Programming Experience

PROJECT 1 - National-Scale PV Curtailment & Economic Modeling (Python)

Role: Co-author, Published in Sustainable Energy Technologies and Assessments (Q1), 2025.
Contribution: Software, Data Processing, Modeling, Validation.

For this peer-reviewed publication, I developed the complete Python workflow used to process national PV curtailment data, synchronize it with electricity market prices, compute energy and revenue losses, and create the time-series visualizations and statistical summaries shown in Figures 2, 3, and 4 of the article. My code formed the computational backbone of the study's descriptive analysis and economic interpretation.

I processed more than two years of high-frequency operational data from the Spanish system operator (REE e·sios), including PDBF (day-ahead scheduled PV production), PDVP (provisionally feasible PV output), P48 (real-time operating program), and programmed vs. real-time curtailment indicators. Using Python (pandas), I built scripts to parse raw JSON/CSV data files, standardize inconsistent date formats, correct CET/CEST daylight-saving conflicts, remove negative or invalid values, interpolate missing hours, and ensure perfect hourly alignment across all datasets. This data-harmonization stage was essential for reliable curtailment calculations.

I implemented all algorithms used to compute daily programmed and real-time curtailment ratios, shown in Figure 2 (top plot). Programmed curtailment was computed as (PDBF - PDVP)/PDBF, while real-time curtailment was computed as (PDVP - P48)/PDBF. I also implemented the computation of daily economic losses, shown in Figure 2 (bottom plot), calculated as the sum across hours of lost programmed energy multiplied by hourly day-ahead market prices. These resulting time series revealed curtailment seasonality, increasing curtailment intensity over time, and the emergence of large curtailment spikes in 2023 - 2024.

For Figure 3 (top plot), I developed the monthly aggregation pipeline that calculates, for each month, the fractional share of total curtailed energy and the fractional share of total economic loss. This required grouping by month, merging energy and price datasets, and removing outliers. I also generated the monthly distribution of hourly market prices (Figure 3, bottom plot) using boxplots normalized to the mean annual price. For this, I wrote normalization algorithms, applied group-by operations, computed monthly statistical distributions, and implemented outlier filtering using the interquartile range (IQR) method.

For Figure 4, I built the scripts that compute the hourly contribution of each hour of the day to total curtailed energy and revenue loss across the entire dataset (October 2022 - September 2024). The resulting bar charts (Figure 4, top plot) highlight the concentration of curtailment between 10:00 and 16:00. I also computed hourly boxplots of normalized electricity prices (Figure 4, bottom plot), capturing intra-day price dynamics and the characteristic midday price depression associated with high PV generation. These tasks required higher-resolution hourly grouping, normalization by daily mean prices, and construction of hourly boxplots from approximately 17,000 hourly price values.

I independently created all visualizations using matplotlib and seaborn. This included configuring multi-axis layouts, customizing tick locators, ensuring consistent color-coding, implementing gridlines, refining visual clarity, normalizing scales, and exporting publication-quality high-resolution figures. All Figures 2 - 4 in the published article were produced through my Python scripts.

To ensure numerical correctness, I implemented several validation and reproducibility checks. These included verifying that aggregated hourly values matched REE daily totals, cross-validating the classification of programmed vs. real-time curtailment, testing robustness to the removal of weekends and holidays, and confirming the stability of typical hourly profiles across different data windows. These checks guaranteed consistent and accurate results.

Tools Used: Python, pandas, numpy, matplotlib, seaborn, python-dateutil; data from REE e·sios (curtailment and PV production) and OMIE (day-ahead market prices).

PROJECT 2 - Master's Thesis: Energy and Economic Analysis of Utilizing PV Curtailment Through a 4-Hour Lithium-Ion Battery Storage System (Python)

My master's thesis expanded the work of my co-authored PV curtailment publication by developing a complete computational framework for analyzing photovoltaic curtailment, modeling curtailment blocks, estimating battery storage requirements, evaluating revenue losses, and performing system-level storage optimization. I independently designed and implemented the entire coding pipeline in Python, which processed multi-year electricity system datasets, constructed advanced time-series models, and generated every analytical figure and quantitative result presented in the thesis. I built a comprehensive Python data-processing workflow integrating three large datasets: REE e·sios data (hourly PV production, scheduled generation, feasible plans, real-time output, and curtailment), OMIE day-ahead electricity market prices, and long-term PV yield data from PVGIS. This involved parsing raw JSON/CSV files, resolving inconsistent use of CET/CEST timezones, correcting missing or invalid values, and merging all data sources into a unified, perfectly synchronized hourly dataset. I wrote Python scripts that harmonized timestamps, interpolated missing hours, removed anomalous values, created multi-year averages to form national "typical hourly PV profiles," and produced the clean data environment used for all subsequent modeling.

I implemented all algorithms used to compute hourly and daily programmed and real-time curtailment levels. Programmed curtailment was calculated as (PDBF − PDVP)/PDBF, and real-time curtailment as (PDVP − P48)/PDBF. I coded routines to compute daily and monthly curtailment, seasonal patterns, and long-term curtailment trends across 2022 - 2024. I also wrote the full revenue-loss computation pipeline, which multiplies curtailed energy by hourly prices to generate datasets for economic analysis. These computations formed the input for all visualizations and economic results in the thesis.

A core innovation of my thesis was the development of a block-based curtailment model. I programmed a custom algorithm to detect and analyze "curtailment blocks," defined as continuous sequences of consecutive curtailed hours. My code identified each block's start time, duration, and total curtailed energy, sorting and storing all 919 blocks in the dataset. I then wrote additional scripts to generate analytical plots such as block-energy-over-time charts, block-duration vs. energy scatter plots, and block-size vs. impact curves. These revealed the nonlinear escalation of curtailment intensity with block duration and identified extreme events, including a 39-hour curtailment block with over 26,000 MWh curtailed.

I built a storage-capacity modeling engine to evaluate how much energy storage is needed to absorb curtailed PV energy. This included algorithms that compute required capacity at different percentiles of block size, enabling analysis of 10th through 100th percentile storage needs after removing the smallest 11% of blocks. I also created a high-resolution capacity-sweep simulation, testing thousands of possible storage-capacity values (from 100 MWh to over 13,000 MWh), computing how many blocks each size could absorb, and producing a continuous performance curve linking storage size to curtailment coverage. I then implemented an elbow-point detection algorithm using discrete derivatives and slope thresholds to identify the optimal trade-off between cost and coverage. This algorithm located an elbow point near 4,200 MWh, which mitigates approximately 80% of curtailment at a fraction of the cost of full storage build-out.

I developed advanced electricity-price behavioral models in Python to analyze the relationship between curtailment and market conditions. Using hourly OMIE data, I generated daily price-evolution curves, typical hourly price patterns, and monthly price-distribution boxplots. These required group-by transformations, normalization by daily or yearly averages, and outlier filtering using the interquartile-range method. The resulting figures captured midday price collapses, evening price peaks, and seasonal price shifts, allowing a full integration of energy, curtailment, and economic signals.

I wrote the complete revenue-modeling engine used to evaluate the monetary impact of curtailment on PV operators. Revenue loss per block was computed as the sum of curtailed energy multiplied by hourly prices. I then developed the scatter plots comparing block energy to revenue loss and block duration to revenue loss, which revealed nonlinear relationships and showed that long-duration events contribute disproportionately to total economic losses.

I also implemented a battery-economic model in Python to assess the feasibility of a 4-hour lithium-ion storage system. This model incorporated CAPEX, round-trip efficiency, revenue recovery from shifting curtailed energy to higher-priced hours, and scenario simulations across multiple storage sizes. The code evaluated cost-benefit trade-offs, quantified expected revenue recovery for all block categories, and identified economically preferable storage configurations.

All visualizations in the thesis were fully coded and generated by me using matplotlib, seaborn, numpy, and custom plotting utilities. These included time-series graphs, block-energy analyses, block-duration scatter plots, boxplots of hourly and monthly price distributions, storage-capacity performance curves, elbow-point graphs, and curtailment-revenue mappings. I ensured publication-quality figure styling, consistent color schemes, normalized axes, and high-resolution output.

In summary, I independently developed all computational components of the thesis, including data ingestion, curtailment block modeling, electricity-price behavioral analysis, revenue computation, storage-capacity optimization, battery-economic modeling, and all scientific visualizations. The thesis is built upon an end-to-end modeling framework entirely implemented in Python, demonstrating full ownership of the analytical, algorithmic, and computational aspects of the research.

PROJECT 3 - Automated Image Analysis & Machine Learning on 13,200 PV Soiling Micrographs (ImageJ + Python)

From February to July 2024, I conducted a large-scale computational analysis of **13,200 microscope images** of PV soiling samples collected across **nine global sites** as part of an international outdoor soiling campaign, under the supervision of Prof. Leonardo Micheli. I designed and implemented an **automated end-to-end workflow** combining ImageJ scripts, Python-based statistical modeling, and machine learning preprocessing to extract particle morphology parameters and correlate them with optical transmittance losses.

I first created a fully automated **ImageJ macro pipeline** capable of processing thousands of BMP micrographs with consistent thresholding, segmentation, and particle identification. The pipeline standardized scale settings, applied the "Triangle" thresholding method (identified as the most stable through CV testing), and extracted **19 particle-level morphological parameters** for every image, including equivalent diameter, Feret's diameter, minimum Feret, circularity, roundness, solidity, aspect ratio, gray-value statistics, particle count, and fractional area coverage. This automation ensured complete reproducibility across a dataset that would be infeasible to process manually.

I then developed a **Python analytical engine** (Python 3.10, SciPy, NumPy, pandas) to ingest the ImageJ output, clean and harmonize parameter datasets across all sites, and compute statistical descriptors (mean, median, standard deviation, coefficient of variation). I implemented CV-based robustness testing for each morphological parameter, enabling a systematic evaluation of which particle descriptors were stable or highly variable across different regions, environmental conditions, and accumulation periods. This revealed that while size-based parameters (diameter, Feret) exhibited high variability, reaching CV values up to 350% metrics such as circularity, roundness, solidity, area coverage, and particle count remained far more robust.

To assess the feasibility of predicting optical performance from image-derived metrics, I developed a **parameter - transmittance correlation module** using Python. For each micrograph, I computed mean parameter values and directly matched them with same-day transmittance measurements. I then generated best-fit linear models between each parameter and transmittance and ranked their predictive strength using the **coefficient of determination (R^2)**.

This approach expanded beyond traditional reliance on surface coverage and identified the parameters (area, diameter, mean, mode, Feret) with the strongest predictive value at multiple sites. Conversely, parameters such as FeretX and FeretY displayed negligible correlation, demonstrating their limited diagnostic relevance.

I produced all visualizations, boxplots, CV distributions, cross-site comparisons, and parameter - transmittance correlation plots, using matplotlib and seaborn. These figures revealed regional differences in particle morphology, such as large irregular dust in desert sites (Qatar, Jordan) versus compact, rounded particles in temperate regions (Spain, Australia). The computational analysis showed that site-specific environmental behavior must be considered when designing predictive soiling models and maintenance strategies.

In summary, I created a **fully reproducible computational framework** integrating ImageJ automation, Python-based statistical analysis, correlation modeling, and large-scale visualization for one of the largest soiling micrograph datasets ever processed. This work provided robust insights into the relationship between particle morphology and optical losses and produced the complete analytical foundation for the final research report.

Tools Used: ImageJ macro scripting; Python (pandas, NumPy, SciPy, matplotlib, seaborn); BMP micrograph datasets from nine global PV sites; optical transmittance measurements.

PROJECT 4 - Statistical Signal Processing & Spectral Analysis Using MATLAB

In this project, I independently performed a complete computational analysis of eight experimental data sets (each containing 10,000 time-equidistant samples) using **MATLAB**. The goal was to identify the underlying physical nature of unknown signals, turbulent, sinusoidal, noise, and mixed signals, by applying advanced statistical, correlation-based, and spectral techniques. I developed all analysis scripts myself, covering the full workflow from raw data ingestion to final classification.

I began by constructing MATLAB routines to calculate the **first four statistical moments** for each dataset: mean, standard deviation, skewness, and kurtosis. These statistical indicators were used to initially differentiate signal types, for example, identifying Gaussian noise through a kurtosis value near 3 and skewness near zero, and recognizing pure sinusoidal signals through near-zero mean and symmetric distributions. I also implemented probability density function (PDF) estimation using histfit to visually compare empirical distributions against Gaussian behavior.

To analyze temporal structure, I coded custom functions to compute **auto-correlation sequences** for each dataset. Using nested loops and vectorized operations, I evaluated the similarity of each signal with itself over increasing time lags, normalized by the variance and effective sample count. The auto-correlation method allowed me to distinguish fast-changing noise (correlation dropping immediately to zero) from periodic sinusoidal signals (high correlation at multiples of the period) and from turbulent or turbulent+noise signals exhibiting broader, non-periodic structures.

I extended this analysis by writing scripts to compute **cross-correlation functions** between selected pairs of signals (e.g., sinusoidal vs. sinusoidal+noise, turbulent vs. turbulent+noise). These functions quantified how the addition of noise altered temporal similarity patterns and provided insight into how composite signals retain features of their pure components. For example, I demonstrated that the cross-correlation structure of a sinusoidal+noise signal preserves the shape of the pure sinusoid's auto-correlation curve but exhibits reduced amplitude due to noise contamination.

To investigate signal composition in the frequency domain, I implemented **power spectral density (PSD) analysis** using the Fast Fourier Transform (fft). I computed magnitude spectra, applied logarithmic scaling, and analyzed frequency peaks to classify signals. Pure sinusoidal signals exhibited sharp dominant frequencies corresponding to their fundamental period, whereas noise signals showed broadband, high-frequency characteristics. Mixed turbulent+noise signals demonstrated hybrid spectral features with both broadband content and elevated peaks. I also used spectral analysis to motivate filter design approaches for noise removal.

Throughout the project, I experimented with **filters, bandwidth changes, and windowing functions**, investigating how different filtering strategies influenced the clarity and interpretability of noisy signals. This included examining how sample size affects statistical moment accuracy and comparing algorithms for PSD estimation.

All figures, including time-domain plots, PDFs, auto-correlation curves, cross-correlation graphs, and spectral density plots, were generated using MATLAB. I organized all custom scripts into reproducible modules included in the final technical report.

In summary, I developed a comprehensive MATLAB-based signal analysis toolkit capable of statistical characterization, temporal correlation analysis, frequency-domain decomposition, filtering, and final classification of unknown signals. This project strengthened my expertise in numerical methods, digital signal processing, and scientific programming applied to experimental fluid mechanics.

Tools Used: MATLAB, statistical moment functions, custom auto-correlation and cross-correlation algorithms, FFT-based spectral analysis, filtering/windowing routines.

The four projects presented above represent my most significant programming and computational research contributions to date. Each project required designing complete end-to-end analytical pipelines, implementing custom algorithms, processing large datasets, and producing publication-quality scientific visualizations. They also demonstrate my ability to work independently with multiple programming environments, including **Python**, **MATLAB**, and **ImageJ**, and to apply computational tools to energy systems analysis, techno-economic modeling, signal processing, and image-based diagnostics.

Beyond these major projects, I have completed additional coursework and research assignments involving Python, MATLAB, PVsyst, ENVI-met, and other analytical tools, including data cleaning routines, numerical simulations, thermodynamic modeling, and optimization tasks. Together, these experiences reflect a strong and versatile programming foundation that supports my long-term research direction in computational energy systems, scenario modeling, and climate-transition analysis.