Green Grid Simulation

Emiliano Hinojosa Guzmán
Diego Amín Pallares Hernandez
José Salcedo Uribe

February 17, 2026

1 DESIGN CHOICES, CHALLENGES, AND FUTURE IMPROVEMENTS

Design choices. The simulation was designed as a modular digital twin where each physical subsystem is represented by an independent component (`panel`, `battery`, `home`, `weather`, `inverter`, and `grid`). A central SimPy process (`Simulate`) advances the system in hourly ticks and updates all components in a fixed order, while the inverter acts as the energy orchestration layer. This structure allowed us to compare strategies (`LOAD_PRIORITY`, `CHARGE_PRIORITY`, `PRODUCE_PRIORITY`) without rewriting the rest of the model. We also centralized all tunable parameters in `config.py` to make experiments reproducible and easy to modify.
Main challenges faced during implementation.

- **Race conditions and state freshness across components.** One of the hardest issues was guaranteeing that each module used the latest available information at every tick (for example, weather → panel generation → inverter dispatch → battery level). We addressed this by keeping a strict update sequence inside the simulation loop and logging state variables every step to validate consistency.
- **Keeping the architecture clean and readable.** As strategy logic and failure handling grew, the risk of mixing responsibilities increased. We mitigated this by preserving clear component boundaries, keeping shared constants in `config.py`, and using explicit metric dictionaries for reporting.
- **Numerical thresholds in `inverter.update()`.** SimPy and floating-point operations occasionally treated very small values as effectively zero, which caused unstable behav-

ior in branch logic. To stabilize decisions, we introduced small tolerances (for example checks such as `> 0.001`) before charging, discharging, importing, or exporting.

Potential improvements for future iterations.

- Refactor `inverter.update()` into smaller helper methods per strategy to reduce complexity and simplify testing.
- Add automated unit tests and scenario regression tests (including edge cases for near-zero energy flows and inverter failures).
- Introduce stochastic household demand profiles instead of constant base load to better represent real consumption behavior.
- Run Monte Carlo experiments across random seeds and seasons, then report confidence intervals rather than single-run values.
- Improve battery realism by explicitly enforcing a minimum SoC floor in dispatch logic and modeling degradation over time.

2 ANSWERS TO THE EXPECTED OUTPUT QUESTIONS

The following results were obtained from a 30-day simulation using the current configuration in `simulation/config.py`, with import cost = 0.75 and export credit = 0.90 per kWh.

1. **Average battery state of charge over the month.** The average SoC is **71.22%** (default run with `LOAD_PRIORITY`).
2. **How often the battery reaches full or empty state.** The battery is at full charge for **143 hours** and at empty state for **12 hours**.
3. **Total solar energy generated over the month.** Total solar generation is **610.92 kWh**.
4. **Total household energy consumed over the month.** Total household consumption is **340.00 kWh**.
5. **Energy imported from / exported to the grid.** The system imports **25.50 kWh** and exports **254.76 kWh**.
6. **Inverter failures and total downtime.** The inverter fails **10 times** with a total downtime of **40 hours**.
7. **Average cloud coverage during the month.** Average cloud coverage is **0.4266** (approximately **42.66%**).
8. **Peak load demand observed.** Peak load demand is **500 W**.
9. **Frequency of unmet load events.** Unmet load occurs **0 times** in the default monthly run.

10. **Battery system efficiency (round-trip losses).** The configured round-trip battery efficiency is **95%**, implying roughly **5%** round-trip energy loss.

11. **Effect of energy management strategy on performance.**
    - `LOAD_PRIORITY`: Balanced behavior, net balance **+210.16**, no unmet load.
    - `CHARGE_PRIORITY`: Very high SoC (**98.65%**) but heavy imports (**211.90 kWh**), net balance **-158.92**.
    - `PRODUCE_PRIORITY`: Highest exports (**460.32 kWh**) and highest net balance, but battery stays near empty and unmet load increases (**19** events).

12. **Most cost-effective strategy at given tariffs.** With export credit 0.90 and import cost 0.75 per kWh, `PRODUCE_PRIORITY` is most cost-effective, with net balance **+257.24**.

13. **Impact of cloud coverage on solar generation and battery usage.** Increasing cloud coverage reduces generation and export, and lowers SoC:
    - Clear (0.0–0.2): ~**1402.68 Wh/h** solar, SoC ~**77.96%**.
    - Partly cloudy (0.2–0.6): ~**950.41 Wh/h** solar, SoC ~**72.14%**.
    - Mostly cloudy (0.6–0.8): ~**519.26 Wh/h** solar, SoC ~**73.45%**.
    - Overcast (0.8–1.0): ~**176.94 Wh/h** solar, SoC ~**44.80%**, and near-zero export.

14. **Seasonal performance (summer vs winter, etc.).**
    - Summer performs best: **779.77 kWh** solar, average cloud **0.259**, net balance **+358.44**.
    - Spring and fall are intermediate: net balances **+210.16** and **+172.04**.
    - Winter performs worst: **365.91 kWh** solar, average cloud **0.662**, net balance **+2.43**, and **6** unmet events.

15. **Average duration of inverter failures and impact on energy availability.** The average failure duration is **4.0 hours** per event, matching the configured failure duration. These outages increase grid dependence and reduce effective local energy availability, with stronger negative effects during low-generation conditions (especially winter).

3 Development Log

This section summarizes day-to-day progress based on `git log`. In the repository history, `Valkary` corresponds to **José Salcedo Uribe**.

2026-02-11

Participant: Emiliano Hinojosa

- Created the initial project baseline (`b92af01`).

- Implemented the first working structure of the simulator with core components: battery, grid, home, inverter, panel, weather, and `system.py`.
- Added initial configuration and repository ignores needed for the first runnable version.

2026-02-12

Participant: Emiliano Hinojosa

- Implemented weather modeling and integration with panel/system updates (`9003769`).
- Improved battery usage behavior in dispatch logic (`ffe0389`).
- Updated related configuration and generation-flow files to keep the model coherent.

2026-02-16

Participant: Diego Amín Pallares Hernandez

- Refined and cleaned core simulation logic across `inverter.py`, `battery.py`, `system.py`, and strategy comparison (`943aa60`).
- Adjusted configuration and numerical behavior to improve stability and readability.

2026-02-16

Participant: José Salcedo Uribe (`Valkary`)

- Updated project documentation and setup artifacts (`README.md`, `requirements.txt`) and stopped ignoring runtime configuration (`6c6c709`).
- Added helper automation for extracting and aggregating simulation metrics (`simulation/sym_results.py`) to support analysis/reporting (`8bc202d`).

4 Conclusions

This project delivered a functional digital twin of a residential solar-battery-grid system with measurable technical and economic outputs. To make each team member's contribution explicit, the final conclusions are separated below.
Emiliano Hinojosa My main contribution was building the software foundation of the project. I designed the overall architecture and implemented the minimal viable simulator, establishing the component interactions and strategy framework that made the later analysis possible. My goal was to turn the project requirements into a working and extensible system.

Diego Amín Pallares Hernandez My main contribution was improving code quality and simulation robustness. I cleaned and organized key parts of the codebase and introduced practical thresholds in numerical decisions, especially in inverter logic where near-zero floating-point values could cause unstable behavior. This made the simulation more reliable and easier to maintain.

José Salcedo Uribe My main contribution was reporting and result extraction. I developed helper scripts to gather and structure the most relevant metrics and transformed those outputs into clear written analysis for the report. My work ensured the technical results were well documented, interpretable, and ready for presentation.