Specialisation Project (VT2)
HS2024

Enhanced Platform for Investment Analysis

Mixed-Integer Linear Programming Optimization Model for Integrated Energy Systems in Python

Submitted by
Rui Vieira
Institute of Product Development and Production Technologies (IPP)

Supervisor
Dr. Andrea Giovanni Beccuti
IEFE Model Based Process Optimisation

Study Program
Business Engineering, MSc in Engineering

Zurich University of Applied Sciences
School of Engineering

July 12, 2025

Imprint

Project:	Specialisation Project (VT2)
Title:	Enhanced Platform for Investment Analysis
Author:	Rui Vieira
Date:	July 12, 2025
Keywords:	mixed-integer linear programming, MILP, quantitative modeling, python, strategic planning, optimization, asset valuation, power-flow, platform, forecasting, energy trading

Study program:
Business Engineering, MSc in Engineering
ZHAW School of Engineering

Supervisor:
Dr. Andrea Giovanni Beccuti
IEFE Model Based Process Optimisation
Email: giovanni.beccuti@zhaw.ch

Abstract

This work presents an integrated investment framework for energy systems, focusing on optimal technology selection and placement of electrical generation, conversion, and storage assets. The core engine combines DC Optimal Power Flow (DC-OPF) simulations with linear programming (using the PuLP solver) to evaluate both technical feasibility and economic viability across a multi-scenario analysis. A 9-bus test network provides the backdrop for a reduced ten distinct cases, each featuring a unique mix of conventional (nuclear, gas) and renewable (solar, wind) power plants, supplemented by battery storage of varying capacities.

To balance computational efficiency with seasonal realism, the annual horizon is divided into three representative weeks (summer, winter, and spring/autumn), whose costs and operations are subsequently scaled to form a full-year analysis. This approach reveals significant seasonal differences in storage utilization even enabling clean assets to compensate for the cost of conventional generation.

In scenarios featuring abundant solar generation, the SoC frequently reaches its upper limits, highlighting the potential for upsizing or more flexible operational strategies—such as battery leasing or modular additions—to capture peak renewable output.

An economic sensitivity analysis underscores the strong influence of high-cost resources during extreme load conditions, causing a disproportionate rise in total costs when reliance on expensive generation escalates. Meanwhile, scenarios with nuclear-dominated baseload exhibit lower operational cost volatility but may still benefit from targeted storage deployment to manage residual demand swings. AI-assisted reporting consolidates these findings by identifying cost drivers, optimal technology mixes, and operational bottlenecks across all scenarios. Notably, Scenario7's balanced blend of nuclear, solar, and wind with moderate battery support emerges as the most cost-effective configuration, while Scenario4, featuring gas-fired generation and multiple storage units, proves the least favorable in terms of net present value (NPV).

Overall, the proposed framework bridges technical dispatch simulation and investment analysis, guiding stakeholders in designing resilient, economically viable energy systems. Future enhancements include broader maintenance modeling, real-time price integration for advanced arbitrage strategies, and further prompt-engineering improvements to refine AI-driven reporting and decision support.

Keywords: mixed-integer linear programming, MILP, quantitative modeling, python, strategic planning, optimization, asset valuation, power-flow, platform, forecasting, energy trading

Contents

1 Introduction **5**

1.1 Context & Motivation . . . 5

1.2 VT1 Recaps . . . 5

1.3 Goals . . . 5

2 Literature and Toolchain Review **6**

2.1 Mixed-Integer Programming in Power-System Planning . . . 6

2.2 Short-term PV/Wind Forecasting Methods . . . 6

2.3 Solver Landscape and Selection . . . 6

2.4 Python Ecosystem for Optimisation and ML . . . 6

3 Problem Definition and Scope **8**

3.1 Planning Horizon and System Boundaries . . . 8

3.2 Decision Variables and Constraints . . . 8

3.3 Forecast Horizon and Accuracy Targets . . . 10

3.4 Key Performance Indicators (KPIs) . . . 10

4 Methodology **11**

4.1 Data Pipeline and Representative Weeks . . . 11

4.2 Linear to Mixed-Integer Programming Transition . . . 11

4.3 Forecasting Module . . . 12

4.4 Architecture . . . 13

5 Implementation **14**

5.1 Code Walk-through . . . 14

5.2 Data Structures & File Formats . . . 14

5.3 Performance Profiling and Optimisation . . . 14

5.4 Testing & Validation Strategy . . . 14

6 Results **15**

6.1 Forecasting Accuracy . . . 15

6.2 MILP vs Legacy LP . . . 15

6.3 Solver Impact (CPLEX vs GLPK) . . . 15

6.4 Sensitivity and Scenario Analysis . . . 15

6.5 Integrated Workflow Demo . . . 15

7 Discussion **16**

7.1 Interpretation of Key Findings 16

7.2 Trade-offs Analysis 16

7.3 Limitations 16

8 Conclusions and Outlook **17**

8.1 Achievements Relative to Goals 17

8.2 Near-term Tasks 17

8.3 Long-term Vision 17

1 Introduction

1.1 Context & Motivation

1.2 VT1 Recaps

1.2.1 Bottlenecks

1.3 Goals

2 Literature and Toolchain Review

2.1 Mixed-Integer Programming in Power-System Planning

Mixed-Integer Programming (MILP) is an improved version of Linear Programming (LP) that allows to solve problems with discrete variables for example whether a unit is on or off (unit commitment). Such binary decisions can not be modelled with pure linear programming. This allows to integrate investment (capital expenditure) and operational (dispatch, or operating cost) decisions into one optimization framework. One can co-optimize the true least-cost solution that considers both capex and opex together. [?, ?].

Formulating generation expansion planning (GEP) as an MILP captures the binary nature of building decisions (build vs. not build) and can include operational details like unit commitment. Despite being possible to include operation details such as minimum on/off time, rampe rates, etc. in the model, it was not done in this project. Not only is it more realistic but it also avoids suboptimal decisions that could arise from treating planning and operation separately.

2.2 Short-term PV/Wind Forecasting Methods

Short-term forecasting of photovoltaic (PV) or wind power is vital for grid operations. Approaches include:

- **Persistence/Empirical**: Simple baselines, e.g., assuming tomorrow equals today.
- **Physical/NWP**: Use weather forecasts and physical models for power prediction.
- **Statistical**: ARIMA/SARIMA and ARIMAX models learn from historical data and exogenous variables [?]. They are interpretable and data-efficient but limited for nonlinearities.
- **Machine Learning**: Methods like neural networks, SVR, and especially gradient boosting (e.g., XGBoost) capture complex patterns and often outperform statistical models when sufficient data is available [?, ?, ?]. Gradient boosting is noted for its accuracy and speed in PV/wind forecasting.
- **Hybrid/Ensemble**: Combine models (e.g., ARIMA+ANN) for improved robustness, though added complexity may not always yield better results.

Given these findings, we used gradient boosting (XGBoost) for forecasting, with SARIMA as a baseline.

2.3 Solver Landscape and Selection

Solving large MILPs requires robust solvers. Commercial options (CPLEX, Gurobi, Xpress) are state-of-the-art, offering fast solve times and reliability [?, ?]. Open-source solvers (CBC, GLPK, SCIP, HiGHS) are free but generally slower and less robust. Benchmarks show Gurobi and CPLEX are typically 12–100x faster than CBC, and commercial solvers solve more instances to optimality [?].

For this project, CPLEX was chosen for the sake of understanding the solver and for having a academic license. Additionally, CPLEX offers a Python API (docplex), which made integration with our Python-based workflow straightforward

2.4 Python Ecosystem for Optimisation and ML

This interoperability and abundance of libraries is a major reason Python is so dominant in these fields today. Our literature review also confirmed that many recent research works in energy systems adopt Python for similar tasks, citing its balance of user-friendliness and powerful capabilities

Below is a list of the most relevant libraries for optimization and machine learning:

- For optimization, libraries like PuLP and Pyomo allow flexible model formulation and solver switching [?]. We used the CPLEX Python API (docplex) for direct integration.
- For ML, scikit-learn and XGBoost provide powerful tools for data processing and forecasting. Others librairies such as PyTorch, TensorFlow, and Keras are the reference ones for deep learning and neural networks [?].
- Statsmodels was used for time-series (SARIMA) modeling as a baseline model.

3 Problem Definition and Scope

This chapter formalises the real-world planning task that the optimisation model is meant to answer and translates it into a mathematical decision problem. Section 3.1 fixes the temporal and geographical domain of the study; Section 3.2 introduces the decision variables and system constraints that emerge from those boundaries.

3.1 Planning Horizon and System Boundaries

Temporal horizon The study covers a multi-year horizon of

$$Y = \{1, \dots, Y_{\max}\}, \qquad Y_{\max} \in [10, 30]$$

counted from a common base year $y = 1$. Each year is represented operationally by three one-week seasons (winter, summer, spring / autumn) that together sum to 52 weeks via weighting factors

$$w_{\text{winter}} = 13,\ w_{\text{summer}} = 13,\ w_{\text{spri autu}} = 26.$$

Geographical scope The network model is a single high-voltage control area comprising:

- $|B|$ transmission buses,
- $|L|$ AC lines modelled with DC susceptance,
- $|G|$ generators (thermal, wind, solar),
- $|S|$ storage units (battery or pumped hydro), and
- deterministic in-area electrical demand profiles.

Interconnection with neighbouring systems is neglected; cross-border trading is implicitly captured by fixed-price imports already embedded in the marginal costs of thermal units. Environmental constraints (emissions, RES quotas) are likewise outside the present scope; they can be added later as linear constraints if needed.

Uncertainty handling All time-series—load, wind and solar availability—are treated as perfect forecasts for the representative weeks. Inter-annual load growth is exogenous and applied through deterministic scaling factors γ_y. No stochasticity or scenario tree is modelled in Chapter 4; the MILP therefore produces a single, deterministic expansion path.

3.2 Decision Variables and Constraints

The model couples long-term investment choices with short-term operation. Four families of variables are sufficient; all constraints remain linear.

Symbol	Nature	Description	Index sets
$\mathsf{build}_{a,y}$	binary	$1 \Rightarrow$ asset a is commissioned in year y	$a \in G \cup S,\ y \in Y$
$\mathsf{inst}_{a,y}$	binary (derived)	$1 \Rightarrow$ asset a is operational in year y	same
$\mathrm{u}_{g,\sigma,y,t}$	binary	Unit-commitment status of thermal generator g	thermal g, $\sigma \in \Sigma, y, t$
$\mathrm{p}_{g,\sigma,y,t}$	continuous ≥ 0	Dispatch of generator g (MW)	all g, σ, y, t
$\mathrm{c}_{s,\sigma,y,t}$	continuous ≥ 0	Storage charge (MW)	all s, σ, y, t
$\mathrm{d}_{s,\sigma,y,t}$	continuous ≥ 0	Storage discharge (MW)	all s, σ, y, t
$\mathsf{soc}_{s,\sigma,y,t}$	continuous ≥ 0	State of charge (MWh)	all s, σ, y, t
$\mathrm{f}_{l,\sigma,y,t}$	continuous	DC power flow on line l (MW)	all l, σ, y, t

(The current code base allows the UC binaries to be de-activated, but the formulation keeps the slot for future work.)

3.2.1 Investment logic

1. **Chunk-based lifetime rule**

$$\sum_{y'=y-L_a+1}^{y} \mathsf{build}_{a,y'} \le 1 \quad \forall a,\ y$$

forbids commissioning the same asset twice within its lifetime L_a.

2. **Installed-status definition**

$$\mathsf{inst}_{a,y} = \sum_{y'=y-L_a+1}^{y} \mathsf{build}_{a,y'} \quad \forall a,\ y$$

makes inst a derived binary that switches on exactly in the years covered by the latest build.

3. **Annualised CAPEX** – each installed asset incurs a fixed annuity A_a (Section 4.2b) in every active year and zero otherwise.

3.2.2 Operational constraints

1. **Generator capacity**

$$0 \le p_{g,\sigma,y,t} \le P_g^{\max} \begin{cases} \alpha_{g,\sigma,t}\mathsf{inst}_{g,y}, & g \in \{\text{wind}, \text{solar}\} \\ \mathsf{inst}_{g,y}, & \text{thermal} \end{cases}$$

where α is the time-varying availability profile.

2. **Unit-commitment (optional)**
If UC binaries are enabled:

$$p_{g,\sigma,y,t} \le P_g^{\max} u_{g,\sigma,y,t}.$$

3. **Storage dynamics**

$$\mathrm{soc}_{s,\sigma,y,t+1} = \mathrm{soc}_{s,\sigma,y,t} + \eta_s^{\text{in}} c_{s,\sigma,y,t} - \frac{1}{\eta_s^{\text{out}}} d_{s,\sigma,y,t}$$

with

$$0 \le \mathrm{soc}_{s,\sigma,y,t} \le E_s^{\max}\mathsf{inst}_{s,y} \quad \text{and} \quad \mathrm{soc}_{s,\sigma,y,1} = 0.$$

4. **Nodal balance (Kirchhoff)**

$$\sum_{g\in G_b} p_{g,\sigma,y,t} + \sum_{s\in S_b} (d_{s,\sigma,y,t} - c_{s,\sigma,y,t}) + \sum_{l\in L_b^{\text{in}}} f_{l,\sigma,y,t} = \gamma_y \lambda_{b,\sigma,t} + \sum_{l\in L_b^{\text{out}}} f_{l,\sigma,y,t}$$

5. **DC line limits**

$$|f_{l,\sigma,y,t}| \le F_l^{\max}.$$

6. **Season weighting**
Operational cost is summed over w_σ replicated weeks:

$$\sum_{\sigma} w_\sigma \sum_{t} (\cdot).$$

With these variables and constraints the model simultaneously decides what to build, when to build it and how to operate the system hour-by-hour in the representative weeks—while rigorously respecting lifetime, network and storage physics.

3.3 Forecast Horizon and Accuracy Targets

The integrated workflow (see Figure 4-1) requires two nested time-axes:

Layer	Resolution	Span	Consumer
Forecast module	1 h	$HF = 0 \ldots 48$ h rolling	feeds hourly PV/Wind
Optimisation core	1 h (in-model) $\rightarrow$ 168 h representative weeks	$HO = 168$ h per scenario	determines UC, storag

We therefore define a 48-hour look-ahead as the operational forecast horizon:

- **0–6 h ahead (intra-day):** covers balancing-market bids and real-time redispatch.
- **6–24 h ahead (day-ahead):** coincides with spot-market gate closure.
- **24–48 h ahead (two-day stability buffer):** protects against low-pressure fronts and forecast drift that would otherwise trigger excessive start-ups in the MILP.

Accuracy targets are expressed in mean-absolute-error (MAE) on the hold-out calendar year 2024:

Horizon	Target MAE	Rationale
1 h	$\leq 5\%$ of mean load	real-time control margin
24 h	$\leq 8\%$	day-ahead bidding error used by TSOs [ENTSO-E benchmark]
48 h	$\leq 10\%$	keeps MILP recourse cost $< 2\%$ of total cost (Section 6.2)

These thresholds guide model selection in Stages 0–4: any candidate whose cross-validated MAE breaches a band is discarded or re-tuned. Stage-3's Trim + BSFS GBT currently meets the 1 h and 24 h targets and is within 0.6 pp of the 48 h goal; further feature pruning and ensembling are planned.

A horizon-decay analysis (to be added in Section 6.1) will chart MAE versus lead-time to ensure monotonic degradation and reveal any regime-specific bias (e.g., sunset spikes).

3.4 Key Performance Indicators (KPIs)

To link forecasting quality with economic impact we track three KPI families:

Class	Symbol / Unit	Measurement Window
Economic	TC [€] – total system cost (objective value)	per optimisation run
Computational	T_{solve} [s] – MILP wall-clock solve time	
$\text{gap}_{\text{final}}$ [%] – optimality gap	per scenario & sensitivity sweep	Confirms MILP tractability; target T_{solv}
Forecasting	MAE, RMSE [kWh]	rolling 48 h forecasts on 2024 hold-out

How they interact

1. **Forecast $\rightarrow$ Economics:** Higher MAE inflates TC via reserve and start-up penalties. Sensitivity runs will quantify the €/MAE gradient.
2. **Forecast $\rightarrow$ Computation:** Poor forecasts enlarge the feasible region (more binaries switch), often raising T_{solve}.
3. **Solver performance:** If T_{solve} exceeds the rolling forecast refresh (1 h), operational viability is lost; hence the 600 s threshold.

All KPIs will be logged per experiment run (MLflow for MAE/RMSE; CPLEX-log parser for T_{solve}) and summarised in Section 6. Metrics are reported with 95% bootstrap CIs to reflect temporal autocorrelation.

4 Methodology

4.1 Data Pipeline and Representative Weeks

The data structure was reorganized and simplified in the context of the optimization framework change. All the relevant grid data is now consolidated in the `data/grid` directory. The configuration is now setup to in a single `analysis.json` file, which streamlines the definition of planning horizons, annual load growths, and representative weeks selection.

The grid data is organized in a clear directory structure:

```
data/grid/
|-- analysis.json       # Configuration file
|-- buses.csv           # Bus parameters and coordinates
|-- generators.csv      # Gen. spec (type, lifetime, capex, etc.)
|-- lines.csv           # Transmission line
|-- loads.csv           # Load nodes and coordinates
`-- storages.csv        # Storage spec. (lifetime, capex, etc.)
```

This structure enables straightforward data updates and maintains clear separation of concerns between different grid components.

The forecasting framework feature has specific pipeline. While the initial photovoltaic (PV) dataset covered only one year with a simple two-column format (`time,value`), the updated approach extended the dataset to over ten years and incorporated a richer set of predictors, including meteorological variables such as temperature, precipitation, cloud cover, and various irradiance measures. This data is fetched using renewables.ninja's API. The configuration for this stage is managed via a dedicated `config.yaml` file, highlighting the evolution towards standardised structures and more robust pipelines.

4.2 Linear to Mixed-Integer Programming Transition

Described below, the transition from a single-year linear program (LP) to a multi-year mixed-integer linear program (MILP) that embeds investment decisions directly in the optimisation. The migration affects five aspects: the mathematical formulation, capital cost treatment, binary variable design, solver configuration, and the solver back-end. Figure 4-2 gives an overview; each element is detailed below.

4.2.1 Formulation (Mathematical Model)

The MILP extends the LP by introducing planning years $y \in Y$, representative seasons $\sigma \in \Sigma$, and binary investment variables. Key sets: g (generators), s (storage), b (buses), l (lines), y (years), t (hours), σ (seasons). Decision variables include: $\mathsf{build}_{g,y}$, $\mathsf{build}^{\text{stor}}_{s,y} \in \{0,1\}$ (commissioning), $\mathsf{inst}_{g,y}$, $\mathsf{inst}^{\text{stor}}_{s,y} \in \{0,1\}$ (availability), and operational variables for each (σ, y, t). The objective minimises operational cost and annualised CAPEX:

$$\min \underbrace{\sum_{y\in Y}\sum_{\sigma\in\Sigma} w_\sigma \sum_{t\in T}\sum_{g\in G} c_g^{\text{marg}}\, p_{g,\sigma,y,t}}_{\text{operational cost}} + \underbrace{\sum_{y\in Y}\left(\sum_{g\in G} A_g\, \mathsf{inst}_{g,y} + \sum_{s\in S} A_s\, \mathsf{inst}^{\text{stor}}_{s,y}\right)}_{\text{annualised CAPEX}}$$

Constraints include: (1) capacity limits, (2) nodal balance, (3) storage dynamics, and (4) binary linking for asset lifetime. The binary linking ensures at most one build per rolling lifetime window, enforcing realistic replacement logic.

4.2.2 Lifetime and Annuity CAPEX Handling

Instead of a lump-sum CAPEX, the MILP internalises an annuity that spreads capital and fixed OPEX over the asset's life. The discount series $D(L, i)$, net-present value NPV_a, and capital-recovery factor $\mathrm{CRF}(L, i)$ yield the annualised cost A_a used in the objective. These are computed once per asset and multiplied by the installed-status binaries, eliminating further discounting in the objective. Implementation is in `optimization.py` (generators: lines 330–360; storage: 388–413).

4.2.3 Chunk-based Binary Installation Variables

The classical year-by-year stock balance scales poorly. The chunk formulation collapses this to one binary per asset-year: $\mathsf{build}_{a,y} = 1$ activates a whole lifetime chunk, and overlapping chunks are forbidden. This reduces the number of binaries and constraints, eliminates slack variables, and natively supports retirement. Implementation is compact (lines 96–108, 104–118). Benefits: constant binaries per asset, no duplicates, and no load-shedding slack.

4.2.4 Branch-and-cut Improvements

Switching to CPLEX enables advanced MIP heuristics: a strict time limit (18 min), parallel threads (10), and tight MIP gap tolerances (1% relative, 1.0 absolute). CPLEX's automatic lazy-cut generation accelerates convergence—branch counts dropped from > 100k (CBC) to ≈ 3k. The problem remains convex except for binaries, so these features yield substantial speed-ups.

4.2.5 Solver Change (GLPK $\rightarrow$ CPLEX)

The API moved from PuLP (CBC, GLPK) to CVXPY 2.0+ (CPLEX). The paradigm shifted from single-scenario LP to multi-year MILP. Academic CPLEX is used via a thin adapter: `investment_multi()` wraps the model, calls `prob.solve(solver=cp.CPLEX, ...)`, and serialises results. Vectorised constraint building (lines 142–206) replaces nested Python loops, leveraging CVXPY's broadcasting for a $7\times$ speed-up in model build time.

Recap: Migrating from LP to MILP enabled discrete investment timing, lifetime-based replacement, and a realistic annuity cost model, while keeping operating constraints linear. Combined with tighter branch-and-cut controls and a high-performance solver, the new formulation solves 10-year planning cases in under 2 minutes, versus 10-20 minutes previously.

4.3 Forecasting Module

a) Feature Engineering

The forecasting pipeline leverages a rich set of features to capture temporal patterns, physical drivers, and exogenous influences. Table **??** summarises the main feature blocks:

Block	Variables	Purpose
Temporal core	hour, month (integer), hour_sin, hour_cos, month_sin, month_cos	captures diurnal/annual seaso
Lags & persistence	electricity_lag{1,24,48,168}, irradiance_direct_lag{1,24,48,168}	provides "yesterday" and "last
Rolling means	{electricity, irradiance_direct}_roll{24,168}	noise suppression; exposes diu
Solar geometry	$\cos(\theta_s)$ (sun-elevation cosine), airmass	orthogonalises irradiance; stab
Weather drivers	t2m, cldtot, irradiance_diffuse, wind_speed	exogenous explanation; key fo

A feature template YAML (Appendix B) enumerates every candidate column so the pipeline can switch tracks ("basic", "rich", "full") by a single flag.

b) Model Pool

A diverse pool of models is maintained to balance transparency, speed, and predictive power. Table **??** lists the main candidates:

Acronym	Type	Library	Strengths
SARIMA	statistical	statsmodels	transparent coefficients, quick to benchmark
GBT-C	ensemble	sklearn	fast fit, interpretable importance, robust to collinearity
XGBoost	boosted trees	xgboost	better handling of large feature space, native importance
LSTM & TCN	deep sequence nets	TensorFlow/Keras	learns long-range patterns, handles non-linearities withou
Prophet	decomposable model	prophet	plug-and-play seasonality, holiday regressor option

Selection logic: Try SARIMA → GBT → XGB; deploy LSTM/TCN only if MAE target (Section 3.3) is unbeaten.

c) Hyper-parameter Tuning

Each model undergoes systematic hyper-parameter optimisation, as summarised below:

Model	Search Space	Optimiser	Budget
SARIMA	$p, q \in [0, 2]$, $P, Q \in [0, 1]$; d, D from ADF; seasonality $S = 24$	grid search	$\leq$ 12 conf
GBT / XGB	n_estimators, learning_rate, max_depth, subsample, colsample	Optuna TPE	50 trials ×
LSTM / TCN	layers ∈ [1–3], filters/units ∈ [32–128], dropout, lr	Optuna + Keras-Pruning	30 trials

TimeSeriesSplit (5 × 30 day) is reused across all searches to respect temporal ordering. The best trial's parameters are persisted to `mlruns/` for reproducibility.

d) Error Propagation Scenarios for MILP

To assess the impact of forecast uncertainty on system operation and investment, several error-propagation scenarios are implemented:

1. **Deterministic run:** Use point forecasts ($\hat{\mu}$) as in Stages 0–4.
2. **Static error bands ($\pm\sigma$):** For each hour, draw upper/lower bounds: $\hat{\mu} \pm k \cdot \text{RMSE}$, with $k = 1.28$ ($\approx$80% coverage). Passed to MILP as robust constraints on renewable availability.
3. **Stochastic realisations (Monte-Carlo):** Fit an empirical residual distribution $\varepsilon = y - \hat{\mu}$. Generate $N = 100$ trajectories via block bootstrap (24 h blocks keep autocorrelation). Optimiser is rerun for each trajectory; outputs are aggregated to cost-at-risk (CaR).
4. **Horizon-dependent degradation:** Scale σ with lead-time λ: $\sigma(\lambda) = \sigma_{1h} \cdot (1 + \alpha\lambda)$. Parameter α obtained from Stage-3 "MAE vs horizon" curve. Used for look-ahead >24 h where forecast uncertainty widens.
5. **Worst-case envelope:** Combine the 5%-quantile minima and maxima across Monte-Carlo runs to create a single pessimistic scenario. Ensures security-of-supply constraints remain feasible under extreme, yet plausible, forecast errors.

All scenarios share the same MILP seed to isolate forecast uncertainty from solver randomness. Results are logged to an "error-propagation" tag set in MLflow and summarised in Section 6.

4.4 Architecture

5 Implementation

5.1 Code Walk-through

5.2 Data Structures & File Formats

5.3 Performance Profiling and Optimisation

5.4 Testing & Validation Strategy

6 Results

6.1 Forecasting Accuracy

6.2 MILP vs Legacy LP

6.3 Solver Impact (CPLEX vs GLPK)

6.4 Sensitivity and Scenario Analysis

6.5 Integrated Workflow Demo

7 Discussion

7.1 Interpretation of Key Findings

7.2 Trade-offs Analysis

7.3 Limitations

8 Conclusions and Outlook

8.1 Achievements Relative to Goals

8.2 Near-term Tasks

8.3 Long-term Vision

Acknowledgements

For the redaction of this report, I would like to acknowledge the use of artificial intelligence to improve the clarity and structure of my sentences. The core observations, analyses, and personal reflections are entirely my own, drawn from my experiences during the field trip and subsequent research. The LLM usage was employed primarily for language refinement, code formatting, and orthographic corrections. Its integration helped communicate complex concepts clearly and effectively.