PyH2A Optimization Module Report

Prepared by
Abdul Hanan Nawaz
Frankfurt University of Applied Sciences

Take Home Assignment for Student Assistant LCA Role
Submitted to
Water Splitting Group

September 2025

Image courtesy of Dr. Jacob Schneidewind.

Table of Content

1. Introduction

The objective of this task is to develop a new analysis module that enables general-purpose optimization within the PyH2A framework. The module is designed to adjust specified model parameters within their defined bounds in order to minimize the Levelized Cost of Hydrogen (LCOH). By providing a flexible optimization workflow, the module supports systematic exploration of techno-economic parameters and helps identify the most cost-effective plant configurations.

2. Specific Test

To validate the new optimization module, a test case was performed using the input file:

pyH2A/data/PV_E/Base/PV_E_Base.md

This file describes a photovoltaic (PV) + electrolysis hydrogen production plant. In this test, the parameter selected for optimization was the nominal power of the PV system:

- Parameter: Photovoltaic > Nominal Power (kW) > Value
- Optimization Objective: Minimize LCOH
- Constraint: All other parameters remain fixed at baseline values

Rationale
There exists an optimal ratio between PV nominal power and electrolyzer nominal power:

- If the PV system is too small, it will not provide sufficient power to the electrolyzer, reducing hydrogen production.
- If the PV system is too large, it will generate more electricity than the electrolyzer can consume, leading to inefficiencies.

Furthermore, PV electricity production is inherently variable across the year due to seasonal and daily variations in solar availability. For this reason, the optimal PV-to-electrolyzer ratio deviates from a simple 1:1 relationship.

This optimization therefore aims to determine the PV capacity that minimizes the overall cost of hydrogen production under realistic solar generation conditions.

3. Specifications

The new optimization module was implemented following the established structure of **pyH2A analysis modules**. The key design specifications were:

- **Module structure:**
  The optimization module follows the same structural conventions as existing pyH2A analysis modules to ensure compatibility and maintainability.
- **Input definition:**
  A new **parameter table** was added to the input file to define the variables subject to optimization and their respective bounds. This table is conceptually similar to the parameter table used by Monte_Carlo_Analysis.py.
- **Optimization algorithm:**
  A **global optimization method** was required. The chosen approach

was differential_evolution from the SciPy library, which iteratively updates parameter values and evaluates the impact on the **discounted cash flow (DCF) analysis**.

- **Output requirements:**
  The module should clearly print the optimized parameter values and the corresponding minimized **Levelized Cost of Hydrogen (LCOH)**, providing a transparent summary of the optimization results.

4. My Approach

To implement the optimization analysis, I developed a new class Optimization_Analysis that integrates seamlessly into the pyH2A framework. The main idea is to allow flexible optimization of any model parameter defined in the input file by repeatedly running the discounted cash flow (DCF) calculation until the Levelized Cost of Hydrogen (LCOH) is minimized. The updated implementation improves robustness, speed, and transparency of results.

4.1 Input Processing

- The input file is first converted into a fully processed Python dictionary using convert_input_to_dictionary().
- A baseline case is generated with Discounted_Cash_Flow() to serve as a reference for comparison.
- The optimization parameters are extracted from the new Optimization_Analysis section in the input file. For each parameter, the following are validated and stored:
  - Path in the input dictionary (e.g., ['Photovoltaic', 'Nominal Power (kW)', 'Value']).
  - Type
  - Bounds (lower and upper limits defined in the input file, with validation that lower < upper).

4.2 Objective Function

The objective function defines the quantity to be minimized, the hydrogen production cost (h2_cost).

- Each trial solution vector x suggested by the optimizer is first checked against a **cache** (rounded to 6 decimal places by default). This avoids redundant DCF runs for repeated parameter sets, greatly improving performance.
- If not cached:
  - A deep copy of the input dictionary is created (to avoid overwriting baseline values).
  - Each parameter in x is inserted into the copied dictionary using set_by_path().
  - A new Discounted_Cash_Flow analysis is performed.
  - Failures during calculation are caught, and invalid solutions are penalized with inf.
- The hydrogen cost is then returned and stored in the cache.
- Optional verbose logging shows parameter values and trial costs for debugging and transparency.

4.3 Optimization Algorithm

The optimization is carried out using the Differential Evolution algorithm from SciPy:

- This global optimization method is well-suited for non-linear, multi-dimensional problems like hydrogen techno-economic analysis.
- Enhancements in this version include:
    - **Parallelization** using the workers argument (default: all cores).
    - Explicitly setting updating="deferred" to match parallel execution mode.
    - Caching of solutions to avoid recomputation.
- Key parameters include:
    - maxiter → maximum number of generations.
    - popsize → population size per generation.
    - seed → optional random seed for reproducibility.
    - polish=True → performs a final local optimization after convergence.

4.4 Results Output

Once optimization is complete, the following are reported:

- Optimal H_2 cost in $/kg.
- Optimized values of each parameter, with their names reconstructed from the input path for clarity (e.g., Photovoltaic > Nominal Power (kW) > Value).
- Full results are returned as a dictionary including the SciPy result object for reproducibility and further analysis.

This structured and robust approach ensures both **computational efficiency** (via caching and parallelization) and **transparency** (clear reporting of parameter impacts).

5. Input File Modifications

To enable the new optimization workflow, I extended the existing input file pyH2A/data/PV_E/Base/PV_E_Base.md. All original sections of the file remain unchanged; only two new tables were added at the end: **Optimization_Analysis** and **Optimization_Analysis_Methods**.

5.1 Optimization_Analysis

This table specifies which parameters should be optimized, along with their types and bounds.

Parameter	Name	Type	Lower	Upper
Photovoltaic > Nominal Power (kW) > Value	PV Size	value	1	3

- **Parameter:** Path to the input variable being optimized (here, the nominal PV system power).
- **Name:** User-friendly label (*PV Size*).
- **Type:** Optimization mode, set to value (direct numerical optimization).
- **Lower / Upper:** Bounds for the parameter search. This defines the optimization range for the PV-to-electrolyzer power ratio.

5.2 Optimization_Analysis_Methods

This table defines how the optimization procedure is executed.

Name	Method Name	Arguments
Run Optimization	run_optimization	{"maxiter":50, "popsize":30}

- **Name:** Identifier of the optimization routine.
- **Method Name:** Calls the run_optimization method implemented in the new analysis module.
- **Arguments:** Key settings for the differential evolution algorithm:
  - maxiter=50 → maximum number of generations.
  - popsize=30 → population size for each generation.

5.3 Summary

By adding these two tables, the new module automatically detects the parameter to optimize and applies the specified optimization method. The result is a **flexible and generalizable input structure**, consistent with the design of other pyH2A analysis modules (e.g., Monte Carlo analysis), while enabling **global optimization of techno-economic parameters**.

6. Results and Discussion

The optimization run was performed using the modified input file (PV_E_Base.md) with the new **Optimization_Analysis** module enabled. The optimization targeted the PV system nominal power as the free parameter, while all other parameters remained at baseline values.

Optimization Results:

- **Optimal Levelized Cost of Hydrogen (LCOH):** 3.5778 $/kg
- **Parameter 'Photovoltaic > Nominal Power (kW) > Value' optimized to:** 1.497
- **Baseline LCOH (before optimization):** 3. 3.5777931317137512 $/kg

Discussion:

- The optimization confirmed that the PV-to-electrolyzer ratio requires slight adjustment from the nominal 1.5 baseline to an optimal value of ~1.497.
- The cost improvement compared to the baseline is marginal (3.577793 → 3.5778 $/kg), which indicates that the base configuration was already close to optimal.
- This result validates that the optimization module is functioning correctly: it successfully identifies the cost-minimizing configuration within the defined parameter bounds.
- While the cost reduction was small in this single-variable case, the framework allows for multi-parameter optimization. Such extensions are expected to yield more significant cost reductions and insights.

7. Conclusion

In this assignment, a new **Optimization_Analysis** module was developed and integrated into the PyH2A framework. The module enables systematic optimization of techno-economic parameters to minimize hydrogen production costs.

- The module was tested on a PV + electrolysis hydrogen plant, optimizing PV nominal power.

- The optimal configuration slightly adjusted PV size (1.497 vs. 1.5), resulting in a minimal cost improvement.
- The test demonstrates the robustness and correctness of the implementation, while also highlighting that meaningful cost reductions may require multi-variable optimization or wider parameter bounds.

Overall, the optimization module provides a flexible and general purpose tool for techno-economic studies of hydrogen production systems, and can be extended to explore more complex optimization problems in future work.

7. Reference

- Schneidewind, J. (2025). *pyH2A: Python framework for techno-economic analysis of hydrogen production*. GitHub repository. Retrieved from https://github.com/jschneidewind/pyH2A