.t. } 0 \le b_{R,t} \le \bar{b}_t, \quad p_{R,t} \ge 0, \quad t = 1, \dots, 24, \end{aligned}$$

where $\pi_t(\mathbf{b}_t, \mathbf{p}_t)$ denotes the **endogenous clearing price** given all producers' bids $\mathbf{b}_t, \mathbf{p}_t$.

- The **inner minimization** over $g_t \in \mathcal{G}_t$ represents **robust (worst-case) profit** under deep uncertainty in renewable output.
- This yields a **robust game** among producers: each maximizes their own worst-case profit given others' bids.

## 🎮 6. Market Equilibrium (Merit-Order Equilibrium)

The market outcome satisfies a **Nash equilibrium** condition across producers $i = 1, \dots, 4$:

$$(b_{i,t}^*, p_{i,t}^*) \in \arg\max_{b_{i,t}, p_{i,t}} \sum_{t=1}^{24} U_{i,t}(b_{i,t}, p_{i,t}; \pi_t(\mathbf{b}_t, \mathbf{p}_t)),$$

where $U_{i,t}$ is the profit function for each producer, and $\pi_t(\cdot)$ is the market-clearing rule above.

The renewable producer's $U_{R,t}$ incorporates the worst-case realization of $g_t$:

$$U_{R,t}(b_{R,t}, p_{R,t}) = \min_{g_t \in \mathcal{G}_t} \Pi_t^R(b_{R,t}, p_{R,t}; g_t, \pi_t(\mathbf{b}_t, \mathbf{p}_t)).$$

## 📊 7. Model Components (Summary Table)

| Category | Symbol | Description |
|---|---|---|
| **Indices** | $t = 1, \ldots, 24$ | Hours of next day |
| **Decisions (renewable)** | $b_{R,t}, p_{R,t}$ | Bid quantity and offer price |
| **Decisions (others)** | $b_{i,t}, p_{i,t}, i = 1, 2, 3$ | Competitors' bids |
| **Endogenous variable** | $\pi_t$ | Clearing price from market merit-order |
| **Deep uncertainty** | $g_t \in \mathcal{G}_t$ | Unknown renewable generation (no known probability law) |
| **Parameters** | $D_t, c_t^{\text{under}}, c_t^{\text{over}}, \bar{b}_t$ | Demand, imbalance penalties, bid limits |
| **Profit function** | $\Pi_t^R(b_{R,t}, p_{R,t}; g_t, \pi_t)$ | Hourly profit conditional on acceptance |
| **Objective** | Robust expected profit over 24 hours | Maximize worst-case total profit |

## 🧱 8. Compact Robust Formulation

For a single day (24 hours):

$$\boxed{\begin{aligned} &\max_{\{b_{R,t}, p_{R,t}\}} \sum_{t=1}^{24} \min_{g_t \in \mathcal{G}_t} \left[ \mathbf{1}_{\{p_{R,t} \le \pi_t(\mathbf{b}_t, \mathbf{p}_t)\}} \left( \pi_t(\mathbf{b}_t, \mathbf{p}_t) b_{R,t} - c_t^{\text{under}} [b_{R,t} - g_t]_+ - c_t^{\text{over}} [g_t - b_{R,t}]_+ \right) \right] \\ &\quad \text{s.t.} \quad 0 \le b_{R,t} \le \bar{b}_t, \quad p_{R,t} \ge 0, \quad \forall t. \end{aligned}}$$

This is a **robust–equilibrium bidding model** under:

- **Deep uncertainty** (no probability law for $g_t$), handled via a worst-case term;
- **Endogenous prices**, determined by the market-clearing merit order;
- **Strategic interaction** with other producers (each optimizing their own profit).

### ✅ Key Insights:

- **No stochastic expectation:** since renewable output is *deeply uncertain*, we use a **robust (min–max)** structure instead of $\mathbb{E}[\cdot]$.
- **Endogenous market price:** the clearing price is determined *within* the model from all bids, not exogenously drawn.

- **Equilibrium-based formulation:** the renewable producer's optimization is part of a broader market equilibrium.

---

Would you like me to extend this next into a **formal equilibrium model** (e.g., a complementarity or mixed-integer equilibrium problem) showing explicitly how the clearing price and accepted bids are determined from all producers' decisions?

> You can assume the total energy demand $D_t$ follows a normal distribution with constant mean and variance. You can also assume that 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 the model specification.

Excellent — now we can formalize the model fully, integrating all your clarifications in a **realistic and complete structure** that combines:

- **Endogenous market clearing** (merit-order mechanism),
- **Deep uncertainty in renewable generation**,
- **Stochasticity in total demand and competitors' bids** (modeled with normal distributions), and
- **Robust expected profit maximization** for the renewable producer.

Let's build the refined specification carefully.

---

## ⚙️ 1. Model Overview

The **day-ahead electricity market** is cleared once per day for 24 hourly intervals.
Four producers participate:

- $i = 1, 2, 3$: conventional producers,
- $i = R$: renewable producer (our decision-maker).

Each producer $i$ submits for each hour $t = 1, \dots, 24$:

$$(b_{it}, p_{it}), \quad b_{it} \geq 0,\ p_{it} \geq 0,$$

where $b_{it}$ is the offer quantity (MWh) and $p_{it}$ the offer price (€/MWh).

The market operator then clears the market based on all bids.

---

## 💡 2. Uncertainty Structure

We distinguish between **aleatory (probabilistic)** and **deep (non-probabilistic)** uncertainties.

### (a) Deep uncertainty

- **Renewable generation** $g_t$:
  cannot be characterized by a known probability distribution; it lies in a known **uncertainty set**

$$g_t \in \mathcal{G}_t = [\underline{g}_t, \bar{g}_t].$$

### (b) Probabilistic uncertainties

- **Total demand** $D_t$ is random:

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

  assumed independent across hours.
- **Competitors' bids** $(b_{it}, p_{it})$ for $i = 1, 2, 3$ are random with known joint distributions:

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

  assumed independent across $i$ and $t$.

Thus, the renewable producer faces:

- stochastic competitor and demand conditions (through $D_t$, $b_{it}$, $p_{it}$),
- deep uncertainty in its own generation $g_t$.

---

## ⚙️ 3. Market-Clearing Mechanism (Endogenous Price)

For each hour $t$:

1. Aggregate supply curve: all producers' bids $(b_{it}, p_{it})$.
2. Sort bids in **ascending order of price** to form the **merit order**.
3. The **market clearing price** $\pi_t$ satisfies:

$$\pi_t = \min \left\{ p \;\middle|\; \sum_{i=1}^{4} b_{it} \, \mathbf{1}_{\{p_{it} \le p\}} \ge D_t \right\}.$$

All producers with $p_{it} \le \pi_t$ are **accepted** and must deliver $b_{it}$ at the price $\pi_t$.

The random variables $D_t$, $b_{it}$, and $p_{it}$ jointly determine $\pi_t$.

---

## 💰 4. Renewable Producer's Profit Function

If the renewable producer's offer $(b_{Rt}, p_{Rt})$ is **accepted** ($p_{Rt} \le \pi_t$), its realized profit is:

$$\Pi_t^R(b_{Rt}, p_{Rt}; g_t, \pi_t) = \pi_t b_{Rt} - C_t(g_t, b_{Rt}),$$

with two-sided imbalance penalties:

$$C_t(g_t, b_{Rt}) = c_t^{\text{under}} [b_{Rt} - g_t]_+ + c_t^{\text{over}} [g_t - b_{Rt}]_+.$$

If $p_{Rt} > \pi_t$, the bid is rejected and profit is zero.

Thus,

$$\Pi_t^R(b_{Rt}, p_{Rt}; g_t, \pi_t) = \mathbf{1}_{\{p_{Rt} \leq \pi_t\}} \left( \pi_t b_{Rt} - c_t^{\text{under}} [b_{Rt} - g_t]_+ - c_t^{\text{over}} [g_t - b_{Rt}]_+ \right).$$

## 🎯 5. Renewable Producer's Objective

Since:

- $g_t$ is **deeply uncertain** (unknown distribution), and
- the clearing price $\pi_t$ is a **random variable** determined by stochastic demand and competitors' bids,

the renewable producer adopts a **robust-expected optimization** approach:
maximize expected profit (over aleatory randomness) **in the worst case** over deep uncertainty.

$$\boxed{\max_{\{b_{Rt}, p_{Rt}\}} \ \mathbb{E}_{D_t, b_{it}, p_{it}} \left[ \sum_{t=1}^{24} \min_{g_t \in \mathcal{G}_t} \Pi_t^R(b_{Rt}, p_{Rt}; g_t, \pi_t(D_t, \{b_{it}, p_{it}\}_{i=1}^3, b_{Rt}, p_{Rt})) \right]}$$

subject to

$$0 \leq b_{Rt} \leq \bar{b}_t, \quad p_{Rt} \geq 0, \quad \forall t = 1, \ldots, 24.$$

## 📊 6. Model Components (Summary Table)

| Category | Symbol | Description | Type |
|---|---|---|---|
| **Indices** | $t = 1, \ldots, 24$ | Hours of next day | — |
| **Decision variables** | $b_{Rt}, p_{Rt}$ | Renewable bid quantity and price | Decision |
| **Competitor offers** | $b_{it}, p_{it}$ for $i = 1, 2, 3$ | Bids from conventional producers | Random (Normal) |
| **Demand** | $D_t \sim \mathcal{N}(\mu_D, \sigma_D^2)$ | Total hourly energy demand | Random (Normal) |
| **Renewable generation** | $g_t \in [\underline{g}_t, \bar{g}_t]$ | Realized renewable output | Deep uncertainty |
| **Market price** | $\pi_t(\cdot)$ | Clearing price from merit-order mechanism | Endogenous random variable |
| **Penalties** | $c_t^{\text{under}}, c_t^{\text{over}}$ | Under- and over-delivery penalties | Parameters |
| **Bid limits** | $\bar{b}_t$ | Max feasible bid | Parameter |

## 🔄 7. Interpretation of the Structure

1. **Endogeneity of Prices:**
   Prices $\pi_t$ are determined by the **aggregate stochastic supply** and **stochastic demand**, not exogenously.

2. **Two Layers of Uncertainty:**
   - **Outer expectation** over random variables $(D_t, b_{it}, p_{it})$; these have known Normal distributions.
   - **Inner minimization** over $g_t$, representing worst-case renewable availability.
3. **Static 24-hour optimization:**
   The renewable producer submits all bids for the next day simultaneously; no inter-hour dynamics are considered.
4. **Game-theoretic but partially stochastic environment:**
   The renewable producer treats competitors' bids as random rather than strategic (a "stochastic merit-order" framework).

---

## 🧱 8. Compact Mathematical Specification

$$
\begin{aligned}
\max_{\{b_{Rt}, p_{Rt}\}} \quad & \mathbb{E}_{D_t, b_{it}, p_{it}} \left[ \sum_{t=1}^{24} \min_{g_t \in [\underline{g}_t, \bar{g}_t]} \mathbf{1}_{\{p_{Rt} \le \pi_t(D_t, \{b_{it}, p_{it}\}, b_{Rt}, p_{Rt})\}} \left( \pi_t(\cdot) b_{Rt} - c_t^{\text{under}} [b_{Rt} - g_t]_+ - c_t^{\text{over}} [g_t - b_{Rt}]_+ \right. \right. \\
\text{s.t.} \quad & 0 \le b_{Rt} \le \bar{b}_t, \quad p_{Rt} \ge 0, \quad t = 1, \ldots, 24.
\end{aligned}
$$

---

## ✅ 9. Conceptual Summary

- **Decision layer:** choose hourly bids $(b_{Rt}, p_{Rt})$.
- **Random environment layer:** stochastic demand and competitor bids ⇒ stochastic merit-order clearing price.
- **Deep uncertainty layer:** unknown renewable generation, handled via robust (worst-case) evaluation.
- **Outcome:** a robust expected-profit optimization with an endogenous, stochastic clearing price.

---

Would you like me to extend this specification into a **scenario-based computational model**, showing how to approximate the expectation over random bids/demand and the minimization over $g_t$ via sample averages and robust envelopes?

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 exactly the kind of systems-level refinement that clarifies how multiple stakeholders interact within a common market and physical environment.

We now explicitly model the **shared environment** (the electricity market and physical system) and then formulate **two stakeholder models**:

1. The **renewable producer** — a self-interested market participant optimizing its bids under uncertainty.
2. The **system regulator** — a social planner or market operator pursuing system-wide objectives (efficiency, reliability, decarbonization, etc.), indirectly influencing market outcomes via rules or parameters.

Both share the same **environmental variables and parameters**, and interact *indirectly* through the **market-clearing mechanism** and resulting prices and penalties.

---

# 🌐 1. Common Environment (Shared Between Stakeholders)

The environment represents the **day-ahead electricity market** for 24 hourly intervals of the following day.

## (a) Global Parameters

| Symbol | Description |
|---|---|
| $t = 1, \ldots, 24$ | Hour of the next day |
| $i \in \{1, 2, 3, R\}$ | Producers: 3 conventional + 1 renewable |
| $\mu_D, \sigma_D^2$ | Mean and variance of hourly demand (Normal) |
| $\mu_{b_i}, \sigma_{b_i}^2$ | Mean and variance of bid quantities for conventional producer $i$ |
| $\mu_{p_i}, \sigma_{p_i}^2$ | Mean and variance of offer prices for conventional producer $i$ |
| $c_t^{\text{under}}, c_t^{\text{over}}$ | Penalty rates for renewable imbalance |
| $[\underline{g}_t, \bar{g}_t]$ | Feasible set of renewable generation (deep uncertainty) |
| $\bar{b}_t$ | Max feasible bid of the renewable producer |
| $D_t \sim \mathcal{N}(\mu_D, \sigma_D^2)$ | Stochastic hourly demand |
| Market rule | Uniform-price market with merit-order clearing |

---

## (b) Global (Environmental) Variables

| Symbol | Description | Type |
|---|---|---|
| $b_{it}, p_{it}$ | Hourly bids from all producers | Random (Normal for $i = 1, 2, 3$); Decision for $i = R$ |
| $D_t$ | Hourly system demand | Random |
| $\pi_t$ | Market clearing price (€/MWh) | Endogenous variable determined by bids and demand |
| $g_t$ | Realized renewable generation (MWh) | Deeply uncertain, $g_t \in [\underline{g}_t, \bar{g}_t]$ |

| Symbol | Description | Type |
|---|---|---|
| $\mathcal{A}_t = \{i \mid p_{it} \leq \pi_t\}$ | Accepted producers in hour $t$ | Deterministic function of bids and demand |
| $Q_t = \sum_{i \in \mathcal{A}_t} b_{it}$ | Total accepted supply | Derived variable |

The market clearing mechanism couples all stakeholders through $\pi_t$:

$$\pi_t = \min \left\{ p \;\middle|\; \sum_{i : p_{it} \leq p} b_{it} \geq D_t \right\}.$$

This shared environment defines the **system dynamics** connecting both agents.

---

# 🎯 2. Perspective (1): Renewable Producer

## (a) Role

The renewable producer is a **strategic agent** that submits hourly bids $(b_{Rt}, p_{Rt})$ before the market clears.
It aims to **maximize economic performance** while being robust to uncertainty in its own generation $g_t$.

---

## (b) Control Variables

| Variable | Meaning |
|---|---|
| $b_{Rt}$ | Bid quantity for hour $t$ (MWh) |
| $p_{Rt}$ | Offer price for hour $t$ (€/MWh) |

---

## (c) State Variables and Transitions

- **State variable:** $r_t$ = informational or physical state (e.g., forecasted wind/solar availability).
  The environment does not evolve dynamically within the 24-hour bidding problem, but the renewable resource may evolve as:

  $$r_{t+1} = f(r_t, \omega_t),$$

  where $\omega_t$ is a weather disturbance.
  Actual generation is constrained by $g_t \in [\underline{g}_t(r_t), \bar{g}_t(r_t)]$.
- **Transition:** purely informational — used to define the uncertainty set $\mathcal{G}_t$.

---

## (d) Objective Functions (Multiple)

Because the renewable producer may have *economic* and *operational* motivations, we separate its objectives rather than aggregate them:

1. **Economic Objective (expected profit):**

$$J_1^R = \mathbb{E}_{D_t, b_{it}, p_{it}} \left[ \sum_{t=1}^{24} \mathbf{1}_{\{p_{Rt} \le \pi_t\}} \left( \pi_t b_{Rt} - c_t^{\text{under}} [b_{Rt} - g_t]_+ - c_t^{\text{over}} [g_t - b_{Rt}]_+ \right) \right].$$

2. **Robustness Objective (worst-case resilience):**

$$J_2^R = \min_{g_t \in [\underline{g}_t, \bar{g}_t]} \sum_{t=1}^{24} \mathbf{1}_{\{p_{Rt} \le \pi_t\}} \left( \pi_t b_{Rt} - c_t^{\text{under}} [b_{Rt} - g_t]_+ - c_t^{\text{over}} [g_t - b_{Rt}]_+ \right).$$

The producer may pursue either $J_1^R$ or $J_2^R$ (risk-neutral vs. robust stance), or report both to compare strategies.

---

### (e) Constraints

$$0 \le b_{Rt} \le \bar{b}_t, \quad p_{Rt} \ge 0, \quad \forall t.$$

---

### (f) Renewable Producer's Model Summary

| Element | Specification |
|---|---|
| **Control variables** | $b_{Rt}, p_{Rt}$ |
| **State variable** | $r_t$ (resource forecast state) |
| **Transition** | $r_{t+1} = f(r_t, \omega_t)$ |
| **Uncertainties** | Deep uncertainty: $g_t \in [\underline{g}_t, \bar{g}_t]$; stochastic demand and competitor bids |
| **Objectives** | $J_1^R$ (expected profit), $J_2^R$ (worst-case profit) |
| **Coupling to environment** | Via endogenous market price $\pi_t(\cdot)$ |

---

## 🏛️ 3. Perspective (2): System Regulator

### (a) Role

The **system regulator** (or market operator / policymaker) oversees market efficiency, reliability, and environmental performance.
The regulator does **not** choose bids directly but can adjust *market design parameters* or *penalty coefficients* to steer the system toward socially desirable outcomes.

---

### (b) Control Variables

| Variable | Meaning |
|---|---|
| $c_t^{\text{under}}, c_t^{\text{over}}$ | Imbalance penalty coefficients |
| Market rules | Design of pricing rule, e.g., uniform-price mechanism or reserve requirements |
| Forecasting/penalty structure | Definitions of $[\underline{g}_t, \bar{g}_t]$ or required reserve margins |

The regulator can change these policy parameters to indirectly influence producer behavior and equilibrium prices.

---

### (c) State Variables and Transitions

| Variable | Meaning |
|---|---|
| $s_t$ | System state summarizing reliability, emissions, or market performance indicators |
| Transition | $s_{t+1} = F(s_t, \pi_t, Q_t, D_t)$, capturing how market outcomes influence the system state (e.g., reliability or emissions stock) |

Example: if renewable shortfalls occur, reliability $s_t$ deteriorates; if renewable bids increase, emissions decrease.

---

### (d) Objective Functions (Multiple)

The regulator typically has **multiple, conflicting goals**, so we specify several distinct objective functions rather than a weighted sum:

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

$$J_1^{\mathrm{Reg}} = \mathbb{E}_{D_t, b_{it}, p_{it}} \left[ \sum_{t=1}^{24} (U(D_t) - \pi_t D_t) \right],$$

   where $U(D_t)$ is the consumer benefit function (e.g., demand utility).

2. **Reliability (penalize imbalance risk):**

$$J_2^{\mathrm{Reg}} = -\sum_{t=1}^{24} \mathbb{E}\big[ |Q_t - D_t| \big],$$

   minimizing expected supply-demand mismatches due to renewable uncertainty.

3. **Environmental Objective (emission reduction):**

$$J_3^{\mathrm{Reg}} = -\sum_{t=1}^{24} \mathbb{E} \left[ \sum_{i=1}^{3} \alpha_i b_{it} \mathbf{1}_{\{p_{it} \le \pi_t\}} \right],$$

   where $\alpha_i$ is the emission rate ($tCO_2$/MWh) for conventional producer $i$.

Each objective captures a separate social or policy goal: efficiency, reliability, and decarbonization.

---

### (e) Constraints

- Market balance:

$$\sum_{i=1}^{4} b_{it} \mathbf{1}_{\{p_{it} \le \pi_t\}} = D_t, \quad \forall t.$$

- System reliability and capacity limits:

$$Q_t \le \bar{Q}_t, \quad s_t \in \mathcal{S}.$$

---

### (f) Regulator's Model Summary

| Element | Specification |
|---|---|
| **Control variables** | $c_t^{\text{under}}, c_t^{\text{over}}$, market design parameters |
| **State variables** | $s_t$ (system performance indicators) |
| **Transition function** | $s_{t+1} = F(s_t, \pi_t, Q_t, D_t)$ |
| **Uncertainties** | Demand $D_t$, producers' bids, renewable availability |
| **Objectives** | $J_1^{\text{Reg}}$ (efficiency), $J_2^{\text{Reg}}$ (reliability), $J_3^{\text{Reg}}$ (emissions) |
| **Coupling to environment** | Via market-clearing prices and penalty coefficients affecting producers' incentives |

---

## 🔗 4. Shared Coupling and Interaction Mechanism

Both the renewable producer and the system regulator operate within — and influence — the **shared market environment**:

- The **renewable producer** directly affects the merit order and market clearing price through its bids $(b_{Rt}, p_{Rt})$.
- The **regulator** indirectly affects the producer's incentives and the resulting prices through parameters $(c_t^{\text{under}}, c_t^{\text{over}})$ and the market rule.
- The market clearing price $\pi_t$ is a **shared endogenous variable**, coupling their decisions.

This can be represented schematically as:

$$\begin{cases} \pi_t = \text{MarketClearing}(b_{Rt}, p_{Rt}; \text{competitor bids}, D_t), \\ g_t \in [\underline{g}_t, \bar{g}_t], \\ \text{Renewable: } (b_{Rt}, p_{Rt}) \text{ maximize } J_1^R, J_2^R, \\ \text{Regulator: } (c_t^{\text{under}}, c_t^{\text{over}}, \text{rules}) \text{ maximize } J_1^{\text{Reg}}, J_2^{\text{Reg}}, J_3^{\text{Reg}}. \end{cases}$$

---

## ✅ 5. Conceptual Summary

| Aspect | Renewable Producer | System Regulator |
|---|---|---|
| **Decision scope** | Submit bids $(b_{Rt}, p_{Rt})$ | Set policy parameters and penalty rates |
| **Primary control effect** | Directly influences clearing price | Indirectly influences producer behavior and equilibrium outcomes |
| **Uncertainty handling** | Deep uncertainty in $g_t$; stochastic demand/competitors | Aggregate stochastic outcomes of market |
| **Objective structure** | Economic profit and robustness | Efficiency, reliability, emissions |
| **Coupling** | Through market clearing and penalties | Same |
| **Shared environment** | Market operator, stochastic demand, endogenous prices, renewable availability | Same |

---

This refined specification now provides:

- A **common stochastic–robust market environment**,
- Two **independent control agents** (renewable producer and regulator),
- Explicit **coupling via market clearing**, and
- Separate, **multi-objective formulations** reflecting each stakeholder's perspective.

---

Would you like me to extend this next into a **bilevel (regulator–producer) optimization framework** to show how the regulator's policy design anticipates the producer's optimal response?

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

Excellent — that's a clear and important directive.
We'll now refine the model into an **implementable, fully quantifiable form**, keeping only measurable, operationally definable quantities and relationships.

We will retain two stakeholder perspectives — **(1) renewable producer** and **(2) system regulator** — within a **shared, well-defined market environment**.
All conceptual or qualitative components (like "social welfare" or "emissions") will be replaced by measurable or model-ready quantities.

---

# 🌐 1. Common Environment (Quantifiable and Shared)

The **shared environment** is a day-ahead electricity market that clears once per day for 24 hours $t = 1, \dots, 24$.
It determines prices, dispatch, and imbalances based on bids and demand.

### (a) Global Parameters

| Symbol | Units | Description |
|---|---|---|
| $N = 4$ | — | Total producers (3 conventional + 1 renewable) |
| $t = 1, \dots, 24$ | — | Hour index |
| $\mu_D, \sigma_D$ | MWh, MWh | Mean and standard deviation of hourly demand $D_t$ |
| $\mu_{b_i}, \sigma_{b_i}$ | MWh | Mean and standard deviation of bid quantities for conventional producer $i$ |
| $\mu_{p_i}, \sigma_{p_i}$ | €/MWh | Mean and standard deviation of offer prices for conventional producer $i$ |
| $[\underline{g}_t, \bar{g}_t]$ | MWh | Feasible interval for renewable generation (deep uncertainty) |
| $c_t^{\text{under}}, c_t^{\text{over}}$ | €/MWh | Penalty coefficients for renewable under-/over-delivery |
| $\bar{b}_t$ | MWh | Maximum bid allowed for renewable producer |
| $D_t \sim \mathcal{N}(\mu_D, \sigma_D^2)$ | MWh | Hourly system demand (random) |

---

## (b) Environmental Variables

| Symbol | Units | Description | Type |
|---|---|---|---|
| $b_{it}, p_{it}$ | MWh, €/MWh | Hourly bid quantity and price from producer $i$ | Random for $i = 1, 2, 3$; decision for $i = R$ |
| $D_t$ | MWh | Total system demand | Random |
| $\pi_t$ | €/MWh | Market clearing price (endogenous) | Determined by bids and demand |
| $g_t$ | MWh | Renewable generation | Deeply uncertain ($g_t \in [\underline{g}_t, \bar{g}_t]$) |
| $\mathcal{A}_t$ | — | Set of accepted producers in hour $t$ | Determined by market clearing rule |

## (c) Market-Clearing Rule (Quantifiable)

For each hour $t$:

1. All offers $(b_{it}, p_{it})$ are sorted in ascending order of $p_{it}$.
2. The **clearing price** $\pi_t$ is determined by the lowest price at which cumulative supply meets demand $D_t$:

$$\pi_t = \min \left\{ p \mid \sum_{i: p_{it} \le p} b_{it} \ge D_t \right\}.$$

3. Producers $i$ with $p_{it} \le \pi_t$ are accepted, i.e., $i \in \mathcal{A}_t$.

This rule is entirely computable given the bids and demand samples.

# ⚙️ 2. Renewable Producer Model

## (a) Control Variables

| Variable | Units | Description |
|---|---|---|
| $b_{Rt}$ | MWh | Quantity bid for hour $t$ |
| $p_{Rt}$ | €/MWh | Offer price for hour $t$ |

## (b) Uncertain Inputs (Exogenous to Producer)

| Symbol | Units | Distribution / Range |
|---|---|---|
| $D_t$ | MWh | $\mathcal{N}(\mu_D, \sigma_D^2)$ |
| $b_{it}, p_{it}, i = 1, 2, 3$ | MWh, €/MWh | Independent normals $\mathcal{N}(\mu_{b_i}, \sigma_{b_i}^2)$, $\mathcal{N}(\mu_{p_i}, \sigma_{p_i}^2)$ |
| $g_t$ | MWh | Deep uncertainty set $[\underline{g}_t, \bar{g}_t]$ |

## (c) Outcome Variables (Derived)

| Symbol | Units | Definition |
| --- | --- | --- |
| $\pi_t$ | €/MWh | Clearing price from market rule |
| $\delta_t = \mathbf{1}_{\{p_{Rt} \le \pi_t\}}$ | — | Acceptance indicator for renewable bid |
| $\mathrm{imb}_t = g_t - b_{Rt}$ | MWh | Imbalance (positive = over-delivery) |

### (d) Quantifiable Profit Function

Hourly profit:

$$\Pi_t^R(b_{Rt}, p_{Rt}; g_t, \pi_t) = \delta_t \left(\pi_t b_{Rt} - c_t^{\text{under}}[b_{Rt} - g_t]_+ - c_t^{\text{over}}[g_t - b_{Rt}]_+\right).$$

Total daily profit (random due to $D_t, b_{it}, p_{it}$):

$$\Pi^R = \sum_{t=1}^{24} \Pi_t^R.$$

### (e) Objective Functions (Measurable)

1. **Expected profit under stochastic demand and competitors:**

$$J_1^R = \mathbb{E}_{D_t, b_{it}, p_{it}}\left[\Pi^R(b_{Rt}, p_{Rt}; g_t, \pi_t)\right].$$

2. **Worst-case profit under renewable uncertainty:**

$$J_2^R = \min_{g_t \in [\underline{g}_t, \bar{g}_t]} \mathbb{E}_{D_t, b_{it}, p_{it}}\left[\Pi^R(b_{Rt}, p_{Rt}; g_t, \pi_t)\right].$$

Both are directly computable using Monte Carlo samples of $D_t, b_{it}, p_{it}$ and bounds for $g_t$.

### (f) Optimization Problem (Implementable)

$$\begin{aligned} \max_{b_{Rt}, p_{Rt}} \quad & J_1^R \quad \text{and/or} \quad J_2^R, \\ \text{s.t.} \quad & 0 \le b_{Rt} \le \bar{b}_t, \quad p_{Rt} \ge 0, \quad t = 1, \dots, 24. \end{aligned}$$

## 🏛️ 3. System Regulator Model

### (a) Control Variables (Quantifiable)

| Variable | Units | Description |
| --- | --- | --- |
| $c_t^{\text{under}}$ | €/MWh | Penalty rate for under-delivery |
| $c_t^{\text{over}}$ | €/MWh | Penalty rate for over-delivery |
| $L_t$ | MWh | Maximum imbalance tolerance (reliability constraint) |

These are policy parameters that the regulator can set or adjust.

### (b) Observable Quantities from the Environment

| Symbol | Units | Description |
|---|---|---|
| $D_t$ | MWh | Realized system demand |
| $Q_t = \sum_{i \in \mathcal{A}_t} b_{it}$ | MWh | Total accepted supply |
| $\pi_t$ | €/MWh | Market clearing price |
| $\text{imb}_t = g_t - b_{Rt}$ | MWh | Renewable imbalance |

All are directly measurable or computable from bids, market results, and renewable output.

### (c) Quantifiable Objectives

1. **Price stability objective:** minimize price variance

$$J_1^{\text{Reg}} = -\text{Var}(\pi_t).$$

   (High variance indicates market volatility.)

2. **System balance (reliability) objective:** minimize expected absolute imbalance

$$J_2^{\text{Reg}} = -\sum_{t=1}^{24} \mathbb{E}[|Q_t - D_t|].$$

3. **Cost containment (social cost):** minimize expected total payout to producers

$$J_3^{\text{Reg}} = -\sum_{t=1}^{24} \mathbb{E}[\pi_t D_t].$$

Each $J_k^{\text{Reg}}$ is directly measurable and computable from observed or simulated market outcomes.

### (d) Constraints

$$\begin{aligned} &0 \le c_t^{\text{under}}, c_t^{\text{over}} \le c_{\max}, \\ &\mathbb{E}[|Q_t - D_t|] \le L_t, \quad t = 1, \ldots, 24. \end{aligned}$$

### (e) Optimization Problem (Implementable)

The regulator's implementable design problem:

$$\begin{aligned} \max_{c_t^{\text{under}}, c_t^{\text{over}}} \quad & (J_1^{\text{Reg}}, J_2^{\text{Reg}}, J_3^{\text{Reg}}) \\ \text{s.t.} \quad & \text{Market clearing rule and renewable response: } (b_{Rt}, p_{Rt}) = \arg\max J_1^R, \\ & 0 \le c_t^{\text{under}}, c_t^{\text{over}} \le c_{\max}. \end{aligned}$$

This creates a bilevel interaction where the regulator's policy parameters affect the renewable producer's bidding strategy, and both affect market outcomes.

# 🔗 4. Quantifiable Coupling Between the Two Agents

| Coupling Variable | Definition | Quantifiable Source |
|---|---|---|
| $\pi_t$ | Clearing price | Computed from merit-order rule |
| $Q_t$ | Total accepted supply | Derived from accepted bids |
| $g_t$ | Renewable output | Measured (realization or bound) |
| $c_t^{\text{under}}, c_t^{\text{over}}$ | Penalty coefficients | Regulator decision |
| $b_{Rt}, p_{Rt}$ | Renewable bids | Producer decision |

Both stakeholders influence each other only via these measurable variables through the **market clearing equation**.

---

# ✅ 5. Summary Table

| Aspect | Renewable Producer | System Regulator |
|---|---|---|
| **Control variables** | $b_{Rt}, p_{Rt}$ | $c_t^{\text{under}}, c_t^{\text{over}}$ |
| **Random inputs** | $D_t, b_{it}, p_{it}$ | Same |
| **Deep uncertainty** | $g_t \in [\underline{g}_t, \bar{g}_t]$ | Same |
| **Objective(s)** | $J_1^R$ (expected profit), $J_2^R$ (worst-case profit) | $J_1^{\text{Reg}}$ (price stability), $J_2^{\text{Reg}}$ (reliability), $J_3^{\text{Reg}}$ (cost containment) |
| **Shared variables** | $\pi_t, Q_t, g_t, c_t^{\text{under}}, c_t^{\text{over}}$ | Same |
| **Constraints** | Bid bounds | Penalty bounds, balance limits |
| **Coupling mechanism** | Merit-order market clearing | Same |

---

This refined specification now includes **only quantifiable, implementable elements**:

- All random variables have explicit distributions or bounded uncertainty sets.
- All objectives are measurable from simulated or real market data.
- All constraints and transitions are algebraic or directly computable.
- Both agents interact solely via observable market quantities.

---

Would you like me to express this interaction formally as a **bilevel (leader–follower) optimization structure** next, with explicit equilibrium conditions for $\pi_t$ and accepted bids?

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. enables 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 can build a **unified, internally consistent model** that integrates both stakeholder perspectives (renewable producer and regulator) within a **shared market environment**, satisfying your four requirements:

1. A single, shared environment and state definition.
2. Embedded, modular sub-models for each stakeholder (each with its own controls and objectives).
3. Each stakeholder treats the other's controls as exogenous.
4. All previously defined components (variables, uncertainties, parameters, and market mechanisms) are included.

The result is a **modular multi-agent market system model**.

---

# 🌐 1. Unified Common Environment

This environment defines the shared **states**, **uncertainties**, and **market-clearing mechanism** that both stakeholders depend on.

### (a) Sets and Indices

$$t \in \{1, \ldots, 24\}, \quad i \in \{1, 2, 3, R\}.$$

### (b) Global Parameters

| Symbol | Units | Description |
|---|---|---|
| $\mu_D, \sigma_D$ | MWh | Mean and std. deviation of demand |
| $\mu_{b_i}, \sigma_{b_i}$ | MWh | Mean and std. deviation of bid quantity for conventional producer $i$ |
| $\mu_{p_i}, \sigma_{p_i}$ | €/MWh | Mean and std. deviation of offer price for conventional producer $i$ |
| $[\underline{g}_t, \bar{g}_t]$ | MWh | Feasible generation range for renewable producer |
| $\bar{b}_t$ | MWh | Max bid allowed for renewable producer |
| $c_t^{\text{under}}, c_t^{\text{over}}$ | €/MWh | Penalty rates (controlled by regulator) |
| $c_{\max}$ | €/MWh | Maximum allowable penalty rate |
| $L_t$ | MWh | Target limit on expected imbalance |

---

### (c) Random Variables (Uncertainties)

| Symbol | Units | Distribution |
|---|---|---|
| $D_t$ | MWh | $\mathcal{N}(\mu_D, \sigma_D^2)$ |
| $b_{it}, p_{it}, i = 1, 2, 3$ | MWh, €/MWh | $b_{it} \sim \mathcal{N}(\mu_{b_i}, \sigma_{b_i}^2)$, $p_{it} \sim \mathcal{N}(\mu_{p_i}, \sigma_{p_i}^2)$ |
| $g_t$ | MWh | Deep uncertainty set $[\underline{g}_t, \bar{g}_t]$ |

### (d) Shared Environmental Variables

| Symbol | Units | Description | Type |
|---|---|---|---|
| $\pi_t$ | €/MWh | Market clearing price | Endogenous |
| $\mathcal{A}_t$ | — | Accepted producers set | Derived |
| $Q_t = \sum_{i \in \mathcal{A}_t} b_{it}$ | MWh | Total accepted supply | Derived |
| $\text{imb}_t = g_t - b_{Rt}$ | MWh | Renewable imbalance | Derived |
| $s_t$ | Vector | System state (includes $D_t, \pi_t, Q_t, g_t$) | State variable |

### (e) Market-Clearing Function (Shared Mechanism)

For each hour $t$:

$$\pi_t = \min \left\{ p \mid \sum_{i: p_{it} \leq p} b_{it} \geq D_t \right\},$$

and

$$\mathcal{A}_t = \{ i \mid p_{it} \leq \pi_t \}.$$

This rule couples all participants' bids and the stochastic environment, ensuring internal consistency.

## ⚙️ 2. Embedded Perspective Modules

The unified model embeds **two modular sub-models** that share the environment but have separate controls and objectives.

### (A) Module 1: Renewable Producer Submodel

#### 1. Control Variables

| Symbol | Units | Description |
|---|---|---|
| $b_{Rt}$ | MWh | Renewable bid quantity per hour |
| $p_{Rt}$ | €/MWh | Renewable offer price per hour |

#### 2. State and Uncertainties

| Symbol | Description |
|---|---|
| $s_t = (D_t, \pi_t, Q_t, g_t)$ | Shared system state |
| $g_t \in [\underline{g}_t, \bar{g}_t]$ | Deep uncertainty in renewable output |
| $D_t, b_{it}, p_{it}$ | Random from known distributions |

#### 3. Derived Variables

$$\delta_t = \mathbf{1}_{\{p_{Rt} \le \pi_t\}}, \quad \mathrm{imb}_t = g_t - b_{Rt}.$$

#### 4. Profit Function

$$\Pi_t^R(b_{Rt}, p_{Rt}; g_t, \pi_t) = \delta_t \left( \pi_t b_{Rt} - c_t^{\mathrm{under}} [b_{Rt} - g_t]_+ - c_t^{\mathrm{over}} [g_t - b_{Rt}]_+ \right).$$

#### 5. Objectives

All are quantifiable given sampled $(D_t, b_{it}, p_{it}, g_t)$:

- **Expected profit:**

$$J_1^R = \mathbb{E}_{D_t, b_{it}, p_{it}} \left[ \sum_{t=1}^{24} \Pi_t^R \right].$$

- **Worst-case profit (robustness):**

$$J_2^R = \min_{g_t \in [\underline{g}_t, \bar{g}_t]} \mathbb{E}_{D_t, b_{it}, p_{it}} \left[ \sum_{t=1}^{24} \Pi_t^R \right].$$

#### 6. Constraints

$$0 \le b_{Rt} \le \bar{b}_t, \quad p_{Rt} \ge 0.$$

#### 7. Treatment of Exogenous Inputs

The renewable producer treats:

- Regulator controls $c_t^{\mathrm{under}}, c_t^{\mathrm{over}}$ as **fixed parameters**.
- Market demand $D_t$ and competitors' bids $(b_{it}, p_{it})$ as **random inputs**.

---

### (B) Module 2: System Regulator Submodel

#### 1. Control Variables

| Symbol | Units | Description |
|---|---|---|
| $c_t^{\mathrm{under}}$ | €/MWh | Under-delivery penalty rate |
| $c_t^{\mathrm{over}}$ | €/MWh | Over-delivery penalty rate |

#### 2. State and Uncertainties

| Symbol | Description |
|---|---|
| Shared state $s_t = (D_t, \pi_t, Q_t, g_t)$ | Observed from environment |
| $D_t, b_{it}, p_{it}, g_t$ | As defined in environment |

#### 3. System-Level Metrics (Derived from Environment)

$$\begin{aligned}&\text{Price variance: } \mathrm{Var}(\pi_t),\\&\text{Expected imbalance: } \mathbb{E}[|Q_t - D_t|],\\&\text{Expected total cost: } \mathbb{E}[\pi_t D_t].\end{aligned}$$

**4. Objectives**

- **Price stability (minimize volatility):**

$$J_1^{\mathrm{Reg}} = -\mathrm{Var}(\pi_t).$$

- **Reliability (minimize imbalance):**

$$J_2^{\mathrm{Reg}} = -\sum_{t=1}^{24} \mathbb{E}[|Q_t - D_t|].$$

- **Cost containment (minimize expected total payout):**

$$J_3^{\mathrm{Reg}} = -\sum_{t=1}^{24} \mathbb{E}[\pi_t D_t].$$

All quantities are computable from simulated or real market data.

**5. Constraints**

$$0 \le c_t^{\mathrm{under}}, c_t^{\mathrm{over}} \le c_{\max}, \quad \mathbb{E}[|Q_t - D_t|] \le L_t.$$

**6. Treatment of Exogenous Inputs**

The regulator treats:

- Renewable bids $(b_{Rt}, p_{Rt})$ as **exogenous**.
- Market outcomes $(\pi_t, Q_t)$ as **functions of the bids and demand**.

---

# 🔗 3. Coupling and Interaction

Both modules share the **same environment** but influence it through different channels:

| Coupling Variable | Source | Affects |
|---|---|---|
| $\pi_t$ | Market clearing from all bids | Both modules (price in profit and system metrics) |
| $Q_t$ | Sum of accepted bids | Regulator's reliability measure |
| $c_t^{\mathrm{under}}, c_t^{\mathrm{over}}$ | Regulator's control | Producer's imbalance penalties |
| $b_{Rt}, p_{Rt}$ | Renewable's control | Market clearing price, regulator outcomes |

Interaction is **indirect**, consistent with the assumption that each agent treats the other's controls as exogenous when optimizing its own objective.

---

# 🧮 4. Unified System Definition

We now express the unified model as a structured composition.

---

### (A) Environment (shared)

$$
\begin{aligned}
&D_t \sim \mathcal{N}(\mu_D, \sigma_D^2), \\
&b_{it} \sim \mathcal{N}(\mu_{b_i}, \sigma_{b_i}^2), \quad p_{it} \sim \mathcal{N}(\mu_{p_i}, \sigma_{p_i}^2), \quad i = 1,2,3, \\
&g_t \in [\underline{g}_t, \bar{g}_t], \\
&\pi_t = \min \left\{ p : \sum_{i: p_{it} \le p} b_{it} \ge D_t \right\}, \\
&Q_t = \sum_{i: p_{it} \le \pi_t} b_{it}.
\end{aligned}
$$

---

### (B) Renewable Producer Module

$$
\begin{aligned}
\max_{b_{Rt}, p_{Rt}} \quad & \{J_1^R, J_2^R\} \\
\text{s.t.} \quad & 0 \le b_{Rt} \le \bar{b}_t, \; p_{Rt} \ge 0, \\
& \text{Given exogenous } (c_t^{\text{under}}, c_t^{\text{over}}).
\end{aligned}
$$

---

### (C) System Regulator Module

$$
\begin{aligned}
\max_{c_t^{\text{under}}, c_t^{\text{over}}} \quad & \{J_1^{\text{Reg}}, J_2^{\text{Reg}}, J_3^{\text{Reg}}\} \\
\text{s.t.} \quad & 0 \le c_t^{\text{under}}, c_t^{\text{over}} \le c_{\max}, \\
& \mathbb{E}[|Q_t - D_t|] \le L_t, \\
& \text{Given exogenous } (b_{Rt}, p_{Rt}).
\end{aligned}
$$

---

### (D) Full System Composition

$$
\boxed{
\begin{cases}
\textbf{Environment:} & (D_t, b_{it}, p_{it}, g_t, \pi_t, Q_t), \\
\textbf{Renewable module:} & \text{Chooses } (b_{Rt}, p_{Rt}) \text{ to optimize } J_1^R, J_2^R, \\
\textbf{Regulator module:} & \text{Chooses } (c_t^{\text{under}}, c_t^{\text{over}}) \text{ to optimize } J_1^{\text{Reg}}, J_2^{\text{Reg}}, J_3^{\text{Reg}}, \\
\textbf{Coupling:} & \pi_t = \text{MarketClearing}(b_{it}, p_{it}, b_{Rt}, p_{Rt}, D_t), \\
& \text{each treats the other's controls as exogenous.}
\end{cases}
}
$$

---

# ✅ 5. Key Properties of the Unified Model

| Property | Description |
|---|---|
| **Single environment** | Common stochastic–robust market with shared variables $(\pi_t, Q_t, D_t, g_t)$. |
| **Modular submodels** | Each stakeholder has its own control and objective set, evaluated independently. |
| **Exogenous treatment of others' controls** | Each module's optimization assumes the other's parameters fixed. |
| **Quantifiable and implementable** | All elements are either measurable, simulated, or directly computable. |
| **Expandable** | Additional producers or regulators can be added by extending the control sets. |

## In short:

This unified model defines a **shared stochastic–robust market system** with

- **renewable producer controls** $(b_{Rt}, p_{Rt})$,
- **regulator controls** $(c_t^{\text{under}}, c_t^{\text{over}})$,
- **endogenous shared outcomes** $(\pi_t, Q_t, \text{imb}_t)$,
  and modular, directly evaluable **objectives** for each perspective.

Would you like me to now formalize this unified model as a **bilevel or simultaneous game formulation** (e.g., Stackelberg or Nash structure) to show how these submodules interact strategically?

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.

Here's a **modular, implementable Python representation** of the unified model.
It uses classes to represent each stakeholder's perspective (renewable producer and regulator) and a shared environment class that simulates market outcomes.

At the end, a **single interface function** ties everything together:
it accepts a dictionary of **control variables** and **uncertain parameters**, simulates the unified model, and returns **all objective function values** as outputs.

The code is designed for clarity and modularity (not for execution here).

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

```python
# ============================================================
# 1. Shared Environment
# ============================================================

@dataclass
class MarketEnvironment:
    """Shared market environment."""
    T: int = 24                          # number of hours
    mu_D: float = 1000.0                 # mean demand (MWh)
    sigma_D: float = 100.0               # std dev of demand
    mu_b: List[float] = field(default_factory=lambda: [400, 350, 300])
    sigma_b: List[float] = field(default_factory=lambda: [50, 40, 30])
    mu_p: List[float] = field(default_factory=lambda: [40, 45, 50])
    sigma_p: List[float] = field(default_factory=lambda: [5, 5, 5])
    g_bounds: List[Tuple[float, float]] = field(default_factory=lambda: [(0, 500)] * 24)

    # --- simulation state variables ---
    D_t: np.ndarray = field(init=False)
    bids_conv: List[Tuple[np.ndarray, np.ndarray]] = field(init=False)
    g_t: np.ndarray = field(init=False)

    def sample_uncertainties(self):
        """Sample random demand and competitor bids."""
        self.D_t = np.random.normal(self.mu_D, self.sigma_D, self.T)
        self.bids_conv = []
        for i in range(3):  # 3 conventional producers
            b_i = np.random.normal(self.mu_b[i], self.sigma_b[i], self.T)
            p_i = np.random.normal(self.mu_p[i], self.sigma_p[i], self.T)
            self.bids_conv.append((b_i, p_i))

    def sample_generation(self, mode: str = "midpoint"):
        """Select renewable generation within uncertainty bounds."""
        low, high = np.array([gb[0] for gb in self.g_bounds]), np.array([gb[1] for gb in
self.g_bounds])
        if mode == "worst_case":
            self.g_t = low
        elif mode == "best_case":
            self.g_t = high
        else:
            self.g_t = 0.5 * (low + high)

    def clear_market(self, b_Rt: np.ndarray, p_Rt: np.ndarray) -> Tuple[np.ndarray, np.ndarray,
np.ndarray]:
        """Compute clearing prices, acceptance indicators, and total supply."""
        pi_t, delta_t, Q_t = np.zeros(self.T), np.zeros(self.T), np.zeros(self.T)

        for t in range(self.T):
            # Aggregate bids: conventional + renewable
            bids = []
            for i in range(3):
                b_i, p_i = self.bids_conv[i][0][t], self.bids_conv[i][1][t]
                bids.append((p_i, b_i))
            bids.append((p_Rt[t], b_Rt[t]))  # renewable bid
            bids.sort(key=lambda x: x[0])    # sort by price

            demand = self.D_t[t]
            cumulative = 0.0
            clearing_price = bids[-1][0]
            accepted_supply = 0.0
            for price, qty in bids:
                cumulative += qty
                accepted_supply += qty
                if cumulative >= demand:
                    clearing_price = price
                    break

            # Compute acceptance for renewable producer
            pi_t[t] = clearing_price
            delta_t[t] = 1.0 if p_Rt[t] <= pi_t[t] else 0.0
            Q_t[t] = accepted_supply

        return pi_t, delta_t, Q_t


# ============================================================
```

```
# 2. Renewable Producer Module
# ============================================================

@dataclass
class RenewableProducer:
    """Renewable producer perspective."""
    env: MarketEnvironment
    b_Rt: np.ndarray
    p_Rt: np.ndarray
    c_under: np.ndarray
    c_over: np.ndarray
    b_max: float = 500.0

    def compute_profit(self, pi_t, delta_t, g_t) -> np.ndarray:
        """Compute hourly profits."""
        under_penalty = self.c_under * np.maximum(self.b_Rt - g_t, 0)
        over_penalty = self.c_over * np.maximum(g_t - self.b_Rt, 0)
        profit = delta_t * (pi_t * self.b_Rt - under_penalty - over_penalty)
        return profit

    def objective_expected_profit(self, pi_t, delta_t, g_t) -> float:
        """Expected profit under given market outcomes."""
        profit = self.compute_profit(pi_t, delta_t, g_t)
        return np.mean(np.sum(profit))

    def objective_worst_case_profit(self, pi_t, delta_t) -> float:
        """Worst-case profit under deep uncertainty (lower bound of generation)."""
        low = np.array([gb[0] for gb in self.env.g_bounds])
        profit = self.compute_profit(pi_t, delta_t, low)
        return np.sum(profit)


# ============================================================
# 3. System Regulator Module
# ============================================================

@dataclass
class SystemRegulator:
    """System regulator perspective."""
    env: MarketEnvironment
    c_under: np.ndarray
    c_over: np.ndarray
    L_t: np.ndarray
    c_max: float = 200.0

    def evaluate_metrics(self, pi_t, Q_t):
        """Compute measurable system-level metrics."""
        # Price variance
        J1 = -np.var(pi_t)
        # Expected imbalance
        imbalance = np.abs(Q_t - self.env.D_t)
        J2 = -np.mean(imbalance)
        # Expected total cost
        J3 = -np.mean(pi_t * self.env.D_t)
        return J1, J2, J3


# ============================================================
# 4. Unified Model Interface
# ============================================================

def evaluate_unified_model(
    controls: Dict[str, np.ndarray],
    uncertainties: Dict[str, Dict]
) -> Dict[str, float]:
    """
    Interface for unified model evaluation.
    Inputs:
        controls: dictionary with control arrays:
            {
              'b_Rt': np.array(24),
              'p_Rt': np.array(24),
              'c_under': np.array(24),
              'c_over': np.array(24)
            }
```

```
        uncertainties: dictionary with uncertain parameter values:
            {
              'mu_D': float, 'sigma_D': float,
              'mu_b': list, 'sigma_b': list,
              'mu_p': list, 'sigma_p': list,
              'g_bounds': list of (low, high)
            }
    Output:
        dict of objective function values:
            {
              'J1_R': expected_profit,
              'J2_R': worst_case_profit,
              'J1_Reg': price_stability,
              'J2_Reg': reliability,
              'J3_Reg': cost_containment
            }
    """

    # ---- Initialize shared environment ----
    env = MarketEnvironment(
        mu_D=uncertainties['mu_D'],
        sigma_D=uncertainties['sigma_D'],
        mu_b=uncertainties['mu_b'],
        sigma_b=uncertainties['sigma_b'],
        mu_p=uncertainties['mu_p'],
        sigma_p=uncertainties['sigma_p'],
        g_bounds=uncertainties['g_bounds']
    )

    env.sample_uncertainties()
    env.sample_generation(mode="midpoint")

    # ---- Regulator and Producer modules ----
    regulator = SystemRegulator(env, controls['c_under'], controls['c_over'], L_t=np.ones(24) *
10.0)
    producer = RenewableProducer(env, controls['b_Rt'], controls['p_Rt'],
                                 controls['c_under'], controls['c_over'])

    # ---- Market simulation ----
    pi_t, delta_t, Q_t = env.clear_market(producer.b_Rt, producer.p_Rt)

    # ---- Renewable objectives ----
    J1_R = producer.objective_expected_profit(pi_t, delta_t, env.g_t)
    J2_R = producer.objective_worst_case_profit(pi_t, delta_t)

    # ---- Regulator objectives ----
    J1_Reg, J2_Reg, J3_Reg = regulator.evaluate_metrics(pi_t, Q_t)

    return {
        'J1_R': float(J1_R),
        'J2_R': float(J2_R),
        'J1_Reg': float(J1_Reg),
        'J2_Reg': float(J2_Reg),
        'J3_Reg': float(J3_Reg)
    }
```

---

## ✅ Explanation

### 1. Modular class structure

- **`MarketEnvironment`**: handles stochastic demand, competitor bids, renewable generation, and price clearing.
- **`RenewableProducer`**: contains the renewable agent's decisions and profit objectives.
- **`SystemRegulator`**: contains regulator policy parameters and measurable system objectives.

### 2. Shared environment

- Both modules read and write to the same `MarketEnvironment` instance.
- Clearing prices `π_t` and accepted quantities `Q_t` couple the two perspectives.

**3. Unified interface**

- `evaluate_unified_model()` is the main entry point.
  It:
  - Accepts control dictionaries (both stakeholder controls),
  - Accepts uncertain environment parameters,
  - Simulates the market,
  - Returns all objective values as a dictionary.

**4. Implementable and testable**

- Every component is numeric and directly computable with real or simulated data.
- The structure supports modular evaluation of each perspective independently.

---

Would you like me to extend this with a small **example dictionary** of control and uncertainty inputs (for clarity on how to call it)?

mu_D, sigma_D, mu_b, sigma_b, mu_p, sigma_p are constants instead of uncertain parameters, so the implementation should not take them as inputs. Please refine the implementation.

Perfect — that's a precise and important refinement.
We'll make the environment's statistical parameters (`mu_D`, `sigma_D`, `mu_b`, `sigma_b`, `mu_p`, `sigma_p`) **fixed constants**, defined within the model rather than passed as inputs.

The only inputs to the interface will then be:

- **Control variables:** renewable and regulator decisions, and
- **Uncertainty parameters:** renewable generation bounds (`g_bounds`).

Everything else remains modular, quantifiable, and fully consistent.

Below is the **refined unified Python implementation**.

---

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


# ============================================================
# 1. Shared Environment
# ============================================================

@dataclass
class MarketEnvironment:
```

```
    """Shared market environment with fixed stochastic parameters."""
    T: int = 24                          # number of hours
    # --- Fixed constants ---
    mu_D: float = 1000.0                 # mean demand (MWh)
    sigma_D: float = 100.0               # std. deviation of demand
    mu_b: List[float] = field(default_factory=lambda: [400, 350, 300])
    sigma_b: List[float] = field(default_factory=lambda: [50, 40, 30])
    mu_p: List[float] = field(default_factory=lambda: [40, 45, 50])
    sigma_p: List[float] = field(default_factory=lambda: [5, 5, 5])
    g_bounds: List[Tuple[float, float]] = field(default_factory=lambda: [(0, 500)] * 24)

    # --- simulation state variables (initialized later) ---
    D_t: np.ndarray = field(init=False)
    bids_conv: List[Tuple[np.ndarray, np.ndarray]] = field(init=False)
    g_t: np.ndarray = field(init=False)

    def sample_uncertainties(self):
        """Sample random demand and competitor bids."""
        self.D_t = np.random.normal(self.mu_D, self.sigma_D, self.T)
        self.bids_conv = []
        for i in range(3):  # 3 conventional producers
            b_i = np.random.normal(self.mu_b[i], self.sigma_b[i], self.T)
            p_i = np.random.normal(self.mu_p[i], self.sigma_p[i], self.T)
            self.bids_conv.append((b_i, p_i))

    def sample_generation(self, mode: str = "midpoint"):
        """Select renewable generation within uncertainty bounds."""
        low, high = np.array([gb[0] for gb in self.g_bounds]), np.array([gb[1] for gb in
self.g_bounds])
        if mode == "worst_case":
            self.g_t = low
        elif mode == "best_case":
            self.g_t = high
        else:
            self.g_t = 0.5 * (low + high)

    def clear_market(self, b_Rt: np.ndarray, p_Rt: np.ndarray) -> Tuple[np.ndarray, np.ndarray,
np.ndarray]:
        """Compute clearing prices, acceptance indicators, and total accepted supply."""
        pi_t, delta_t, Q_t = np.zeros(self.T), np.zeros(self.T), np.zeros(self.T)

        for t in range(self.T):
            # Combine conventional and renewable bids
            bids = []
            for i in range(3):
                b_i, p_i = self.bids_conv[i][0][t], self.bids_conv[i][1][t]
                bids.append((p_i, b_i))
            bids.append((p_Rt[t], b_Rt[t]))
            bids.sort(key=lambda x: x[0])  # sort by offer price

            demand = self.D_t[t]
            cumulative = 0.0
            clearing_price = bids[-1][0]
            accepted_supply = 0.0
            for price, qty in bids:
                cumulative += qty
                accepted_supply += qty
                if cumulative >= demand:
                    clearing_price = price
                    break

            pi_t[t] = clearing_price
            delta_t[t] = 1.0 if p_Rt[t] <= pi_t[t] else 0.0
            Q_t[t] = accepted_supply

        return pi_t, delta_t, Q_t


# ============================================================
# 2. Renewable Producer Module
# ============================================================

@dataclass
class RenewableProducer:
    """Renewable producer perspective."""
```

```python
    env: MarketEnvironment
    b_Rt: np.ndarray
    p_Rt: np.ndarray
    c_under: np.ndarray
    c_over: np.ndarray
    b_max: float = 500.0

    def compute_profit(self, pi_t, delta_t, g_t) -> np.ndarray:
        """Compute hourly profits."""
        under_penalty = self.c_under * np.maximum(self.b_Rt - g_t, 0)
        over_penalty = self.c_over * np.maximum(g_t - self.b_Rt, 0)
        profit = delta_t * (pi_t * self.b_Rt - under_penalty - over_penalty)
        return profit

    def objective_expected_profit(self, pi_t, delta_t, g_t) -> float:
        """Expected profit under given outcomes."""
        profit = self.compute_profit(pi_t, delta_t, g_t)
        return np.sum(profit)

    def objective_worst_case_profit(self, pi_t, delta_t) -> float:
        """Worst-case profit under deep uncertainty."""
        low = np.array([gb[0] for gb in self.env.g_bounds])
        profit = self.compute_profit(pi_t, delta_t, low)
        return np.sum(profit)


# ============================================================
# 3. System Regulator Module
# ============================================================

@dataclass
class SystemRegulator:
    """System regulator perspective."""
    env: MarketEnvironment
    c_under: np.ndarray
    c_over: np.ndarray
    L_t: np.ndarray
    c_max: float = 200.0

    def evaluate_metrics(self, pi_t, Q_t):
        """Compute measurable regulator objectives."""
        # Price stability (negative variance)
        J1 = -np.var(pi_t)
        # Reliability (negative expected imbalance)
        J2 = -np.mean(np.abs(Q_t - self.env.D_t))
        # Cost containment (negative expected total cost)
        J3 = -np.mean(pi_t * self.env.D_t)
        return J1, J2, J3


# ============================================================
# 4. Unified Model Interface
# ============================================================

def evaluate_unified_model(
    controls: Dict[str, np.ndarray],
    uncertainties: Dict[str, List[Tuple[float, float]]]
) -> Dict[str, float]:
    """
    Unified model interface.

    Inputs:
        controls: dict of control variables
            {
              'b_Rt': np.array(24),
              'p_Rt': np.array(24),
              'c_under': np.array(24),
              'c_over': np.array(24)
            }
        uncertainties: dict of uncertain parameters
            {
              'g_bounds': list of (low, high)
            }

    Output:
```

```
        dict of objective function values:
            {
              'J1_R': expected_profit,
              'J2_R': worst_case_profit,
              'J1_Reg': price_stability,
              'J2_Reg': reliability,
              'J3_Reg': cost_containment
            }
    """

    # ---- Initialize shared environment ----
    env = MarketEnvironment(g_bounds=uncertainties['g_bounds'])
    env.sample_uncertainties()
    env.sample_generation(mode="midpoint")

    # ---- Regulator and Producer modules ----
    regulator = SystemRegulator(env, controls['c_under'], controls['c_over'], L_t=np.ones(24) *
10.0)
    producer = RenewableProducer(env, controls['b_Rt'], controls['p_Rt'],
                                 controls['c_under'], controls['c_over'])

    # ---- Market simulation ----
    pi_t, delta_t, Q_t = env.clear_market(producer.b_Rt, producer.p_Rt)

    # ---- Renewable objectives ----
    J1_R = producer.objective_expected_profit(pi_t, delta_t, env.g_t)
    J2_R = producer.objective_worst_case_profit(pi_t, delta_t)

    # ---- Regulator objectives ----
    J1_Reg, J2_Reg, J3_Reg = regulator.evaluate_metrics(pi_t, Q_t)

    return {
        'J1_R': float(J1_R),
        'J2_R': float(J2_R),
        'J1_Reg': float(J1_Reg),
        'J2_Reg': float(J2_Reg),
        'J3_Reg': float(J3_Reg)
    }
```

## ✅ Implementation Notes

### 1. Fixed constants:

- `mu_D`, `sigma_D`, `mu_b`, `sigma_b`, `mu_p`, `sigma_p` are **embedded in `MarketEnvironment`** and not provided by the user.
- They can easily be adjusted in the class definition if needed.

### 2. Input dictionaries:

```python
controls = {
    'b_Rt': np.array([...]),       # renewable bid quantities (24 values)
    'p_Rt': np.array([...]),       # renewable offer prices (24 values)
    'c_under': np.array([...]),    # regulator under-delivery penalties
    'c_over': np.array([...])      # regulator over-delivery penalties
}

uncertainties = {
    'g_bounds': [(low_t, high_t) for t in range(24)]  # renewable generation bounds
}
```

### 3. Output:

A dictionary with all quantifiable objective function values:

```python
{
  'J1_R': expected_profit,
  'J2_R': worst_case_profit,
  'J1_Reg': price_stability,
  'J2_Reg': reliability,
  'J3_Reg': cost_containment
}
```

**4. Design properties:**

- Modular classes → each stakeholder is encapsulated.
- Shared environment ensures consistent state and coupling.
- The interface `evaluate_unified_model()` enables easy external use in optimization, simulation, or policy analysis.

---

Would you like me to extend this implementation with **example input dictionaries** (showing sample numerical data) to demonstrate how to call the interface?

The function objective_worst_case_profit is not implemented correctly. The worst case does not necessarily occur at the lower bound of g. Please correct this.

Excellent observation — and absolutely correct.

The **worst-case profit** for the renewable producer does **not necessarily occur at the lower bound** of generation $g_t$, because the penalty is two-sided:

$$C_t(g_t, b_t) = c_t^{\text{under}}[b_t - g_t]_+ + c_t^{\text{over}}[g_t - b_t]_+.$$

If $c_t^{\text{over}}$ is large and $g_t > b_t$, over-delivery could yield lower profit than an under-delivery shortfall.
So, the correct implementation must check **both bounds** of $g_t$ (and optionally the bid itself, where the penalty function changes slope) to find the **minimum profit** within the feasible interval.

Below is the **refined and fully correct version** of the unified model.
Only the `objective_worst_case_profit()` method has changed, while the rest remains consistent.

---

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


# ============================================================
# 1. Shared Environment
# ============================================================

@dataclass
class MarketEnvironment:
    """Shared market environment with fixed stochastic parameters."""
    T: int = 24
    mu_D: float = 1000.0
```

```
    sigma_D: float = 100.0
    mu_b: List[float] = field(default_factory=lambda: [400, 350, 300])
    sigma_b: List[float] = field(default_factory=lambda: [50, 40, 30])
    mu_p: List[float] = field(default_factory=lambda: [40, 45, 50])
    sigma_p: List[float] = field(default_factory=lambda: [5, 5, 5])
    g_bounds: List[Tuple[float, float]] = field(default_factory=lambda: [(0, 500)] * 24)

    D_t: np.ndarray = field(init=False)
    bids_conv: List[Tuple[np.ndarray, np.ndarray]] = field(init=False)
    g_t: np.ndarray = field(init=False)

    def sample_uncertainties(self):
        """Sample stochastic demand and competitor bids."""
        self.D_t = np.random.normal(self.mu_D, self.sigma_D, self.T)
        self.bids_conv = []
        for i in range(3):
            b_i = np.random.normal(self.mu_b[i], self.sigma_b[i], self.T)
            p_i = np.random.normal(self.mu_p[i], self.sigma_p[i], self.T)
            self.bids_conv.append((b_i, p_i))

    def sample_generation(self, mode: str = "midpoint"):
        """Select generation scenario."""
        low, high = np.array([gb[0] for gb in self.g_bounds]), np.array([gb[1] for gb in
self.g_bounds])
        if mode == "worst_case":
            self.g_t = low
        elif mode == "best_case":
            self.g_t = high
        else:
            self.g_t = 0.5 * (low + high)

    def clear_market(self, b_Rt: np.ndarray, p_Rt: np.ndarray) -> Tuple[np.ndarray, np.ndarray,
np.ndarray]:
        """Compute market clearing price, acceptance, and supply."""
        pi_t, delta_t, Q_t = np.zeros(self.T), np.zeros(self.T), np.zeros(self.T)
        for t in range(self.T):
            bids = []
            for i in range(3):
                b_i, p_i = self.bids_conv[i][0][t], self.bids_conv[i][1][t]
                bids.append((p_i, b_i))
            bids.append((p_Rt[t], b_Rt[t]))
            bids.sort(key=lambda x: x[0])

            demand = self.D_t[t]
            cumulative = 0.0
            clearing_price = bids[-1][0]
            accepted_supply = 0.0
            for price, qty in bids:
                cumulative += qty
                accepted_supply += qty
                if cumulative >= demand:
                    clearing_price = price
                    break

            pi_t[t] = clearing_price
            delta_t[t] = 1.0 if p_Rt[t] <= pi_t[t] else 0.0
            Q_t[t] = accepted_supply
        return pi_t, delta_t, Q_t


# ============================================================
# 2. Renewable Producer Module
# ============================================================

@dataclass
class RenewableProducer:
    """Renewable producer perspective."""
    env: MarketEnvironment
    b_Rt: np.ndarray
    p_Rt: np.ndarray
    c_under: np.ndarray
    c_over: np.ndarray
    b_max: float = 500.0

    def compute_profit(self, pi_t, delta_t, g_t) -> np.ndarray:
```

```
        """Compute hourly profit for given generation."""
        under_penalty = self.c_under * np.maximum(self.b_Rt - g_t, 0)
        over_penalty = self.c_over * np.maximum(g_t - self.b_Rt, 0)
        profit = delta_t * (pi_t * self.b_Rt - under_penalty - over_penalty)
        return profit

    def objective_expected_profit(self, pi_t, delta_t, g_t) -> float:
        """Expected profit for realized generation."""
        profit = self.compute_profit(pi_t, delta_t, g_t)
        return np.sum(profit)

    def objective_worst_case_profit(self, pi_t, delta_t) -> float:
        """
        Worst-case profit under deep uncertainty in g_t.

        For each hour, the profit is piecewise linear in g_t with slope change at g_t = b_Rt.
        The minimum therefore occurs at one of {lower bound, upper bound, b_Rt (if within
bounds)}.
        """
        profit_worst = np.zeros(self.env.T)
        for t in range(self.env.T):
            g_candidates = []
            low, high = self.env.g_bounds[t]
            # candidate points
            g_candidates.append(low)
            g_candidates.append(high)
            if low <= self.b_Rt[t] <= high:
                g_candidates.append(self.b_Rt[t])
            # evaluate profits at candidate g values
            profits = []
            for g_val in g_candidates:
                under_penalty = self.c_under[t] * max(self.b_Rt[t] - g_val, 0)
                over_penalty = self.c_over[t] * max(g_val - self.b_Rt[t], 0)
                p = delta_t[t] * (pi_t[t] * self.b_Rt[t] - under_penalty - over_penalty)
                profits.append(p)
            profit_worst[t] = min(profits)
        return np.sum(profit_worst)


# ============================================================
# 3. System Regulator Module
# ============================================================

@dataclass
class SystemRegulator:
    """System regulator perspective."""
    env: MarketEnvironment
    c_under: np.ndarray
    c_over: np.ndarray
    L_t: np.ndarray
    c_max: float = 200.0

    def evaluate_metrics(self, pi_t, Q_t):
        """Compute measurable regulator objectives."""
        J1 = -np.var(pi_t)                         # price stability
        J2 = -np.mean(np.abs(Q_t - self.env.D_t))  # reliability
        J3 = -np.mean(pi_t * self.env.D_t)         # cost containment
        return J1, J2, J3


# ============================================================
# 4. Unified Model Interface
# ============================================================

def evaluate_unified_model(
    controls: Dict[str, np.ndarray],
    uncertainties: Dict[str, List[Tuple[float, float]]]
) -> Dict[str, float]:
    """
    Unified model interface.

    Inputs:
        controls:
            {
              'b_Rt': np.array(24),
```

```
            'p_Rt': np.array(24),
            'c_under': np.array(24),
            'c_over': np.array(24)
          }
      uncertainties:
          {
            'g_bounds': list of (low, high)
          }

    Output:
      dict of objective function values:
          {
            'J1_R': expected_profit,
            'J2_R': worst_case_profit,
            'J1_Reg': price_stability,
            'J2_Reg': reliability,
            'J3_Reg': cost_containment
          }
    """

    # ---- Initialize shared environment ----
    env = MarketEnvironment(g_bounds=uncertainties['g_bounds'])
    env.sample_uncertainties()
    env.sample_generation(mode="midpoint")

    # ---- Instantiate modules ----
    regulator = SystemRegulator(env, controls['c_under'], controls['c_over'], L_t=np.ones(24) *
10.0)
    producer = RenewableProducer(env, controls['b_Rt'], controls['p_Rt'],
                                 controls['c_under'], controls['c_over'])

    # ---- Market simulation ----
    pi_t, delta_t, Q_t = env.clear_market(producer.b_Rt, producer.p_Rt)

    # ---- Renewable objectives ----
    J1_R = producer.objective_expected_profit(pi_t, delta_t, env.g_t)
    J2_R = producer.objective_worst_case_profit(pi_t, delta_t)

    # ---- Regulator objectives ----
    J1_Reg, J2_Reg, J3_Reg = regulator.evaluate_metrics(pi_t, Q_t)

    return {
        'J1_R': float(J1_R),
        'J2_R': float(J2_R),
        'J1_Reg': float(J1_Reg),
        'J2_Reg': float(J2_Reg),
        'J3_Reg': float(J3_Reg)
    }
```

---

## ✅ Explanation of the Correction

The **worst-case profit** now correctly checks the potential minimizers of the profit function for each hour:

$$g_t \in \{\underline{g}_t, \bar{g}_t, b_{Rt}\}$$

(but only include $b_{Rt}$ if it lies within the feasible range).

The hourly profit function is **piecewise linear** with a kink at $g_t = b_{Rt}$:

- For $g_t < b_{Rt}$: slope = $+c_t^{\text{under}}$
- For $g_t > b_{Rt}$: slope = $-c_t^{\text{over}}$

Therefore, the minimum occurs at one of these breakpoints.
This new method correctly evaluates all candidate points and takes the smallest profit (worst case).

---

## ✅ Result

The implementation is now:

- Fully **quantifiable**,
- **Consistent** with the stochastic–robust unified model, and
- **Correctly handles deep uncertainty** in renewable generation.

---

Would you like me to extend this with a short **example usage snippet** showing how to call `evaluate_unified_model()` with mock control and uncertainty data?