# Market_brief_3

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

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

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

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

Below is a **formal simulation-model specification** for evaluating the **expected revenue of a wind-power producer** participating in a competitive day-ahead electricity spot market under uncertainty. The formulation is deliberately modular so it can support Monte-Carlo simulation, scenario analysis, and later extensions (e.g., storage, imbalance penalties, learning).

---

## 1. Modeling Framework

We model the problem as a **stochastic decision-based simulation** with hourly resolution over a single day (24 periods), embedded in a scenario generator that captures uncertainty in demand, renewable availability, and competitors' bids.

Time index:

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

Agents:

- $i \in \mathcal{I} = \{1, \dots, 5\}$

- Wind producer: $i = w$
- Solar producer: $i = s$
- Conventional producers: $i \in \mathcal{C}$

---

## 2. State Variables

State variables describe information known at bidding time or realized during dispatch.

### Market State

- $D_t$: total system demand (MWh)
- $\mathcal{B}_t = \{(q_{i,t}, p_{i,t})\}_{i \in \mathcal{I}}$: set of submitted bids
- $\pi_t$: market-clearing price (€/MWh)
- $x_{i,t}$: accepted dispatch quantity of producer $i$ (MWh)

### Wind Producer State

- $A_{w,t}$: available wind generation capacity (MWh)
- $C_{w,t}$: realized wind generation (MWh)
- $\omega_t$: information set at time of bidding (forecasts, historical data)

---

## 3. Decision Variables (Wind Producer)

The wind producer chooses bids **before uncertainties are realized**.

For each hour $t$:

- $q_{w,t} \geq 0$: bid quantity (MWh)
- $p_{w,t} \geq 0$: bid price (€/MWh)

Decision vector:

$$\mathbf{u}_w = \{(q_{w,t}, p_{w,t})\}_{t=1}^{24}$$

---

## 4. Stochastic Variables (Short-Term Uncertainty)

These variables are sampled within the simulation.

### Demand

- $D_t \sim \mathcal{D}_t(\theta_D)$

### Renewable Availability

- Wind availability:

$$A_{w,t} \sim \mathcal{W}_t(\theta_W)$$

- Solar availability (affects market price):

$$A_{s,t} \sim \mathcal{S}_t(\theta_S)$$

### Competitor Behavior

- Conventional bids:

$$(q_{i,t}, p_{i,t}) \sim \mathcal{B}_{i,t}(\theta_i), \quad i \in \mathcal{C}$$

---

## 5. Constant Parameters

### Technical Parameters

- $\overline{Q}_{w,t}$: installed wind capacity
- $c_i$: marginal production cost of producer $i$
- $\Delta t = 1$ hour

### Market Rules

- Merit-order clearing
- Uniform pricing
- No strategic withholding beyond bids
- No network constraints (single-node market)

---

## 6. Market-Clearing (Transition Function)

For each hour $t$:

1. **Bid Ordering (Merit Order)**
   Sort all bids by increasing price:

$$p_{(1),t} \leq p_{(2),t} \leq \dots$$

2. **Dispatch Rule**

$$\sum_i x_{i,t} = D_t, \quad 0 \leq x_{i,t} \leq q_{i,t}$$

3. **Market Price**

$$\pi_t = \min\{p_{i,t} : \sum_{j:p_{j,t} \leq p_{i,t}} q_{j,t} \geq D_t\}$$

4. **Wind Dispatch**

$$x_{w,t} = \begin{cases} \min(q_{w,t}, D_t - \sum_{j:p_{j,t}<p_{w,t}} q_{j,t}) & \text{if accepted} \\ 0 & \text{otherwise} \end{cases}$$

5. **Feasibility Check**

$$x_{w,t} \leq C_{w,t} = \min(A_{w,t}, \overline{Q}_{w,t})$$

---

# 7. Revenue and Objective Function

## Hourly Revenue

$$R_{w,t} = \pi_t \cdot x_{w,t}$$

(Optionally include imbalance penalties if $x_{w,t} > C_{w,t}$)

## Daily Revenue

$$R_w = \sum_{t=1}^{24} R_{w,t}$$

## Objective

$$\max_{\mathbf{u}_w} \ \mathbb{E}[R_w]$$

Expectation is taken over all stochastic variables:

$$(D_t, A_{w,t}, A_{s,t}, \mathcal{B}_{\mathcal{C},t})$$

---

# 8. Simulation Structure

1. Sample stochastic variables for a day
2. Apply wind bidding strategy $\mathbf{u}_w$
3. Clear market hour-by-hour
4. Compute realized revenue
5. Repeat across Monte-Carlo runs
6. Estimate expected revenue and risk metrics (e.g., variance, CVaR)

---

# 9. Deep Uncertainties (Long-Term / Structural)

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

## Market Design

- Changes in pricing rules (pay-as-bid vs uniform)
- Introduction of capacity markets
- Renewable priority dispatch rules

### Technology & Cost

- Future wind turbine efficiency
- Capital costs and degradation rates
- Storage integration

### Policy & Regulation

- Carbon pricing levels
- Renewable subsidies or curtailment rules

### Strategic Behavior

- Entry/exit of competitors
- Strategic bidding evolution
- Market power concentration

---

## 10. Outputs of Interest

- Expected daily revenue
- Revenue distribution (risk)
- Price-duration curves
- Sensitivity to bid price/quantity
- Scenario-dependent investment viability

---

### Summary

This formalization defines a **stochastic, agent-based market simulation** where the wind producer optimizes bidding decisions under uncertainty. The framework is suitable for:

- Monte-Carlo simulation
- Robust decision-making
- Investment viability assessment under deep uncertainty

If you want, I can next:

- Convert this into a **mathematical program**
- Provide **pseudo-code**
- Extend it to **multi-day learning or storage**

- Add **risk-averse objectives (CVaR)**

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

Below is a **refined and more explicit simulation-model specification** incorporating the assumption that **market demand follows a Normal distribution with constant mean and variance**. I keep the structure aligned with your original formulation and make only the necessary refinements.

---

# 1. Modeling Scope and Resolution

- **Time horizon:** One day ahead
- **Time steps:**

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

- **Market:** Single-node, competitive, uniform-price day-ahead spot market
- **Participants:** 5 producers (3 conventional, 1 solar, 1 wind)

The model is intended for **Monte Carlo simulation** and **scenario analysis**, not equilibrium solving.

---

# 2. State Variables

## Market State (hour $t$)

- $D_t$: realized market demand (MWh)
- $\pi_t$: market-clearing price (€/MWh)
- $x_{i,t}$: dispatched quantity of producer $i$ (MWh)

## Wind Producer State

- $A_{w,t}$: available wind capacity (MWh)
- $C_{w,t}$: realizable wind generation (MWh)
- $\omega_t$: information available at bidding time (demand distribution parameters, forecasts of wind, beliefs about competitors)

---

# 3. Decision Variables (Wind Producer)

For each hour $t$, chosen **before realization of uncertainty**:

- $q_{w,t} \geq 0$: bid quantity (MWh)

- $p_{w,t} \geq 0$: bid price (€/MWh)

Decision vector:

$$\mathbf{u}_w = \{(q_{w,t}, p_{w,t})\}_{t=1}^{24}$$

Constraint:

$$0 \leq q_{w,t} \leq \overline{Q}_{w,t}$$

---

# 4. Stochastic Variables (Refined)

## 4.1 Market Demand (Key Refinement)

Market demand is assumed to be **Normally distributed with constant parameters**:

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

To ensure physical feasibility:

$$D_t = \max(0, \tilde{D}_t), \quad \tilde{D}_t \sim \mathcal{N}(\mu_D, \sigma_D^2)$$

or equivalently, a truncated normal distribution:

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

**Assumptions:**

- Demand is **independent across hours**, or
- Weakly correlated (can be extended via multivariate normal if needed)

---

## 4.2 Renewable Availability

- Wind availability:

$$A_{w,t} \sim \mathcal{W}_t(\theta_W)$$

- Solar availability:

$$A_{s,t} \sim \mathcal{S}_t(\theta_S)$$

---

## 4.3 Competitor Bids

For each conventional producer $i \in \mathcal{C}$:

$$(q_{i,t}, p_{i,t}) \sim \mathcal{B}_{i,t}(\theta_i)$$

---

# 5. Constant Parameters

OpenAI

### Demand Parameters

- $\mu_D$: mean hourly demand (MWh)
- $\sigma_D^2$: variance of hourly demand (MWh$^2$)

### Technical Parameters

- $\overline{Q}_{w,t}$: installed wind capacity
- $c_i$: marginal cost of producer $i$

### Market Rules

- Uniform pricing
- Merit-order dispatch
- No network constraints

---

## 6. Transition Function: Market Clearing

For each hour $t$:

1. **Bid Aggregation**

$$\mathcal{B}_t = \{(q_{i,t}, p_{i,t})\}_{i \in \mathcal{I}}$$

2. **Merit-Order Sorting**

$$p_{(1),t} \leq p_{(2),t} \leq \cdots \leq p_{(5),t}$$

3. **Dispatch Rule**

$$\sum_i x_{i,t} = D_t, \quad 0 \leq x_{i,t} \leq q_{i,t}$$

4. **Market-Clearing Price**

$$\pi_t = \min \left\{ p_{i,t} \mid \sum_{j: p_{j,t} \leq p_{i,t}} q_{j,t} \geq D_t \right\}$$

5. **Wind Feasibility Constraint**

$$x_{w,t} \leq C_{w,t} = \min(A_{w,t}, \overline{Q}_{w,t})$$

(Optional imbalance penalties can be added if $x_{w,t} > C_{w,t}$).

---

## 7. Revenue Function

### Hourly Revenue

$$R_{w,t} = \pi_t \cdot x_{w,t}$$

### Daily Revenue

$$R_w = \sum_{t=1}^{24} R_{w,t}$$

---

## 8. Objective Function

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

$$\max_{\mathbf{u}_w} \mathbb{E}[R_w]$$

Expectation taken over:

$$\{D_t, A_{w,t}, A_{s,t}, \mathcal{B}_{\mathcal{C},t}\}_{t=1}^{24}$$

Given the Normal demand assumption, this expectation is typically estimated via **Monte Carlo sampling**.

---

## 9. Deep Uncertainty (Unchanged, but Clarified)

The demand distribution parameters themselves are **deeply uncertain**:

- $\mu_D$: long-term demand growth or electrification
- $\sigma_D^2$: volatility due to weather, EV charging, heat pumps

These parameters define **outer-loop scenarios**, while $D_t$ realizations define **inner-loop stochastic uncertainty**.

---

## 10. Interpretation

With normally distributed demand:

- Price formation becomes probabilistic and analytically tractable
- Expected revenue can be approximated conditional on bid rank
- Sensitivity of revenue to $q_{w,t}$ and $p_{w,t}$ can be systematically explored

---

If you want, I can next:

- Derive **closed-form acceptance probabilities** for the wind bid under Normal demand
- Show how this becomes a **stochastic program**
- Extend the demand model to **correlated hours (multivariate normal)**
- Add **risk-averse objectives (variance or CVaR)**

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

Below is a **refined and fully consistent simulation-model specification** that explicitly incorporates the **behavioral assumptions for all producers**, while preserving a clean separation between **operational (stochastic) uncertainty** and **deep (structural) uncertainty**.

---

# 1. Modeling Framework

- **Market:** Day-ahead, uniform-price, single-node electricity spot market
- **Horizon:** 24 hourly dispatch intervals
- **Purpose:** Estimate the expected revenue distribution of a wind-power producer under uncertainty
- **Method:** Monte Carlo simulation with outer-loop scenario analysis for deep uncertainty

---

# 2. Agents and Index Sets

- Producers:

$$i \in \mathcal{I} = \{w, s, c_1, c_2, c_3\}$$

  where:

  - $w$: wind producer (decision-maker)
  - $s$: solar producer
  - $c_k$: conventional producers, $k = 1, 2, 3$

- Time index:

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

---

# 3. State Variables

## Market State (hour $t$)

- $D_t$: realized market demand (MWh)
- $\pi_t$: market-clearing price (€/MWh)

- $x_{i,t}$: dispatched quantity of producer $i$ (MWh)

### Wind Producer State

- $W_t$: realized wind production (MWh)
- $\omega_t$: information set at bidding time

## 4. Decision Variables (Wind Producer)

Chosen day-ahead, before uncertainty is realized:

- $q_{w,t} \geq 0$: bid quantity (MWh)
- $p_{w,t} \geq 0$: bid price (€/MWh)

Constraint:

$$0 \leq q_{w,t} \leq \overline{Q}_w$$

## 5. Stochastic Variables (Operational Uncertainty)

### 5.1 Market Demand

Market demand follows a Normal distribution with constant parameters:

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

(truncated at zero if necessary)

### 5.2 Wind Production (Refined)

Actual wind generation is stochastic:

$$W_t \sim \mathcal{N}(\mu_W, \sigma_W^2)$$

Delivered wind energy:

$$x_{w,t} \leq \min(q_{w,t}, W_t)$$

## 6. Behavioral Models of Other Producers

### 6.1 Conventional Producers

#### Bid Quantities (Stable)

For each conventional producer $c_k$:

$$q_{c_k,t} = \overline{Q}_{c_k}, \quad \forall t$$

**Bid Prices (Stochastic with Deep Uncertainty)**

$$p_{c_k,t} \sim \mathcal{N}(\mu_{c_k}, \sigma_{c_k}^2)$$

where:

- $\mu_{c_k}, \sigma_{c_k}^2$ are **deeply uncertain parameters**, sampled in outer-loop scenarios

---

### 6.2 Solar Producer

**Bid Quantity (Deterministic Diurnal Profile)**

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

**Bid Price (Stochastic with Deep Uncertainty)**

$$p_{s,t} \sim \mathcal{N}(\mu_s, \sigma_s^2)$$

with $\mu_s, \sigma_s^2$ deeply uncertain.

---

## 7. Deep Uncertainty (Outer-Loop Parameters)

The following parameters define **structural uncertainty scenarios**:

### Demand

- $\mu_D, \sigma_D^2$

### Conventional Producers

- $\mu_{c_k}, \sigma_{c_k}^2$, for $k = 1, 2, 3$

### Solar Producer

- $\mu_s, \sigma_s^2$

### Wind Resource

- $\mu_W, \sigma_W^2$

Each scenario corresponds to one draw from a **deep uncertainty space**.

---

## 8. Transition Function: Market Clearing

For each hour $t$:

1. **Bid Collection**

$$\mathcal{B}_t = \{(q_{i,t}, p_{i,t})\}_{i \in \mathcal{I}}$$

2. **Merit-Order Sorting**

$$p_{(1),t} \leq p_{(2),t} \leq \cdots \leq p_{(5),t}$$

3. **Dispatch**

$$\sum_i x_{i,t} = D_t, \quad 0 \leq x_{i,t} \leq q_{i,t}$$

4. **Market-Clearing Price**

$$\pi_t = \min \left\{ p_{i,t} \mid \sum_{j: p_{j,t} \leq p_{i,t}} q_{j,t} \geq D_t \right\}$$

5. **Wind Feasibility**

$$x_{w,t} \leq \min(q_{w,t}, W_t)$$

---

# 9. Revenue Function

### Hourly Revenue

$$R_{w,t} = \pi_t \cdot x_{w,t}$$

### Daily Revenue

$$R_w = \sum_{t=1}^{24} R_{w,t}$$

(Optional: imbalance penalties if $q_{w,t} > W_t$)

---

# 10. Objective Function

The wind producer seeks to maximize expected daily revenue:

$$\max_{\mathbf{u}_w} \ \mathbb{E}[R_w]$$

Expectation taken over **operational uncertainty**, conditional on a given deep-uncertainty scenario.

---

# 11. Simulation Algorithm (Two-Level)

### Outer Loop (Deep Uncertainty)

1. Sample $\{\mu_D, \sigma_D^2, \mu_{c_k}, \sigma_{c_k}^2, \mu_s, \sigma_s^2, \mu_W, \sigma_W^2\}$

### Inner Loop (Operational Uncertainty)

1. For each hour $t$, sample:
   - $D_t$
   - $p_{c_k,t}$
   - $p_{s,t}$
   - $W_t$
2. Clear market
3. Compute revenue

---

## 12. Outputs of Interest

- Expected daily revenue $\mathbb{E}[R_w]$
- Revenue variance and downside risk
- Acceptance probability by hour
- Sensitivity to bid price and quantity
- Robustness across deep-uncertainty scenarios

---

## 13. Interpretation

This refined specification yields a **behaviorally grounded, stochastic simulation model** that:

- Captures diurnal solar patterns
- Reflects stable thermal capacity with uncertain pricing
- Separates wind bidding decisions from stochastic realization
- Explicitly treats demand and price distributions
- Supports robust investment analysis under deep uncertainty

---

If you want, I can next:

- Provide **pseudo-code**
- Derive **acceptance probability expressions**
- Convert this into a **two-stage stochastic optimization model**
- Extend it to **correlated demand and renewable output**

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

Below is a **comprehensive and self-contained simulation-model specification** that incorporates the **imbalance penalty for wind under-delivery**, together with all previously stated assumptions on demand and producer behavior. The structure is suitable for Monte-Carlo simulation and robust (deep-uncertainty) analysis.

---

# 1. Model Scope and Structure

- **Market:** Competitive, single-node, uniform-price day-ahead electricity market
- **Time horizon:** One operating day
- **Time resolution:**

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

- **Agents:**
    - Wind producer (decision-maker)
    - Solar producer
    - Three conventional producers

The model is a **two-level uncertainty framework**:

- **Inner loop:** stochastic operational uncertainty
- **Outer loop:** deep (structural) uncertainty in distribution parameters

---

# 2. Index Sets

- Producers:

$$\mathcal{I} = \{w, s, c_1, c_2, c_3\}$$

- Conventional producers:

$$\mathcal{C} = \{c_1, c_2, c_3\}$$

---

# 3. Decision Variables (Wind Producer)

Chosen day-ahead, before uncertainty is realized:

For each hour $t$:

- $q_{w,t} \geq 0$: bid quantity (MWh)
- $p_{w,t} \geq 0$: bid price (€/MWh)

Constraints:

$$0 \leq q_{w,t} \leq \overline{Q}_w$$

Decision vector:

$$\mathbf{u}_w = \{(q_{w,t}, p_{w,t})\}_{t=1}^{24}$$

---

## 4. State Variables

### Market State (per hour $t$)

- $D_t$: realized system demand (MWh)
- $\pi_t$: market-clearing price (€/MWh)
- $x_{i,t}$: dispatched quantity of producer $i$ (MWh)

### Wind Producer State

- $W_t$: realized wind production (MWh)
- $\delta_{w,t}$: wind delivery shortfall (MWh)

---

## 5. Stochastic Variables (Operational Uncertainty)

### 5.1 Market Demand

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

---

### 5.2 Wind Production

$$W_t \sim \mathcal{N}(\mu_W, \sigma_W^2)$$

(Truncated at zero if needed.)

---

### 5.3 Conventional Producers

- **Bid quantities (stable):**

$$q_{c_k,t} = \overline{Q}_{c_k}, \quad \forall t$$

- **Bid prices:**

$$p_{c_k,t} \sim \mathcal{N}(\mu_{c_k}, \sigma_{c_k}^2)$$

---

### 5.4 Solar Producer

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

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

- **Bid price:**

$$p_{s,t} \sim \mathcal{N}(\mu_s, \sigma_s^2)$$

---

## 6. Constant Parameters

### Market and Technology

- $\overline{Q}_w$: installed wind capacity
- $\overline{Q}_{c_k}$: capacity of conventional producer $c_k$
- $a, b$: solar profile parameters

### Penalty Mechanism

- $\lambda > 0$: imbalance penalty cost (€/MWh) for wind under-delivery

---

## 7. Deep Uncertainties (Outer-Loop Parameters)

These parameters are **not assigned a single probability distribution** and are explored via scenario analysis:

### Demand

- $\mu_D$: mean demand
- $\sigma_D^2$: demand variance

### Wind Resource

- $\mu_W$: mean wind production
- $\sigma_W^2$: wind production variance

### Conventional Producers

- $\mu_{c_k}, \sigma_{c_k}^2$, for $k = 1, 2, 3$

### Solar Producer

- $\mu_s, \sigma_s^2$

Each scenario corresponds to one realization of these parameters.

---

## 8. Transition Function: Market Clearing

For each hour $t$:

### 8.1 Bid Collection

$$\mathcal{B}_t = \{(q_{i,t}, p_{i,t})\}_{i \in \mathcal{I}}$$

---

### 8.2 Merit-Order Dispatch

Sort bids by increasing price and dispatch until demand is met:

$$\sum_{i \in \mathcal{I}} x_{i,t} = D_t, \quad 0 \leq x_{i,t} \leq q_{i,t}$$

---

### 8.3 Market-Clearing Price

$$\pi_t = \min \left\{ p_{i,t} \mid \sum_{j : p_{j,t} \leq p_{i,t}} q_{j,t} \geq D_t \right\}$$

---

### 8.4 Wind Dispatch

If the wind bid is accepted:

$$x_{w,t} \leq q_{w,t}$$

otherwise:

$$x_{w,t} = 0$$

---

## 9. Imbalance and Penalty Mechanism (Key Addition)

### 9.1 Wind Shortfall

If actual production is insufficient:

$$\delta_{w,t} = \max(0, x_{w,t} - W_t)$$

---

### 9.2 Penalty Cost

$$\text{Penalty}_{w,t} = \lambda \cdot \delta_{w,t}$$

# 10. Revenue and Objective Function

## 10.1 Hourly Wind Revenue

$$R_{w,t} = \pi_t \cdot x_{w,t} - \lambda \cdot \delta_{w,t}$$

## 10.2 Daily Revenue

$$R_w = \sum_{t=1}^{24} R_{w,t}$$

## 10.3 Objective Function

The wind producer chooses bids to maximize expected daily revenue:

$$\max_{\mathbf{u}_w} \ \mathbb{E}[R_w]$$

Expectation is taken over all **operational stochastic variables**, conditional on a given deep-uncertainty scenario.

# 11. Simulation Logic (Two-Level)

## Outer Loop (Deep Uncertainty)

1. Select a scenario by fixing:

$$(\mu_D, \sigma_D^2, \mu_W, \sigma_W^2, \mu_{c_k}, \sigma_{c_k}^2, \mu_s, \sigma_s^2)$$

## Inner Loop (Operational Uncertainty)

1. Sample $D_t, W_t, p_{c_k,t}, p_{s,t}$ for all $t$
2. Clear the market hour by hour
3. Compute dispatch, shortfalls, penalties, and revenue
4. Aggregate daily revenue

# 12. Key Model Outputs

- Expected daily wind revenue
- Revenue variance and downside risk
- Expected imbalance penalties

- Acceptance probability of wind bids
- Sensitivity of revenue to bid price, quantity, and penalty level
- Robustness across deep-uncertainty scenarios

---

## 13. Interpretation

This specification results in a **realistic stochastic revenue model** for wind power that:

- Penalizes over-commitment under uncertain production
- Captures price formation via merit-order clearing
- Reflects diurnal solar output and stable thermal capacity
- Separates short-term randomness from long-term deep uncertainty
- Is directly applicable to investment viability and robust bidding analysis

If you want, I can next:

- Reformulate this as a **two-stage stochastic optimization problem**
- Derive **closed-form expected penalty expressions**
- Add **risk-averse objectives (e.g., CVaR)**
- Provide **pseudo-code or numerical examples**

$\mu_D$, $\sigma_D$, $\mu_W$ and $\sigma_W$ are constant parameters rather than deeply uncertain parameters. Please refine your specification.

Below is a **refined and internally consistent specification** reflecting your clarification that $\mu_D, \sigma_D, \mu_W, \sigma_W$ are **fixed (known) parameters**, not deeply uncertain.
All other elements remain unchanged, with a clean separation between **stochastic uncertainty** and **deep uncertainty**.

---

## Comprehensive Simulation Model Specification

*(with imbalance penalties and refined uncertainty structure)*

---

## 1. Model Scope

- **Market:** Competitive, single-node, uniform-price day-ahead electricity market
- **Time horizon:** One day
- **Resolution:** 24 hourly dispatch intervals
- **Purpose:** Evaluate expected revenue of a wind-power producer under uncertainty and penalties

---

## 2. Index Sets

- Time:

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

- Producers:

$$\mathcal{I} = \{w, s, c_1, c_2, c_3\}$$

- Conventional producers:

$$\mathcal{C} = \{c_1, c_2, c_3\}$$

---

## 3. Decision Variables (Wind Producer)

Chosen day-ahead, before uncertainty is realized:

For each hour $t$:

- $q_{w,t} \geq 0$: bid quantity (MWh)
- $p_{w,t} \geq 0$: bid price (€/MWh)

Constraints:

$$0 \leq q_{w,t} \leq \overline{Q}_w$$

Decision vector:

$$\mathbf{u}_w = \{(q_{w,t}, p_{w,t})\}_{t \in \mathcal{T}}$$

---

# 4. State Variables

### Market State (hour $t$)

- $D_t$: realized system demand (MWh)
- $\pi_t$: market-clearing price (€/MWh)
- $x_{i,t}$: dispatched quantity of producer $i$ (MWh)

### Wind Producer State

- $W_t$: realized wind production (MWh)
- $\delta_{w,t}$: wind production shortfall (MWh)

---

# 5. Stochastic Variables (Operational Uncertainty)

### 5.1 Market Demand (Fixed Distribution)

Demand is stochastic but its distribution is known and stationary:

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

---

### 5.2 Wind Production (Fixed Distribution)

Actual wind generation is stochastic with fixed parameters:

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

---

### 5.3 Conventional Producers

- **Bid quantities (stable):**

$$q_{c_k,t} = \overline{Q}_{c_k}, \quad \forall t$$

- **Bid prices (stochastic):**

$$p_{c_k,t} \sim \mathcal{N}(\mu_{c_k}, \sigma_{c_k}^2)$$

---

### 5.4 Solar Producer

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

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

- **Bid price (stochastic):**

$$p_{s,t} \sim \mathcal{N}(\mu_s, \sigma_s^2)$$

---

# 6. Constant Parameters

## Fixed (Known) Parameters

- $\mu_D, \sigma_D$: mean and standard deviation of demand
- $\mu_W, \sigma_W$: mean and standard deviation of wind production
- $\overline{Q}_w$: installed wind capacity
- $\overline{Q}_{c_k}$: capacity of conventional producer $c_k$
- $a, b$: solar production profile parameters
- $\lambda > 0$: penalty cost for wind under-delivery (€/MWh)

---

# 7. Deep Uncertainties (Structural / Outer-Loop)

Deep uncertainty applies **only to competitors' bidding behavior**, not to demand or wind availability.

## Conventional Producers

- Mean bid prices: $\mu_{c_k}$
- Bid-price variances: $\sigma_{c_k}^2$

## Solar Producer

- Mean bid price: $\mu_s$
- Bid-price variance: $\sigma_s^2$

These parameters are **scenario variables** and are not assigned a single probability distribution.

---

# 8. Transition Function: Market Clearing

For each hour $t$:

## 8.1 Bid Collection

$$\mathcal{B}_t = \{(q_{i,t}, p_{i,t})\}_{i \in \mathcal{I}}$$

---

### 8.2 Merit-Order Dispatch

Sort bids in ascending order of price and dispatch until demand is met:

$$\sum_{i \in \mathcal{I}} x_{i,t} = D_t, \quad 0 \leq x_{i,t} \leq q_{i,t}$$

### 8.3 Market-Clearing Price

$$\pi_t = \min \left\{ p_{i,t} \mid \sum_{j : p_{j,t} \leq p_{i,t}} q_{j,t} \geq D_t \right\}$$

### 8.4 Wind Dispatch

If the wind bid is accepted:

$$x_{w,t} \leq q_{w,t}$$

otherwise:

$$x_{w,t} = 0$$

## 9. Imbalance and Penalty Mechanism

### 9.1 Wind Shortfall

If actual production is insufficient:

$$\delta_{w,t} = \max(0, x_{w,t} - W_t)$$

### 9.2 Penalty Cost

$$\text{Penalty}_{w,t} = \lambda \cdot \delta_{w,t}$$

## 10. Revenue and Objective Function

### Hourly Revenue

$$R_{w,t} = \pi_t \cdot x_{w,t} - \lambda \cdot \delta_{w,t}$$

### Daily Revenue

$$R_w = \sum_{t \in \mathcal{T}} R_{w,t}$$

### Objective Function

The wind producer chooses bids to maximize expected daily revenue:

$$\max_{\mathbf{u}_w} \ \mathbb{E}[R_w]$$

Expectation is taken over **operational stochastic variables only**:

$$\{D_t, W_t, p_{c_k,t}, p_{s,t}\}_{t\in\mathcal{T}}$$

conditional on a given deep-uncertainty scenario.

---

## 11. Simulation Structure

### Outer Loop (Deep Uncertainty)

1. Select a scenario:

$$\{\mu_{c_k}, \sigma^2_{c_k}, \mu_s, \sigma^2_s\}$$

### Inner Loop (Monte Carlo)

1. Sample $D_t, W_t, p_{c_k,t}, p_{s,t}$
2. Clear the market hour by hour
3. Compute dispatch, shortfall, penalties, and revenue
4. Aggregate daily revenue statistics

---

## 12. Interpretation

With this refinement:

- **Demand and wind availability risk are quantifiable and stationary**
- **Strategic price uncertainty from competitors remains deeply uncertain**
- Penalties create a clear trade-off between aggressive bidding and imbalance risk
- The model cleanly supports **expected value**, **risk analysis**, and **robust investment assessment**

If you want, I can now:

- Reformulate this as a **two-stage stochastic program**
- Derive **closed-form expressions for expected penalties**
- Introduce **risk-averse objectives (variance, CVaR)**
- Show how optimal bidding changes with the penalty level

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

Below is a **two-perspective refinement** of the model—one from the **wind-power producer** and one from the **system regulator**—built on a **shared market environment**.
Each stakeholder controls **its own decision variables**, interacts through the same market-clearing mechanism, and evaluates outcomes using **distinct objective functions** (listed separately, not weighted).

---

# I. Common Environment (Shared by All Stakeholders)

This section defines the **global state, parameters, stochastic processes, and transition rules** that are *not controlled* by either stakeholder.

---

## I.1 Time and Agents

- Time periods:

$$t \in \mathcal{T} = \{1, \ldots, 24\}$$

- Producers:

$$\mathcal{I} = \{w, s, c_1, c_2, c_3\}$$

where $w$ = wind, $s$ = solar, $c_k$ = conventional.

---

## I.2 Global State Variables

For each hour $t$:

- $D_t$: realized system demand (MWh)
- $\pi_t$: market-clearing price (€/MWh)
- $x_{i,t}$: dispatched quantity of producer $i$ (MWh)
- $W_t$: realized wind production (MWh)
- $\delta_{w,t}$: wind under-delivery (MWh)

These variables evolve according to stochastic processes and market rules.

---

## I.3 Global Stochastic Processes (Operational Uncertainty)

These are **exogenous** to all stakeholders.

### Demand

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

### Wind Production

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

### Competitor Bidding Prices

- Conventional producers:

$$p_{c_k,t} \sim \mathcal{N}(\mu_{c_k}, \sigma_{c_k}^2)$$

- Solar producer:

$$p_{s,t} \sim \mathcal{N}(\mu_s, \sigma_s^2)$$

---

## I.4 Global Deterministic Parameters

- Installed capacities:
  - $\overline{Q}_w$: wind
  - $\overline{Q}_{c_k}$: conventional
- Solar bid quantity profile:

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

- Stable conventional quantities:

$$q_{c_k,t} = \overline{Q}_{c_k}$$

---

## I.5 Deep Uncertainties (Scenario Parameters)

These are fixed within a scenario but vary across scenarios:

- $\mu_{c_k}, \sigma_{c_k}^2$: conventional bid-price distributions
- $\mu_s, \sigma_s^2$: solar bid-price distribution

*(Demand and wind parameters are **not** deeply uncertain.)*

---

## I.6 Market-Clearing Transition Function

For each hour $t$:

1. **Bid set**

$$\mathcal{B}_t = \{(q_{i,t}, p_{i,t})\}_{i \in \mathcal{I}}$$

2. **Merit-order dispatch**

$$\sum_i x_{i,t} = D_t, \quad 0 \le x_{i,t} \le q_{i,t}$$

3. **Market-clearing price**

$$\pi_t = \min \left\{ p_{i,t} : \sum_{j : p_{j,t} \le p_{i,t}} q_{j,t} \ge D_t \right\}$$

4. **Wind shortfall**

$$\delta_{w,t} = \max(0, x_{w,t} - W_t)$$

---

# II. Perspective 1: Wind-Power Producer

The wind producer is a **strategic profit-maximizing agent** operating within the common environment.

---

## II.1 Decision Variables (Controlled by Wind Producer)

Chosen day-ahead:

- $q_{w,t}$: bid quantity (MWh)
- $p_{w,t}$: bid price (€/MWh)

Constraints:

$$0 \le q_{w,t} \le \overline{Q}_w$$

---

## II.2 State Variables (Relevant to Wind Producer)

- Accepted dispatch $x_{w,t}$
- Realized production $W_t$
- Shortfall $\delta_{w,t}$
- Market price $\pi_t$

---

## II.3 Wind-Producer Transition Logic

Wind outcomes are determined by:

- Market-clearing function (shared)
- Physical feasibility:

$$x_{w,t} \leq q_{w,t}$$

---

## II.4 Objective Functions (Multiple, Non-Weighted)

### Objective 1: Maximize Expected Profit

$$\max \ \mathbb{E}\left[\sum_{t=1}^{24} (\pi_t x_{w,t} - \lambda \delta_{w,t})\right]$$

### Objective 2: Minimize Expected Imbalance Exposure

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

### Objective 3: Minimize Revenue Variability (Risk)

$$\min \ \mathrm{Var}\left(\sum_{t=1}^{24} \pi_t x_{w,t}\right)$$

These objectives reflect **profitability–risk trade-offs** but are **not aggregated**.

---

## III. Perspective 2: System Regulator

The regulator does **not bid** but shapes market outcomes via **policy instruments**.

---

## III.1 Decision Variables (Controlled by Regulator)

Examples (any subset may be active):

- $\lambda$: imbalance penalty (€/MWh)
- $\underline{p}_w$: minimum allowed wind bid price
- Market design choice (fixed within a run):
  - uniform pricing vs alternatives (if extended)

These decisions are made **before market operation**.

---

## III.2 State Variables (Observed by Regulator)

- Market prices $\pi_t$
- Dispatch quantities $x_{i,t}$
- Wind shortfalls $\delta_{w,t}$
- Total supplied energy and unmet demand (if any)

---

## III.3 Regulator Transition Logic

The regulator:

- Sets policy parameters (e.g. $\lambda$)
- Observes outcomes produced by the **same market-clearing mechanism**
- Does not interfere with dispatch ex post

---

## III.4 Objective Functions (Multiple, Non-Weighted)

### Objective 1: Minimize System Imbalance

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

### Objective 2: Minimize Price Volatility

$$\min \ \mathrm{Var}(\pi_t)$$

### Objective 3: Promote Renewable Utilization

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

### Objective 4: Maintain Market Efficiency

$$\min \ \mathbb{E}\left[\sum_{i,t} c_i x_{i,t}\right]$$

(where $c_i$ are marginal costs, if modeled)

---

## IV. Relationship Between Perspectives

- Both stakeholders:
  - Share the **same environment**

- Observe the **same stochastic realizations**
- Are coupled only through **market-clearing outcomes**
- Control variables are **independent**
- Objectives may be **conflicting** (e.g., aggressive wind bidding vs system stability)

---

# V. Interpretation

This formulation yields a **multi-actor, stochastic simulation framework** where:

- The **wind producer** optimizes bidding under price and production risk
- The **regulator** tunes policy levers to manage system reliability and efficiency
- Trade-offs emerge endogenously without forcing a single objective

If you want, I can next:

- Formulate this as a **bi-level or game-theoretic model**
- Add **learning or adaptive regulation**
- Extend to **multi-day dynamics**
- Provide **numerical examples or pseudo-code**

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

Below is a **single, unified, internally consistent model** that **composes the wind-producer and regulator perspectives into one shared environment**, while **preserving modularity**, **separate objectives**, and **exogenous treatment of other stakeholders' decisions**.
The result is a **multi-perspective stochastic simulation model** rather than a single optimization problem.

---

# Unified Composed Model for the Day-Ahead Electricity Market

---

## 1. Model Architecture (High-Level)

The model is composed of:

1. **A common environment**
   – shared states, stochastic processes, parameters, and transition rules
2. **Perspective modules**
   – each module controls its own decisions and evaluates its own objectives
3. **Composition rule**
   – decisions from each perspective enter the environment as **exogenous inputs** to other perspectives

Formally, the system is a **stochastic state–transition model** with multiple decision-making agents.

---

## 2. Common Environment (Shared by All Perspectives)

### 2.1 Time and Index Sets

- Time:

$$t \in \mathcal{T} = \{1, \ldots, 24\}$$

- Producers:

$$\mathcal{I} = \{w, s, c_1, c_2, c_3\}$$

- Conventional producers:

$$\mathcal{C} = \{c_1, c_2, c_3\}$$

---

## 2.2 Global State Variables

For each hour $t$:

- $D_t$: realized electricity demand (MWh)
- $\pi_t$: market-clearing price (€/MWh)
- $x_{i,t}$: dispatched quantity of producer $i$ (MWh)
- $W_t$: realized wind production (MWh)
- $\delta_{w,t}$: wind under-delivery (MWh)

State vector:

$$S_t = (D_t, \pi_t, \{x_{i,t}\}_{i \in \mathcal{I}}, W_t, \delta_{w,t})$$

---

## 2.3 Global Stochastic Variables (Operational Uncertainty)

These are **exogenous to all decision-makers**.

- Demand:

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

- Wind production:

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

- Bid prices of other producers:

$$p_{c_k,t} \sim \mathcal{N}(\mu_{c_k}, \sigma_{c_k}^2), \quad p_{s,t} \sim \mathcal{N}(\mu_s, \sigma_s^2)$$

---

## 2.4 Global Deterministic Parameters

- Installed capacities:
  - $\overline{Q}_w$: wind
  - $\overline{Q}_{c_k}$: conventional
- Solar bid quantity:

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

- Conventional bid quantities:

$$q_{c_k,t} = \overline{Q}_{c_k}$$

- Imbalance penalty parameter:

$$\lambda > 0 \quad (€/\mathrm{MWh})$$

## 2.5 Deep Uncertainty (Scenario Parameters)

These are fixed within a scenario and shared across perspectives:

- $\mu_{c_k}, \sigma^2_{c_k}$: conventional bid-price distributions
- $\mu_s, \sigma^2_s$: solar bid-price distribution

## 2.6 Environment Transition Function (Market Clearing)

For each hour $t$, given bids $(q_{i,t}, p_{i,t})$:

1. **Merit-order dispatch**

$$\sum_{i \in \mathcal{I}} x_{i,t} = D_t, \quad 0 \le x_{i,t} \le q_{i,t}$$

2. **Market-clearing price**

$$\pi_t = \min \left\{ p_{i,t} \mid \sum_{j : p_{j,t} \le p_{i,t}} q_{j,t} \ge D_t \right\}$$

3. **Wind shortfall**

$$\delta_{w,t} = \max(0, x_{w,t} - W_t)$$

This transition function is **identical for all perspectives**.

# 3. Perspective Module A: Wind-Power Producer

## 3.1 Decision Variables (Controlled by Wind Producer)

Chosen day-ahead:

- $q_{w,t}$: wind bid quantity (MWh)
- $p_{w,t}$: wind bid price (€/MWh)

Constraints:

$$0 \le q_{w,t} \le \overline{Q}_w$$

## 3.2 Exogenous Inputs (from Environment & Regulator)

- Penalty level $\lambda$
- Demand realizations $D_t$

- Competitors' bids $p_{c_k,t}, p_{s,t}$
- Market-clearing outcomes $(\pi_t, x_{w,t})$

### 3.3 Wind-Producer Objectives (Evaluated Separately)

**Objective W1: Maximize expected profit**

$$\max \ \mathbb{E}\left[\sum_{t=1}^{24} (\pi_t x_{w,t} - \lambda \delta_{w,t})\right]$$

**Objective W2: Minimize expected imbalance**

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

**Objective W3: Minimize revenue volatility**

$$\min \ \mathrm{Var}\left(\sum_{t=1}^{24} \pi_t x_{w,t}\right)$$

The wind producer treats **regulatory parameters as fixed inputs**.

## 4. Perspective Module B: System Regulator

### 4.1 Decision Variables (Controlled by Regulator)

Chosen prior to market operation:

- $\lambda$: imbalance penalty level (€/MWh)
  *(Additional policy levers could be added without altering structure.)*

### 4.2 Exogenous Inputs (from Environment & Wind Producer)

- Wind bids $(q_{w,t}, p_{w,t})$
- Stochastic realizations $(D_t, W_t)$
- Market outcomes $(\pi_t, x_{i,t}, \delta_{w,t})$

### 4.3 Regulator Objectives (Evaluated Separately)

**Objective R1: Minimize system imbalance**

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

**Objective R2: Minimize price volatility**

$$\min \ \mathrm{Var}(\pi_t)$$

**Objective R3: Maximize renewable utilization**

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

**Objective R4: Preserve market efficiency**

$$\min \ \mathbb{E}\left[\sum_{i,t} c_i x_{i,t}\right] \quad \text{(if marginal costs are modeled)}$$

The regulator treats **wind bidding behavior as exogenous**.

---

## 5. Composition Rule (Key Consistency Feature)

- All perspectives:
  - Share the **same state variables**
  - Use the **same stochastic realizations**
  - Interact only via the **market-clearing transition**
- Each perspective:
  - Controls **only its own decisions**
  - Treats all other decisions as **exogenous inputs**
  - Evaluates **its own objective functions independently**

There is **no forced equilibrium or weighted objective**.

---

## 6. Modular Evaluation Enabled by the Unified Model

This composition allows:

- Wind-only analysis: fix $\lambda$, vary wind bids
- Regulator-only analysis: fix wind strategy, vary $\lambda$
- Stress testing: same environment, different objectives
- Robust investment assessment without conflating incentives

---

## 7. Interpretation

The unified model is:

- **Internally consistent** (single environment and state evolution)
- **Modular** (perspective-specific decisions and objectives)

- **Extensible** (additional stakeholders or policies can be added)
- **Policy-relevant** (reveals trade-offs rather than assuming alignment)

If you want, the next natural steps are:

- Formulating this as a **bi-level optimization or game**
- Adding **adaptive or learning agents**
- Extending to **multi-day dynamics**
- Implementing a **numerical simulation blueprint**

Now, provide a modular Python implementation of the unified model, using classes to represent different perspectives. At the end of the program, define a function as the interface. This function should take as inputs a dictionary of decision variable values, a dictionary of uncertain parameter values, and a dictionary of constant parameter values; simulate the model dynamics; and return the value of each objective function. This function does not need to be executed.

Below is a **modular, class-based Python implementation** of the **unified composed model**.
It reflects the shared environment, embeds **perspective-specific decisions and objectives**, treats other perspectives' decisions as **exogenous inputs**, and exposes a **single interface function** at the end.

The code is **structural and illustrative** (not optimized, not executed), exactly as requested.

---

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


# ============================================================
# 1. Common Environment
# ============================================================

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

    def __init__(self, constants: Dict[str, Any], uncertainties: Dict[str, Any]):
        # Fixed parameters
        self.mu_D = constants["mu_D"]
        self.sigma_D = constants["sigma_D"]
        self.mu_W = constants["mu_W"]
        self.sigma_W = constants["sigma_W"]

        self.Q_w = constants["Q_w"]
        self.Q_c = constants["Q_c"]  # dict: producer -> capacity
        self.a = constants["solar_a"]
        self.b = constants["solar_b"]
        self.lambda_penalty = constants["lambda_penalty"]

        # Deep uncertainty parameters
        self.mu_c = uncertainties["mu_c"]          # dict
        self.sigma_c = uncertainties["sigma_c"]    # dict
        self.mu_s = uncertainties["mu_s"]
        self.sigma_s = uncertainties["sigma_s"]

        self.T = 24

    def solar_quantity(self, t: int) -> float:
        return max(0.0, self.a + self.b * np.cos(2 * np.pi * t / 24))

    def sample_demand(self) -> float:
        return max(0.0, np.random.normal(self.mu_D, self.sigma_D))

    def sample_wind(self) -> float:
        return max(0.0, np.random.normal(self.mu_W, self.sigma_W))

    def sample_prices(self) -> Dict[str, float]:
        prices = {}
        for c in self.mu_c:
            prices[c] = np.random.normal(self.mu_c[c], self.sigma_c[c])
```

```python
        prices["solar"] = np.random.normal(self.mu_s, self.sigma_s)
        return prices

    def clear_market(
        self,
        bids: Dict[str, Dict[str, float]],
        demand: float,
        wind_realization: float
    ) -> Dict[str, Any]:
        """
        Merit-order market clearing.
        """
        # Sort bids by price
        sorted_bids = sorted(bids.items(), key=lambda x: x[1]["price"])

        dispatch = {i: 0.0 for i in bids}
        cumulative = 0.0
        clearing_price = 0.0

        for producer, bid in sorted_bids:
            available = bid["quantity"]
            if cumulative + available >= demand:
                dispatch[producer] = demand - cumulative
                clearing_price = bid["price"]
                break
            else:
                dispatch[producer] = available
                cumulative += available

        # Wind shortfall
        wind_dispatch = dispatch.get("wind", 0.0)
        shortfall = max(0.0, wind_dispatch - wind_realization)

        return {
            "dispatch": dispatch,
            "price": clearing_price,
            "wind_shortfall": shortfall
        }


# ============================================================
# 2. Wind Producer Perspective
# ============================================================

class WindProducer:
    """
    Wind producer decision logic and objectives.
    """

    def __init__(self, decisions: Dict[str, Any]):
        self.quantities = decisions["q_w"]  # dict: t -> quantity
        self.prices = decisions["p_w"]      # dict: t -> price

    def bid(self, t: int) -> Dict[str, float]:
        return {
            "quantity": self.quantities[t],
            "price": self.prices[t]
        }

    @staticmethod
    def objectives(history: List[Dict[str, Any]], lambda_penalty: float) -> Dict[str, float]:
        revenues = []
        shortfalls = []

        for h in history:
            revenue = h["price"] * h["dispatch"]["wind"] - lambda_penalty * h["wind_shortfall"]
            revenues.append(revenue)
            shortfalls.append(h["wind_shortfall"])

        return {
            "W1_expected_profit": float(np.mean(revenues)),
            "W2_expected_imbalance": float(np.mean(shortfalls)),
            "W3_revenue_variance": float(np.var(revenues))
        }
```

```python
# ============================================================
# 3. System Regulator Perspective
# ============================================================

class SystemRegulator:
    """
    Regulator objectives evaluated from system outcomes.
    """

    @staticmethod
    def objectives(history: List[Dict[str, Any]]) -> Dict[str, float]:
        prices = [h["price"] for h in history]
        wind_dispatch = [h["dispatch"]["wind"] for h in history]
        shortfalls = [h["wind_shortfall"] for h in history]

        return {
            "R1_expected_imbalance": float(np.mean(shortfalls)),
            "R2_price_variance": float(np.var(prices)),
            "R3_expected_wind_dispatch": float(np.mean(wind_dispatch))
        }


# ============================================================
# 4. Unified Simulation Engine
# ============================================================

class UnifiedMarketModel:
    """
    Composed model embedding all perspectives.
    """

    def __init__(
        self,
        environment: MarketEnvironment,
        wind_producer: WindProducer
    ):
        self.env = environment
        self.wind = wind_producer

    def run(self) -> List[Dict[str, Any]]:
        history = []

        for t in range(1, self.env.T + 1):
            demand = self.env.sample_demand()
            wind_realization = self.env.sample_wind()
            prices = self.env.sample_prices()

            bids = {
                "wind": self.wind.bid(t),
                "solar": {
                    "quantity": self.env.solar_quantity(t),
                    "price": prices["solar"]
                }
            }

            for c in self.env.Q_c:
                bids[c] = {
                    "quantity": self.env.Q_c[c],
                    "price": prices[c]
                }

            result = self.env.clear_market(
                bids=bids,
                demand=demand,
                wind_realization=wind_realization
            )

            history.append(result)

        return history


# ============================================================
# 5. Public Interface Function
```

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

def simulate_unified_model(
    decisions: Dict[str, Any],
    uncertainties: Dict[str, Any],
    constants: Dict[str, Any]
) -> Dict[str, Dict[str, float]]:
    """
    Interface function.

    Inputs:
        decisions     : decision variable values (wind bids)
        uncertainties : uncertain / deep-uncertainty parameters
        constants     : fixed model parameters

    Output:
        Dictionary mapping stakeholder -> objective values
    """

    env = MarketEnvironment(constants, uncertainties)
    wind = WindProducer(decisions)
    model = UnifiedMarketModel(env, wind)

    history = model.run()

    wind_objectives = WindProducer.objectives(history, env.lambda_penalty)
    regulator_objectives = SystemRegulator.objectives(history)

    return {
        "wind_producer": wind_objectives,
        "system_regulator": regulator_objectives
    }
```

---

## Notes on Design Consistency

- **Single shared environment** (`MarketEnvironment`)
- **Perspective isolation** via:
  - `WindProducer` decisions
  - `SystemRegulator` objectives only
- **Exogenous treatment**:
  - Wind treats penalty as fixed
  - Regulator treats bids as fixed
- **Modularity**:
  - Perspectives can be evaluated independently
  - New stakeholders can be added without changing the environment
- **No execution required** (as requested)

---

If you want next steps, I can:

- Add **Monte Carlo batching**
- Add **risk measures (CVaR)**
- Convert this into a **bi-level optimization shell**
- Provide a **unit-test scaffold**

Please refine your Python implementation to model "lambda_penalty" as a decision variable controlled by the system regulator.

Below is a **refined, internally consistent Python implementation** in which **`lambda_penalty` is a decision variable controlled by the system regulator**.
Key changes vs. the previous version:

- **`lambda_penalty`** is **removed from constants** and **provided by the regulator**.
- The **environment reads `lambda_penalty` at run time** (exogenous to the wind producer).
- The **public interface** accepts **both wind decisions and regulator decisions** in the **`decisions`** dictionary.
- Objectives remain **separate and modular**.

The code is **structural** and **does not need to be executed**.

---

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


# ============================================================
# 1. Common Environment
# ============================================================

class MarketEnvironment:
    """
    Shared stochastic environment and market-clearing mechanism.
    The imbalance penalty (lambda_penalty) is provided exogenously
    by the system regulator at run time.
    """

    def __init__(self, constants: Dict[str, Any], uncertainties: Dict[str, Any]):
        # Fixed (known) parameters
        self.mu_D = constants["mu_D"]
        self.sigma_D = constants["sigma_D"]
        self.mu_W = constants["mu_W"]
        self.sigma_W = constants["sigma_W"]

        self.Q_w = constants["Q_w"]
        self.Q_c = constants["Q_c"]          # dict: conventional producer -> capacity
        self.a = constants["solar_a"]
        self.b = constants["solar_b"]

        # Deep-uncertainty parameters (scenario inputs)
        self.mu_c = uncertainties["mu_c"]    # dict
        self.sigma_c = uncertainties["sigma_c"]
        self.mu_s = uncertainties["mu_s"]
        self.sigma_s = uncertainties["sigma_s"]

        self.T = 24

    # ----------------------------
    # Exogenous stochastic processes
    # ----------------------------

    def solar_quantity(self, t: int) -> float:
```

```python
        return max(0.0, self.a + self.b * np.cos(2 * np.pi * t / 24))

    def sample_demand(self) -> float:
        return max(0.0, np.random.normal(self.mu_D, self.sigma_D))

    def sample_wind(self) -> float:
        return max(0.0, np.random.normal(self.mu_W, self.sigma_W))

    def sample_prices(self) -> Dict[str, float]:
        prices = {}
        for c in self.mu_c:
            prices[c] = np.random.normal(self.mu_c[c], self.sigma_c[c])
        prices["solar"] = np.random.normal(self.mu_s, self.sigma_s)
        return prices

    # ----------------------------
    # Market-clearing transition
    # ----------------------------

    def clear_market(
        self,
        bids: Dict[str, Dict[str, float]],
        demand: float,
        wind_realization: float,
        lambda_penalty: float
    ) -> Dict[str, Any]:
        """
        Merit-order market clearing with imbalance penalty.
        """
        # Sort bids by price
        sorted_bids = sorted(bids.items(), key=lambda x: x[1]["price"])

        dispatch = {i: 0.0 for i in bids}
        cumulative = 0.0
        clearing_price = 0.0

        for producer, bid in sorted_bids:
            available = bid["quantity"]
            if cumulative + available >= demand:
                dispatch[producer] = demand - cumulative
                clearing_price = bid["price"]
                break
            else:
                dispatch[producer] = available
                cumulative += available

        # Wind shortfall and penalty
        wind_dispatch = dispatch.get("wind", 0.0)
        shortfall = max(0.0, wind_dispatch - wind_realization)
        penalty = lambda_penalty * shortfall

        return {
            "dispatch": dispatch,
            "price": clearing_price,
            "wind_shortfall": shortfall,
            "wind_penalty": penalty
        }


# ============================================================
# 2. Wind Producer Perspective
# ============================================================

class WindProducer:
    """
    Wind producer: controls bidding decisions and evaluates its objectives.
    """

    def __init__(self, decisions: Dict[str, Any]):
        # decisions["wind"] contains wind-specific decisions
        self.quantities = decisions["wind"]["q_w"]  # dict: t -> quantity
        self.prices = decisions["wind"]["p_w"]      # dict: t -> price

    def bid(self, t: int) -> Dict[str, float]:
        return {
```

```
            "quantity": self.quantities[t],
            "price": self.prices[t]
        }

    @staticmethod
    def objectives(history: List[Dict[str, Any]]) -> Dict[str, float]:
        revenues = []
        shortfalls = []

        for h in history:
            revenue = (
                h["price"] * h["dispatch"]["wind"]
                - h["wind_penalty"]
            )
            revenues.append(revenue)
            shortfalls.append(h["wind_shortfall"])

        return {
            # Objective W1
            "W1_expected_profit": float(np.mean(revenues)),
            # Objective W2
            "W2_expected_imbalance": float(np.mean(shortfalls)),
            # Objective W3
            "W3_revenue_variance": float(np.var(revenues))
        }


# ============================================================
# 3. System Regulator Perspective
# ============================================================

class SystemRegulator:
    """
    System regulator: controls the imbalance penalty and evaluates
    system-level objectives.
    """

    def __init__(self, decisions: Dict[str, Any]):
        # decisions["regulator"] contains regulator-specific decisions
        self.lambda_penalty = decisions["regulator"]["lambda_penalty"]

    @staticmethod
    def objectives(history: List[Dict[str, Any]]) -> Dict[str, float]:
        prices = [h["price"] for h in history]
        wind_dispatch = [h["dispatch"]["wind"] for h in history]
        shortfalls = [h["wind_shortfall"] for h in history]

        return {
            # Objective R1
            "R1_expected_imbalance": float(np.mean(shortfalls)),
            # Objective R2
            "R2_price_variance": float(np.var(prices)),
            # Objective R3
            "R3_expected_wind_dispatch": float(np.mean(wind_dispatch))
        }


# ============================================================
# 4. Unified Composed Model
# ============================================================

class UnifiedMarketModel:
    """
    Unified model embedding all perspectives in a shared environment.
    """

    def __init__(
        self,
        environment: MarketEnvironment,
        wind_producer: WindProducer,
        regulator: SystemRegulator
    ):
        self.env = environment
        self.wind = wind_producer
        self.regulator = regulator
```

```python
    def run(self) -> List[Dict[str, Any]]:
        history = []

        for t in range(1, self.env.T + 1):
            demand = self.env.sample_demand()
            wind_realization = self.env.sample_wind()
            prices = self.env.sample_prices()

            bids = {
                "wind": self.wind.bid(t),
                "solar": {
                    "quantity": self.env.solar_quantity(t),
                    "price": prices["solar"]
                }
            }

            for c in self.env.Q_c:
                bids[c] = {
                    "quantity": self.env.Q_c[c],
                    "price": prices[c]
                }

            result = self.env.clear_market(
                bids=bids,
                demand=demand,
                wind_realization=wind_realization,
                lambda_penalty=self.regulator.lambda_penalty
            )

            history.append(result)

        return history


# ============================================================
# 5. Public Interface Function
# ============================================================

def simulate_unified_model(
    decisions: Dict[str, Any],
    uncertainties: Dict[str, Any],
    constants: Dict[str, Any]
) -> Dict[str, Dict[str, float]]:
    """
    Unified simulation interface.

    Inputs
    ------
    decisions:
        {
          "wind": {
              "q_w": {t: quantity},
              "p_w": {t: price}
          },
          "regulator": {
              "lambda_penalty": float
          }
        }

    uncertainties:
        {
          "mu_c": {producer: mean},
          "sigma_c": {producer: std},
          "mu_s": float,
          "sigma_s": float
        }

    constants:
        {
          "mu_D": float,
          "sigma_D": float,
          "mu_W": float,
          "sigma_W": float,
          "Q_w": float,
```

```
            "Q_c": {producer: capacity},
            "solar_a": float,
            "solar_b": float
        }

    Returns
    -------
    Dictionary mapping perspective -> objective values
    """

    env = MarketEnvironment(constants, uncertainties)
    wind = WindProducer(decisions)
    regulator = SystemRegulator(decisions)

    model = UnifiedMarketModel(env, wind, regulator)
    history = model.run()

    return {
        "wind_producer": WindProducer.objectives(history),
        "system_regulator": SystemRegulator.objectives(history)
    }
```

---

# What This Refinement Achieves

- **Regulator control**: **`lambda_penalty`** is now explicitly a **regulatory decision variable**.
- **Exogeneity respected**:
  - Wind producer treats **`lambda_penalty`** as fixed.
  - Regulator treats wind bids as fixed.
- **Single shared environment** with a consistent transition function.
- **Modular objectives**: no weighted aggregation, no forced equilibrium.

If you want, the next extensions could include:

- Searching over **`lambda_penalty`** to build a **regulatory Pareto frontier**
- Turning the wind problem into a **two-stage stochastic program**
- Adding **adaptive regulation** across multiple days

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

Great catch — this is an important market-clearing detail, and refining it actually **changes the dispatch logic in a fundamental way**.

Below is a **precise refinement** of the Python implementation so that:

> **Any bid with price ≤ clearing price is fully accepted, even if total demand has already been met.**

This implies:

- The market may **over-supply** relative to demand.
- The clearing price is still defined by the marginal bid that first meets demand.
- Dispatch is **price-based**, not demand-constrained for cheaper bids.

I will:

1. Explain the logic change clearly
2. Provide the **minimal but complete code refinement**, focusing on the market-clearing function
3. Show how consistency with objectives is preserved

---

# 1. Refined Market-Clearing Logic (Conceptual)

## Old logic (demand-constrained dispatch)

- Stop dispatch once demand is met
- Partial acceptance of marginal bid
- Later bids rejected

## New logic (price-based full acceptance)

1. Determine the **market-clearing price** as before:

$$\pi_t = \min \left\{ p_i : \sum_{j:p_j \leq p_i} q_j \geq D_t \right\}$$

2. Dispatch rule:

$$x_{i,t} = \begin{cases} q_{i,t} & \text{if } p_{i,t} \leq \pi_t \\ 0 & \text{otherwise} \end{cases}$$

3. Oversupply is allowed:

$$\sum_i x_{i,t} \geq D_t$$

This is common in simplified spot-market simulations and ensures **internal consistency with your specification**.

---

## 2. Refined Python Implementation (Targeted Change)

Only the **`clear_market` method** in **`MarketEnvironment`** needs to be rewritten.
Everything else remains valid and unchanged.

### ✅ Refined `clear_market` Method

```python
def clear_market(
    self,
    bids: Dict[str, Dict[str, float]],
    demand: float,
    wind_realization: float,
    lambda_penalty: float
) -> Dict[str, Any]:
    """
    Merit-order market clearing where any bid with price less than or
    equal to the clearing price is fully accepted, even if demand is exceeded.
    """

    # ------------------------------------------------
    # Step 1: Determine clearing price
    # ------------------------------------------------
    sorted_bids = sorted(bids.items(), key=lambda x: x[1]["price"])

    cumulative = 0.0
    clearing_price = None

    for producer, bid in sorted_bids:
        cumulative += bid["quantity"]
        if cumulative >= demand:
            clearing_price = bid["price"]
            break

    # Safety fallback (should not occur if capacity is sufficient)
    if clearing_price is None:
        clearing_price = max(bid["price"] for bid in bids.values())

    # ------------------------------------------------
    # Step 2: Full acceptance of all bids ≤ clearing price
    # ------------------------------------------------
    dispatch = {}
    for producer, bid in bids.items():
        if bid["price"] <= clearing_price:
            dispatch[producer] = bid["quantity"]
        else:
            dispatch[producer] = 0.0

    # ------------------------------------------------
    # Step 3: Wind shortfall and penalty
    # ------------------------------------------------
    wind_dispatch = dispatch.get("wind", 0.0)
    shortfall = max(0.0, wind_dispatch - wind_realization)
    penalty = lambda_penalty * shortfall

    return {
```

```
        "dispatch": dispatch,
        "price": clearing_price,
        "wind_shortfall": shortfall,
        "wind_penalty": penalty
    }
```

---

## 3. Why This Is Internally Consistent

### ✔️ Matches your specification

- **All bids ≤ clearing price are fully accepted**
- Clearing price is still demand-based
- No partial dispatch

### ✔️ Preserves modularity

- Wind producer still treats market rules as exogenous
- Regulator still controls only `lambda_penalty`
- Objectives remain unchanged

### ✔️ Compatible with oversupply

- Oversupply does **not** affect revenue calculation
- Price formation remains well-defined
- Penalty logic remains valid

---

## 4. Interpretation Implications (Important!)

This refinement has **real economic consequences**:

- Wind producers face **stronger incentives to shade quantities**, because over-acceptance increases imbalance exposure.
- The regulator's penalty parameter `lambda_penalty` becomes **more powerful** as a control lever.
- Market outcomes are **less demand-constrained** and more **price-driven**, which is often desirable in stylized simulations.

---

If you want, next we can:

- Add an **explicit oversupply metric** for the regulator
- Compare outcomes under **partial vs full acceptance**
- Introduce **curtailment or spill costs**
- Turn this into a **formal bilevel optimization problem**

Just tell me where you want to take it next 🚀