

LABOTHAPPY: DEVELOPMENT OF AN OPEN-SOURCE PYTHON LIBRARY FOR THERMODYNAMIC SYSTEM MODELLING AND SIMULATION

Elise Neven^{1*}, Basile Chaudoir^{1*}, Vincent Lemort¹

¹ Thermodynamics Laboratory, University of Liege, Liege, Belgium

* Corresponding Author: elise.neven@uliege.be

Corresponding Author: basile.chaudoir@uliege.be

Abstract. Modelling and simulation are essential in the design, optimization, and control of thermodynamic systems, yet developing reliable models can be time-consuming, and recreating similar models across different platforms or code versions is redundant and inefficient. To address this challenge, researchers at the University of Liege have developed *LaboThapPy*, an open-source Python library designed to gather a wide range of thermodynamic models accumulated over years of research. By using Python's Object-Oriented Programming (OOP) capabilities, *LaboThapPy* offers a modular and flexible toolkit for simulating thermodynamic systems. The library can support a variety of applications, such as, Organic Rankine Cycle (ORC) systems, Brayton cycle systems, and Heat Pump (HP) systems, and can simulate part-load behaviour. This paper presents an overview of *LaboThapPy*'s architecture, capabilities, and applications, demonstrating its potential as a resource for thermodynamic research and engineering.

Keywords. Python library, Thermodynamic system modelling, Open-source software, Object-Oriented Programming

Nomenclature

$\dot{m}$	Mass flow rate (kg/s)
h	Specific enthalpy (J/kg)
$\dot{Q}$	Transferred heat power (W)
$\dot{W}$	Transferred work power (W)

1 Introduction

Over years of research, numerous thermodynamic models have been developed to simulate individual components and entire systems, allowing the design, optimization, and control of thermodynamic systems. At the Thermodynamics Laboratory of the University of Liege, researchers observed a recurring issue: identical or similar models were being recreated repeatedly over different software or codes, while others were lost as researchers departed, resulting in lost time and efficiency for new team members. This challenge underscored the need for a standardized, accessible repository that would gather these models, fostering continuity and productivity across research projects.

The *LaboThapPy* library emerged as a solution to this problem, establishing a common framework for modelling thermodynamic components and enabling

users to assemble these components into larger systems. Designed as a flexible and modular toolkit, *LaboThapPy* offers researchers a platform to build and simulate complex thermodynamic systems. Python was chosen as the development language due to its Object-Oriented Programming (OOP) capabilities, which enhance modularity and extensibility, its wide adoption in scientific research and its free availability. Furthermore, *LaboThapPy* is available as an open-source library on GitHub, encouraging collaboration and open-science practices that are becoming increasingly important in energy system modelling [1].

Although a range of thermodynamic modelling libraries exist (e.g., ASPEN [2], Thermocycle [3], TESPy [4], and oemof [5]), *LaboThapPy* distinguishes itself with its focus on detailed component-level modelling, an aspect often less emphasized by other platforms. This paper presents an overview of *LaboThapPy*, its architecture, capabilities, and contributions, demonstrating its potential as a resource for the thermodynamics research and engineering community.

2 Scope and content

The *LaboThapPy* library currently encompasses a range of steady-state models and sizing models for thermodynamic components, with plans to

incorporate dynamic models in future versions. Its component models span from basic models (for example a constant isentropic efficiency model for compressors), to more complex models (for example semi-empirical models for compressors), offering flexibility to accommodate various levels of model complexity depending on the user's needs.

In addition to core component models, *LaboThapPy* provides tools that extend the modelling process, including functions for generating pressure-enthalpy (p-h) diagrams, model calibration codes, and post-processing utilities for data analysis.

With respect to thermodynamic system models, *LaboThapPy* includes basic templates for standard cycles such as simple Organic Rankine Cycle (ORC) systems and Heat Pump (HP) systems, along with solvers for these straightforward cycle simulations. For more complex configurations, the library offers a flexible structural framework, enabling users to assemble component models into larger systems. However, cycle-specific solvers for advanced configurations are currently left to the user to implement, as the primary focus of *LaboThapPy* remains on component-level model development rather than system-level cycle-solving.

3 Architecture

LaboThapPy is built using an object-oriented programming approach, resulting in a highly modular and adaptable codebase. This modularity provides users with the flexibility to configure various thermodynamic cycles and to select specific model types for each component based on the requirements of their simulation. The architecture of *LaboThapPy* is inspired from established thermodynamic simulation libraries (e.g., Thermocycle [2]).

At the core of *LaboThapPy* are three primary modules:

- **Connector:** Allows the linking of components through energy transfers that occur between them.
- **Components:** Serves as the fundamental building blocks, representing individual parts of a system, such as compressors, turbines, and heat exchangers.
- **Machine:** Acts as a container for each simulation, representing entire systems and organizing the assembly of components and connections into cohesive models.

3.1 Connectors

Connectors are the elements that link different components in the system, representing the transfer of energy between them. Energy flows are represented by three types of connectors:

- **Mass Connector** carries energy through mass flow, defined as $\dot{m} \cdot h$ (mass flow rate multiplied by specific enthalpy). It computes thermodynamic properties at each point using two thermodynamic properties as inputs. The library relies on CoolProp [6] to obtain these thermodynamic properties. Currently, fluid mixtures are not implemented
- **Heat Connector** represents the transfer of thermal energy, defined as $\dot{Q}$ (rate of heat flow), to account for heat energy transfer between components or with the ambient.
- **Work Connector** represents the transfer of mechanical energy, defined as $\dot{W}$ (rate of work), allowing the modelling of power-related interaction within the system.

3.2 Components

Component models are the fundamental building blocks for simulating thermodynamic systems. Representing individual components under steady-state conditions, they can be connected using connectors to form complex system models. Each component model is implemented as a class defined by specific parameters, inputs, outputs and connectors. All component classes inherit from a parent class, *'BaseComponent'*, which provides shared methods and ensures uniformity across models. This structure allows coders to focus only on the specifics of each component model, streamlining collaboration and model development.

Components can be defined in two main ways:

1. **Connectors:** Inputs are set via connectors, enabling components to directly exchange energy with other components within a system. This approach is ideal for full-system simulations.
2. **Direct Inputs/Outputs:** Inputs and outputs are set directly within each component, useful for standalone use of component models.

Currently, the library supports only steady-state models, while future expansions may include dynamic modelling capabilities.

3.3 Machines

The *Machine* module acts as a container for each simulation, organizing the assembly of components into functional circuits and systems. Individual components can be linked to form circuits, which may be either open or closed loops, and each circuit is characterized by a single working fluid. Multiple circuits can then be combined to construct entire systems.

Circuits within a machine can also connect to *sources* and *sinks* for mass, heat, or work. Sources are defined as hypothetical reservoirs that can supply an unlimited amount of mass, heat, or work, while sinks represent theoretical outlets capable of absorbing these flows. Users can also define *'sub-machines'*, which group components to function as single units within larger systems, enhancing modularity.

Similar to the *'BaseComponent'* class, which standardizes component-level models, the *'Circuit'* class serves as a parent class containing essential methods shared across different circuit types, simplifying circuit creation.

The library uses a causal-recursive approach to solve systems, prioritizing robustness and reliability, particularly for complex thermodynamic models. While this method may be slower and less flexible, it simplifies debugging. Conversely, an acausal approach offers more readable models but makes debugging harder and less robust [7].

4 Method of Implementation

The modelling process for simulating a specific thermodynamic system involves several structured steps, illustrated here using the example system in Figure 1. This system comprises an Organic Rankine Cycle (ORC) with R1233Zd(E) as the working fluid, as well as a hot and a cold water loop. The latter features an air mass source and sink. This example will be soon available in the library.

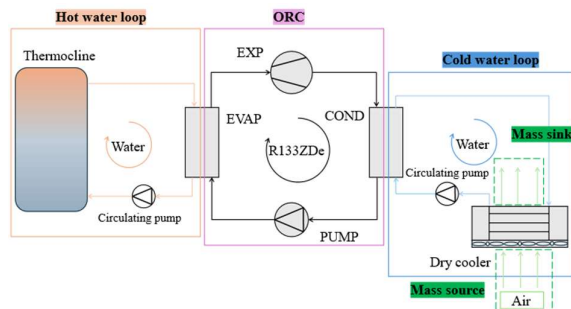


Figure 1: Example of a thermodynamic system.

1. System Initialization: The simulation begins by creating an empty machine class, which serves as a container for all system components and initial parameters. Each main loop in the system, such as the ORC, hot water, and cold water loops, is represented by a corresponding circuit object to help organize the components within each section.

2. Component Instantiation: Within each circuit, individual components are instantiated with their specific parameters. For instance, the ORC circuit includes a pump, heat exchangers, and an expander; the hot water loop includes a pump and a thermal storage; and the cold water loop contains a dry cooler and a pump.

3. Component Connection: Components within each circuit are connected through connectors, which facilitate the flow of energy between them. In this example, mass connectors link components within the ORC, hot water, and cold water loops, enabling continuous energy and mass flow through each part of the system.

4. Initialization of Guesses: After the circuits and components are configured, initial guesses for the solution process are provided either by the user or through a pre-conditioner. These initial guesses serve as a foundation for the subsequent recursive solution process.

5. Recursive Solution Process: The solution process involves traversing each circuit recursively. As the process advances, each component model is solved based on its inputs, parameters, and initial guesses. This recursive process updates and refines the solution iteratively until the system reaches convergence on a stable and accurate solution.

5 Documentation

Comprehensive documentation is essential for an open-source project like *LaboThapPy*, which aims to support collaboration among multiple contributors and users. To facilitate ease of use and ensure consistency, a standardized nomenclature is implemented throughout the code. The documentation is organized in four different levels:

1. In-Code Comments: Comments to clarify complex or less intuitive parts of the code.

2. Docstrings: Each model and function is documented with docstrings that describe its purpose, underlying assumptions, and key parameters.

3. Usage Examples: Practical usage examples demonstrate how to apply models within the library.

4. Related Articles/Reports: For certain models, detailed articles and reports are available, offering in-depth explanations.

Additionally, a *Read the Docs* site [8] has been implemented to provide centralized, accessible documentation. This online resource lists all component models currently available in the library, with descriptions and usage guidelines. The LaboThapPy GitHub repository [9] hosts the code, facilitates collaboration, and includes bug tracking, creating a transparent and collaborative platform for development.

6 Work in Progress

LaboThapPy is an evolving project with active development aimed at expanding its model library and capabilities. Current goals include adding new component models, system applications, test units, charge-sensitive models, and dynamic models. An interesting future feature could be integrating existing frameworks for solving model equations, which could enhance the library's ability to handle dynamic simulations more efficiently.

7 Conclusion

LaboThapPy is a flexible and modular open-source Python library for thermodynamic modelling, developed from models based on years of research at the University of Liege. By offering standardized, object-oriented component models, the library reduces coding redundancy and promotes collaboration across projects. With its structured framework of connectors, components, and machine models, *LaboThapPy* supports the simulation of various thermodynamic systems, including Organic Rankine Cycles (ORC), heat pumps, and other thermodynamic system applications. To ensure accessibility, *LaboThapPy* features layered documentation and is hosted on GitHub, where users can contribute to its development, report issues, and suggest enhancements. This open-source approach not only facilitates research and engineering applications but also encourages ongoing improvement and innovation within the thermodynamic modelling community.

Acknowledgement

Many thanks to all contributors.

References

- [1] S. Pfenninger, J. DeCarolus, L. Hirth, S. Quoilin, and I. Staffell, 'The importance of open data and software: Is energy research lagging behind?', *Energy Policy*, vol. 101, pp. 211–215, Feb. 2017, doi: [10.1016/j.enpol.2016.11.046](https://doi.org/10.1016/j.enpol.2016.11.046).
- [2] Aspen Technology, Inc. *ASPEN Plus*. Bedford, MA, USA: Aspen Technology, 2023.
- [3] S. Quoilin, A. Desideri, J. Wronski, I. Bell, and V. Lemort, 'ThermoCycle: A Modelica library for the simulation of thermodynamic systems', presented at the the 10th International Modelica Conference, March 10–12, 2014, Lund, Sweden, Mar. 2014, pp. 683–692. doi: [10.3384/ecp14096683](https://doi.org/10.3384/ecp14096683).
- [4] F. Witte and I. Tuschy, 'TESPy: Thermal Engineering Systems in Python', *JOSS*, vol. 5, no. 49, p. 2178, May 2020, doi: [10.21105/joss.02178](https://doi.org/10.21105/joss.02178).
- [5] S. Hilpert, C. Kaldemeyer, U. Krien, S. Günther, C. Wingenbach, and G. Plessmann, 'The Open Energy Modelling Framework (oemof) - A new approach to facilitate open science in energy system modelling', *Energy Strategy Reviews*, vol. 22, pp. 16–25, Nov. 2018, doi: [10.1016/j.esr.2018.07.001](https://doi.org/10.1016/j.esr.2018.07.001).
- [6] I. H. Bell, J. Wronski, S. Quoilin, and V. Lemort, 'Pure and Pseudo-pure Fluid Thermophysical Property Evaluation and the Open-Source Thermophysical Property Library CoolProp', *Ind. Eng. Chem. Res.*, vol. 53, no. 6, pp. 2498–2508, Feb. 2014, doi: [10.1021/ie4033999](https://doi.org/10.1021/ie4033999).
- [7] G. Schweiger, H. Nilsson, J. Schoeggl, W. Birk, and A. Posch, 'Modeling and simulation of large-scale Systems: a systematic comparison of modeling paradigms', Sep. 01, 2019, *arXiv*: arXiv:1909.00484. doi: [10.48550/arXiv.1909.00484](https://doi.org/10.48550/arXiv.1909.00484).
- [8] "LaboThapPy documentation" [Online]. Available: labothappy.readthedocs.io (2024)
- [9] "LaboThapPy repository" [Online]. Available: <https://github.com/PyLaboThap/LaboThapPy> (2024)