A Comparison of Numerical Methods for Steady-State Thermodynamic System Simulation

Basile Chaudoir[a] , Elise Neven[a], Sylvain Quoilin[a] and Vincent Lemort[a]

[a] *University of Liège, Thermodynamics Laboratory, Liège, Belgium, Basile.Chaudoir@uliege.be*

Abstract:

Simulation at system level of thermodynamic systems such as heat pumps and power-generation cycles is a cornerstone for studying their design, part-load and dynamic performances, and control strategies. Several choices impact the solving of such complex systems such as the modelling hierarchy (flat or library frames), the circuit solving strategy (sequential or equation-based), and the numerical solving method. Research slowed down on these topics since the 2000s with the emergence of proprietary solving environments, limiting transparency, reproducibility and community-driven development.

This work introduces and compares two open-source thermodynamic system solving environments: CoolSolve, an equation-based parser for the EES language with a flat architecture, and LaboThApPy, a sequential library-integrated solver built on an object-oriented programming Python framework. Both were benchmarked on two thermodynamic simulation cases: a recuperated high-temperature heat pump and a preheated-recuperated ORC, under varying design parameters to assess convergence robustness and speed. Additionally, fixed-point iteration and Newton-based methods were compared within the LaboThApPy frame to provide practical guidelines for system simulation.

LaboThApPy fixed-point iteration tools achieved 100% of convergence rate with the fastest average convergence times (15-41ms). CoolSolve was competitive in speed (40ms) but converged for only 51% of the ORC case test matrix. Finally, the LaboThApPy Newton-based tools showed higher convergence times (37-64ms) with above 86% convergence rate, making fixed-point-iteration the recommended approach for steady-state thermodynamic system simulation of the considered complexity.

Keywords:

ORC; Heat Pump; Steady-State Solver; Convergence Robustness, Fixed-Point Iteration.

1. Introduction

Simulation of thermodynamic systems emerged in the second half of the 20th century alongside developments in digital computing [1]. It rapidly accelerated the engineering progress by diminishing the need for long and expensive experimental campaigns. Numerical simulation of such complex systems involves several methodological choices. The first choice concerns the modelling architecture. Some tools establish a hierarchy between components and systems, leading to a component-based (or library) modelling approach. This paradigm is commonly implemented in object-oriented modelling environments such as Modelica and Aspen Plus, due to their intrinsic modularity, component reusability, and ease of formalization. Other environments, such as EES, as well as general purpose programming languages including Python and MATLAB/Simulink are primarily code-based. They offer greater flexibility for developing more transparent and customizable simulations, though this comes at the expense of user convenience and modularity [2]. However, the use of formalized code could enable the development of a component-based library such as TESPy [3].

Another key methodological aspect is the circuit solving strategy. Tools such as EES and Modelica rely on equation-based frameworks, in which systems are treated as sets of algebraic and differential equations. This formulation allows one to ignore component solving order by automatically determining variable causality, providing a more general solving framework. In contrast, other systems such as Aspen Plus use a sequential approach that defines a solving order. Sequential methods are less flexible, as the variable causality is fixed by the component models. However, fixing explicit variable causality reduces the number of unknowns. Winkler

et al. [4] showed that this reduction in problem size enhances numerical robustness and convergence speed across various initial guesses for steady-state off-design simulation.

Finally, the numerical solving approach is often overlooked in thermal system modeling since the 2000s, yet this choice can be critical for convergence robustness. Winkler et al. [4] also compared a Newton-Raphson solving approach with a Broyden variation and a hybrid solver. Their results showed that, across a range of initial guesses, the hybrid and Broyden solving approaches outperformed the Newton solving method in convergence robustness. Despite the importance of solver selection, systematic comparisons of numerical methods for thermodynamic cycle simulation remain scarce in the recent literature, with Winkler et al. [4] being one of the few dedicated investigations.

Table 1 provides an overview of popular thermodynamic system modeling tools. Most are proprietary software, except for TESPy being open-source. The prevalence of proprietary software limits customization of model components and solving approaches, while reducing transparency and reproducibility. It also makes controlled comparisons of system solving approaches challenging.

Table. 1. *Overview of simulation tools for thermodynamic systems.*

Name	Model Architecture	Circuit Solving	Focus Regime	Numerical Solving
TRNSYS [5]	Flat	Sequential	Transient	Runge-Kutta
Modelica [6]	Hierarchical	Equation-Based	Transient	DASSL/BDF
EES [7]	Flat	Equation-Based	Steady-State	Newton-Raphson
Simulink [8]	Hierarchical	Hybrid	Transient	Ode45
Aspen [9]	Hierarchical	Sequential	Both	Wegstein
TESPy [4]	Hierarchical	Equation-Based	Steady-State	Newton-Raphson

To the authors' knowledge, no open-source tools allow a direct and controlled comparison of sequential vs. equation-based solving under various numerical methods. In this context, the authors introduce two solving environments:

- LaboThApPy, an Object-Oriented Programming library in Python for the design and simulation of thermodynamic systems using a sequential solving approach. It employs component models with different levels of complexity.
- CoolSolve, an equation solver coded in C++ that parses and solves systems of equations written in the EES (Engineering Equation Solver) language. In contrast to LaboThApPy, it solves circuits using an equation-based approach.

The goal of this paper is to compare both environments in terms of convergence speed and robustness. This comparison is performed by varying design parameters with similar initial guess values for two controlled test cases: a recuperated high temperature heat-pump and a preheated-recuperated ORC. To ensure a fair comparison, the models are formulated with identical equations and boundary conditions, and both tools rely on CoolProp [10] for thermodynamic property computation. Several numerical methods are tested for each environment, with the aim of providing practical guidance on solver and architecture selection for steady-state thermodynamic system simulation.

2. Steady-State Thermodynamic Cycle Solving

This section presents a mathematical representation of a thermodynamic cycle solving problem as well as the solving approaches considered in this study.

2.1. Problem Definition

Solving a closed thermodynamic cycle is a highly implicit problem due to its cyclic configuration. Its complexity lies in the interactions between all thermodynamic states $\underline{\boldsymbol{x}}$ (each defined by two independent thermodynamic properties) and the component modelling functions $\mathbf{F}(\underline{\boldsymbol{x}})$ linking them. The problem is usually solved using iterative approaches and the convergence success can be highly sensitive to the initial property guesses. A

converged solution is obtained when energy and mass conservation are satisfied across all system components, or when user-defined residual variables fall below a prescribed tolerance.

For steady-state modeling, appropriate solving approaches are directly linked to the component model detail level. The more the component models become detailed, the fewer state assumptions need to be provided. For system design, a high number of state assumptions are usually imposed by the models or the user directly, whereas for system off-design simulation, models become realistic by fixing fewer variables [11]. There are two common ways to simulate off-design steady-state operation: imposed subcooling and charge-sensitive modelling. The first closes the system of equation by iterating on the low pressure until a given subcooling is reached. The latter fixes the total working fluid mass in the system and iterates on the low pressure to reconcile the imposed total mass of working fluid and the sum of working fluid mass recomputed in the components and piping. Even though the introduced tools can perform both design and off-design simulations, only design simulation is considered in this work.

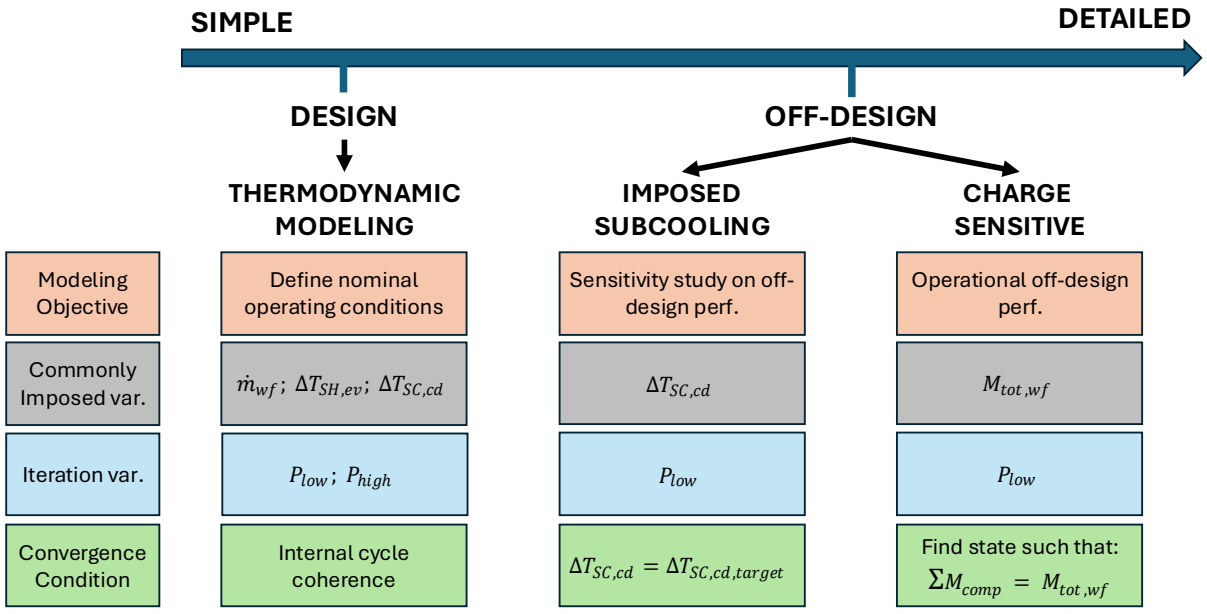

Figure. 1. *Comparative summary of steady-state thermodynamic system modelling approaches.*

2.2. Numerical Approaches

Thermodynamic systems have highly non-linear behaviors (for example, the evolution of temperatures across a heat exchanger is logarithmic). Therefore, even simple thermal system modelling requires non-linear solving approaches. Two broad families of numerical methods are commonly used in thermodynamic system simulation: Fixed-Point Iteration (FPI) and Newton-based methods.

Closed sequential thermodynamic systems can be formulated as an FPI problem. For a simple example: a four-component cycle, the expression for a given $\underline{\boldsymbol{x}}$ point at the n-iteration can be written as:

$$\underline{\boldsymbol{x_n}} = (\mathbf{F_4} \circ \mathbf{F_3} \circ \mathbf{F_2} \circ \mathbf{F_1})(\underline{\boldsymbol{x_{n-1}}}) = \mathbf{G}(\underline{\boldsymbol{x_{n-1}}})$$

The system is solved when the state variations become lower than a user-defined tolerance $\left\|\underline{\boldsymbol{x_n}} - \underline{\boldsymbol{x_{n-1}}}\right\| \leq \varepsilon$.

Convergence is guaranteed as long as the iteration function $\mathbf{G}$ contracts the solution space: this requires that the Jacobian of $\mathbf{G}$ evaluated at $\underline{\boldsymbol{x_n}}$ has all eigenvalues of modulus strictly less than 1 $(all\ \lambda\ of\ \boldsymbol{\nabla G}(\underline{\boldsymbol{x_n}}) < 1)$.

This convergence criterion induces linear convergence.

Alternatively, Newton-based methods require the formulation of a residual function $\mathbf{R}$, and then seek its root $\mathbf{R}(\underline{\boldsymbol{x}}) = \mathbf{0}$. They iterate as follows, where $\mathbf{J}$ is the Jacobian of $\boldsymbol{R}$:

$$\underline{\boldsymbol{x_n}} = \underline{\boldsymbol{x_{n-1}}} - \boldsymbol{J}(\underline{\boldsymbol{x_{n-1}}})^{-1} \cdot \boldsymbol{R}(\underline{\boldsymbol{x_{n-1}}})$$

The system is solved when the residual function value falls below a user-defined tolerance $\left\|\underline{\boldsymbol{R_n}}\right\| \leq \varepsilon$. This method achieves $\lambda \, of \, \boldsymbol{J}\left(\underline{\boldsymbol{x_n}}\right) = 0$ by linearizing $\boldsymbol{R}$ around the current estimate, which guarantees quadratic convergence in the neighborhood of the solution. The conditions for convergence are therefore that the residual function $\boldsymbol{R}$ is smooth and its Jacobian $\boldsymbol{J}$ is non-singular in that region.

From a mathematical standpoint, Newton-based methods offer stronger convergence guarantees and faster convergence rates than FPI when initial guesses are close to the solution. However, modifications to these algorithms can alter this postulate: the Wegstein method can increase FPI convergence order, while confidence intervals for Newton methods can increase its robustness to initial guesses far from the solution. Furthermore, it remains an open question whether mathematical convergence criteria are the most restricting factor in thermodynamic system simulation.

2.3. When Numerical Methods become Physics Bound

The satisfaction of a mathematical convergence criterion is necessary but not sufficient for a thermodynamic system simulation. Physical admissibility, practical component limits, and operational behavior further constrain the solving procedure. Figure 2 illustrates the allowable T-s state space for ORC components. Several problematic regions have been colored to illustrate physical operating constraints of the components:

- **Solid-state region (red)**: No fluid flow can be modelled. This region must be avoided.
- **Unsuitable phase regions (orange)**: These regions shall be avoided in steady-state operation as they could damage the components. They correspond to cavitation for a pump and to appearance of droplets in an expander.
- **Unsuitable semi-phase region (yellow)**: Conditions above the supercritical state that could be problematic for the component operation due to the semi-phase properties.
- **Unsuitable temperature region (grey)**: Represents a temperature domain that would go against the heat exchanger use.

This list of restrictions is not exhaustive, for example, manufacturer limits (on operating pressure and temperature for instance) and secondary fluid temperatures could also restrict **the allowable region that remains in white**. These limits are problem dependent.

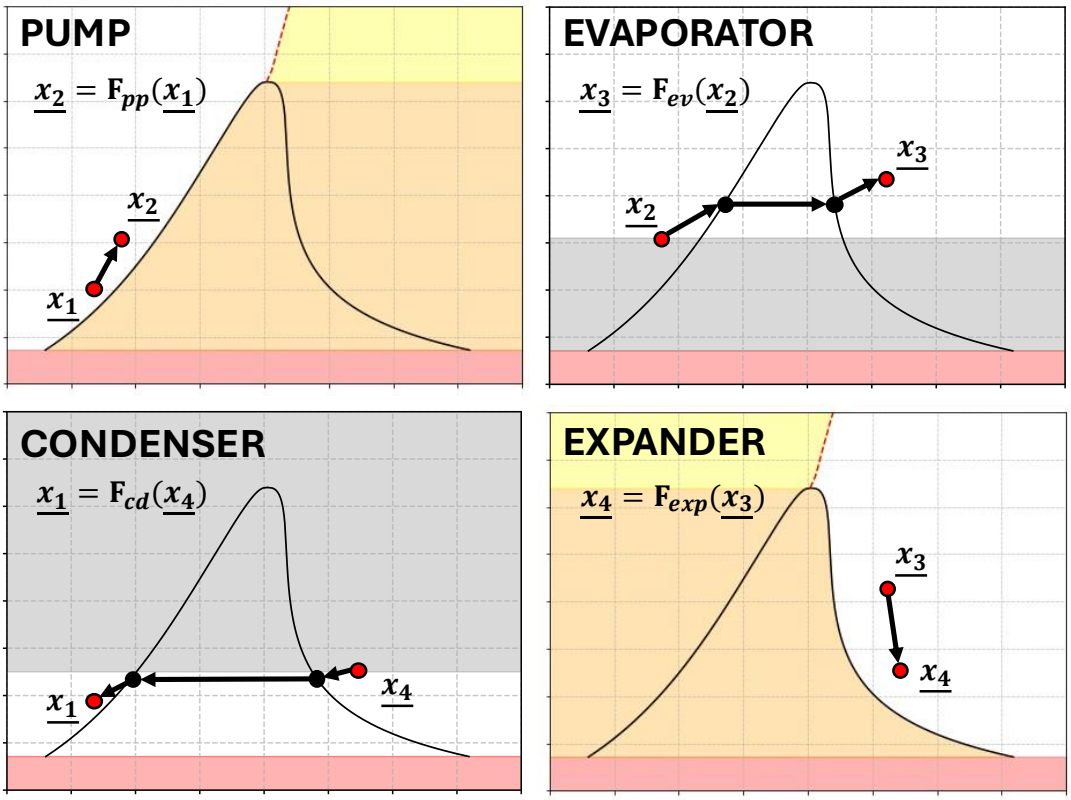

Figure. 2. *Examples of physical and operational constraints on ORC component simulation represented in a T-s space.*

These constraints induce a highly physics-bound problem, and errors at many levels could cause simulation failure:

- **At the cycle level,** an ill-posed problem could lead to an unstable operating point, making the simulation diverge.
- **At the component level,** unsuitable phase conditions or the breaking of the second law of thermodynamics could prevent the models from finding a physical solution.
- **At the correlation level,** the flow conditions could exceed the correlation validity range.
- **At the property database level,** the inputs submitted by the models could lead to incoherent thermodynamic states (e.g., seeking saturation temperature above the critical pressure).
- **At the cycle solver level,** discontinuities in the modelling could occur during phase change or at the critical point. If these discontinuities happen near the solution, some numerical method could fail to converge.

For these reasons, a solver should not be preferred solely based on its mathematical convergence criterion. The best solver itself may be case-dependent, depending on the system architecture but also on the problem definition.

2.4. LaboThApPy Solving Approaches

LaboThApPy is a thermodynamic component library implemented in Python, using Object-Oriented Programming. It was first developed as a thermodynamic simulation tool to allow the sharing and modular interconnection of complex component models developed in the Thermodynamics Laboratory of the University of Liege. In its current form, the library includes components and sequential solving methods for system design and off-design simulation (in steady-state).

The problem formulation and solving are performed as follows:

1. Instantiate the cycle and the chosen component models.
2. Link the components to define the system architecture and connect fluid, work, and heat sources.
3. Select iteration variables and set their initial guess values along with fixed property values.
4. Select residual variables and set their objective (can be either a user-defined target value or the closing of the cycle).
5. Perform a warm-up solving of the cycle. Determine the component solving order by recursively traversing the system's graph and by solving calculable components (i.e., the components having their required inputs set as guesses or fed by another component's results).
6. Iterate on the guess variables to find the residual variable roots. The solving of the components done using the order determined in step 5.
7. Check energy conservation and extract results.

For the step 6, several numerical methods have been implemented. They can be categorized into the two main solving approaches presented in Section 2.1. Table 2 shows the main difference in implementation between the two families of methods.

Table. 2. *Main numerical methods implemented in LaboThApPy.*

Family	FPI	Newton
Guess update philosophy	Based on result history: **Direct model dependence**	Based on Jacobian computation: **Indirect model dependence**
Natural Residual Type	Closing Condition	Target value satisfaction
Implemented Methods	Successive Substitution Wegstein	Newton Levenberg-Marquardt (lm) Powell Hybrid (fsolve)

From Figure 1 and Table 2, it can be deduced that FPI solving approach seems more fitted to thermodynamic problems as the usual convergence criterion is more aligned with its natural residual type. In contrast, Newton methods seem more adapted to off-design simulation as it implies the satisfaction of a target value ($\Delta T_{sc,cd}$ or $M_{tot,wf}$). That said, both methods have inherent limitations:

- FPI directly depends on component models for propagating information and updating variables. This implies that if components do not modify important values, these might never be updated (for example, an expander model that does not directly modify its exhaust pressure).
- Newton methods externally perturb iteration variables to estimate the Jacobian. If those same variables are internally enforced by a component model, the perturbation is silently overwritten, rendering the Jacobian estimate meaningless and potentially causing divergence (for example, a constant pinch condenser model that re-computes its saturation pressure).

These limitations show that a single solving method cannot directly tackle any sequential cycle solving problem without proper adaptations. Therefore, both solving approaches have been adapted to both residual types:

- For FPI methods, a parallel iteration can be added. An iteration variable is chosen together with a residual variable target. Across FPI iterations, the iteration variable is adjusted using the secant method to converge to the target value. To prevent oscillations from the parallel iterations with the FPI iterations, a user-defined relaxation factor can be added to the secant iterative process. This workaround transforms the FPI method into an FPI-Newton hybrid.
- For Newton methods, a closing condition can be added by tracking relative variation of defined state variables between iterations. Instead of actively driving a target, the Newton solver lets the components update thermodynamic states freely. Convergence is then reached when the variation falls below a threshold. This aligns a Newton method's behavior with that of FPI for that specific residual.

2.5. CoolSolve Solving Approaches

CoolSolve is a parsing algorithm for the EES language coded in C++. It has been developed to provide an open and transparent equation-based framework for thermodynamic cycle modelling. This tool is developed as a web application and is focused on steady-state system design and simulation.

The problem formulation and solving are performed as follows:

1. Equations and procedures are written in the EES language.
2. Initial guesses are optionally provided for unknown variables. Un-guessed variables are initialized to 1.
3. The code is parsed to identify variables and their logical links.
4. A structural analysis decomposes the total equation system into sequentially solvable sub-blocks. The Hopcroft-Karp algorithm determines which variable is solved in each equation, then Tarjan's algorithm forms the sub-blocks by grouping mutually dependent equations.
5. Forward-mode automatic differentiation is performed to compute the Jacobian using analytical derivatives. This eliminates finite difference steps and allows for better Newton solves.
6. Each sub-block is solved explicitly for one-equation blocks or using a Newton-based algorithm for implicit blocks.
7. Results are extracted.

CoolSolve implements a wide range of Newton-based solvers including line search, trust-region dogleg, Levenberg-Marquardt, and homotopy continuation methods. However, for the purpose of this architectural comparison with LaboThApPy, only the baseline Newton method is retained, ensuring that observed differences in convergence behavior are attributable to the fundamental architectural distinction between equation-based and component-based solving rather than to solver-level implementation choices.

3. Test Case Presentation

3.1. System Architecture

The test case studied in this paper is the Carnot Battery studied in [cite Carnot], whose architecture is presented in Figure 3. This system features a high-temperature recuperated heat pump and an ORC for integration in a geothermal power plant. The heat pump heat source is water brine at 113°C and the heat is supplied to a PCM storage with a melting temperature of 141°C. The ORC's working fluid is preheated with a small fraction of the 113°C brine and cools down using cold water at 24°C. Despite the simple architectures of both systems, recuperated cycles act as tractable yet non-trivial benchmarks: the internal heat exchanger introduces a second loop of information propagation between the high- and low-pressure sides of the cycle.

3.2. Component Models

The component models (presented in Table 3) used in this study are thermodynamic. They rely on a limited number of parameters to keep the focus on solver convergence. Among these, the constant temperature pinch heat exchanger model is implicit. It iterates on the saturation pressure to find the root of a residual between the computed and the imposed pinch point temperature difference, using the brent algorithm. If a sequential solver feeds this model with a working fluid inlet temperature that prevents the pinch point satisfaction, then the simulation might stop abruptly.

Therefore, a more robust formulation of this model when used in sequential cycle solving has been implemented:

1. Before solving the brent algorithm, the sign of the equation system residual is evaluated for the low- and high-pressure solution bound (namely the triple point and critical pressure with a 5% margin).
2. If both residuals have the same sign, the brent algorithm cannot find a solution, the inlet working fluid temperature is incrementally adjusted upward for condenser and downward for evaporator.
3. Repeat step 2 until the residuals are of different signs.
4. Solve the equation system. If the inlet temperature is modified, then the cycle solver convergence criteria are not met, and at least another solver iteration is required to restore consistency.

Table. 3. *Thermodynamic component modelling summary.*

Component	Modelling Approach	Parameters	Governing Equation(s)
Sensible HX (IHX)	Constant effectiveness	ε = 0.8	$\dot{Q} = \varepsilon \cdot \dot{Q}_{max}$ $\dot{Q}_{max} = min(\dot{Q}_{max,h}, \dot{Q}_{max,c})$ $\dot{Q}_{max,h} = \dot{m}_h \cdot \left(h_{su,h} - h_h\left(T = T_{su,c}\right)\right)$ $\dot{Q}_{max,c} = \dot{m}_c \cdot \left(h_c\left(T = T_{su,h}\right) - h_{su,c}\right)$
Latent HX (EV, CD, PCM HX)	Modified Constant Temperature Pinch	$\Delta T_{pp,ev} = 3$ $\Delta T_{SH} = 1$ $\Delta T_{pp,cd} = 10$ $\Delta T_{SC} = 3$	$\dot{Q} = \dot{m} \cdot \lvert h_{ex} - h_{su} \rvert$ $= \dot{m}_{sf} \cdot \lvert h_{sf,su} - h_{sf,ex} \rvert$ $T_{ex} = T_{sat} \pm \Delta T_{SH/SC}$ $\Delta T_{pp} = min(\underline{\boldsymbol{T_{HX}}} - \underline{\boldsymbol{T_{sf,HX}}})$
Valve	Isenthalpic	–	$h_{ex} = h_{su}$
Compressor	Constant isentropic efficiency (η_{is})	$\eta_{is} = 0.8$	$\dot{W} = \dot{m} \cdot \frac{(h_{is,ex} - h_{su})}{\eta_{is}}$
Expander	Constant (η_{is})	$\eta_{is} = 0.8$	$\dot{W} = \dot{m} \cdot \left(h_{su} - h_{is,ex}\right) * \eta_{is}$
Pump	Constant (η_{is})	$\eta_{is} = 0.7$	$\dot{W} = \dot{m} \cdot \frac{(h_{is,ex} - h_{su})}{\eta_{is}}$

3.3. Design Point

The system has been simulated on its design point using the parameter values in Table 3. Figure 3 shows the thermodynamic states of the solved system. Almost equivalent results were found for the different solving methods mentioned in this section. This design point will serve as baseline for comparisons in Section 4.

4. Simulation Analyses

4.1. Test Matrix

A fair comparison between a sequential and an equation-based solving approach can be ambiguous due to the difference in the number of guesses to provide for these approaches. Also, the choice of these guesses might provide an advantage for a given solving method compared to another.

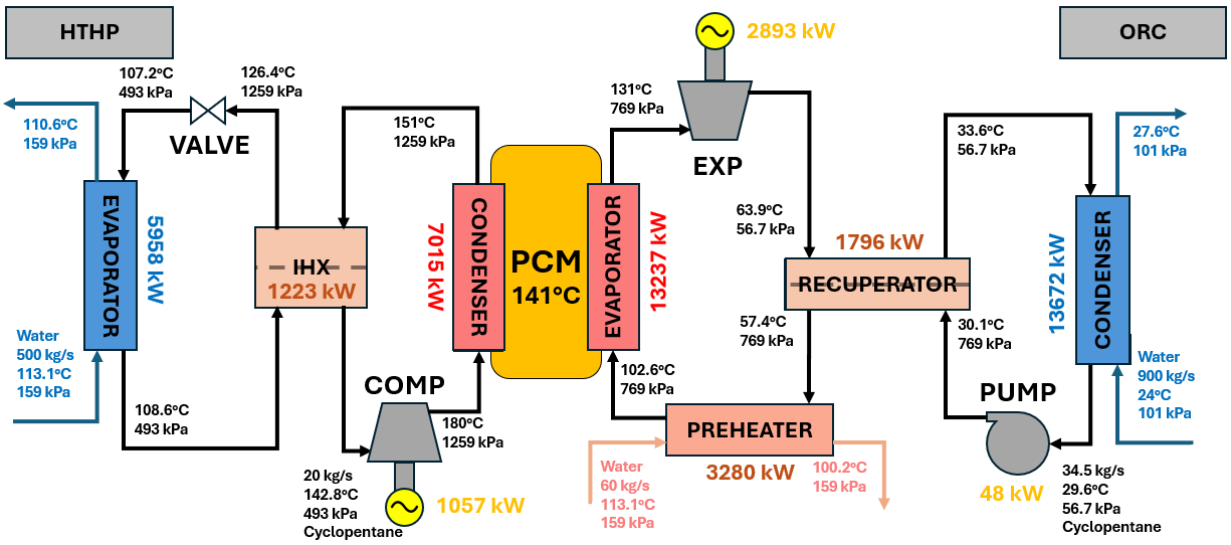

Figure. 3. *Test case architecture with design point solution.*

For these reasons, the baseline initial guesses are set to values from Figure 3 for all variables in the CoolSolve environment, and the same values are set for the iteration variables and guesses in the LaboThApPy approaches. Then, simulations are performed for each combination of constant pinch heat exchanger model parameter according to the ranges in Table 4, creating a 10000-point matrix for each solver and cycle combination.

Table. 4. *Solving options test matrix.*

Variable	$\Delta T_{pp,ev}$	$\Delta T_{sh,ev}$	$\Delta T_{sc,cd}$	$\Delta T_{pp,cd}$
Range	1:1:10	1:1:10	1:1:10	1:1:10

4.2. Heat Pump Simulation

The simulation of the heat pump system converged for all studied numerical methods. Figure 4 shows the average convergence time of each method, with the sequential FPI methods, the Powell method (*fsolve* Python function), and CoolSolve being the fastest.

CoolSolve decomposed the system of 126 equations and 126 variables into 64 blocks, with the largest block containing 63 equations. Despite this decomposition, solving that largest block still introduced most of the numerical complexity and accounted for the majority of the solve time.

LaboThApPy used its sequential approach to exploit logical links between components, requiring only three iteration variables ($P_{high}, P_{low}, and\ h_{su,cp}$), which simplified the numerical solving of the system. However, certain pinch and superheating/subcooling combinations led to thermodynamically inconsistent pinch conditions in constant pinch heat exchanger models. The working fluid inlet temperature had to be modified as detailed in Section 3.2, increasing the required number of iterations.

The fastest method category was the LaboThApPy FPI-based methods. These methods rely on component historical results to update guesses. Since the constant pinch heat exchanger models directly provide the correct pressure levels, these methods have the advantage of declaring convergence as soon as the models converged. Successive substitution (labelled FPI in Figure 4) suffered from ill-conditioned constant pinch models, which drove the average solving time up as it multiplied the required number of iterations by a factor 10 (from 3 to 39 at worst). In contrast, the Wegstein acceleration method used results from previous iterations to drive the convergence faster, multiplying the required number of iterations by a factor of only 2 in ill-conditioned cases (from 3 to 7 at worst).

The second fastest method is CoolSolve. Even though its largest block is much larger than for sequential method, its analytical expression of the derivatives allowed to avoid warmup iterations for Jacobian estimation using finite-differences. This compensated for the computing load of the large equation system.

The slowest methods were the LaboThApPy Newton-based methods. Their higher convergence time originates from the need for estimating a Jacobian using finite differences, coupled with the required the inlet temperature adjustments in ill-conditioned constant-pinch heat exchanger models. Within this category, the best performer is the Powell hybrid method, thanks to its trust-region guess update strategy. It prevented the iteration variables from being updated too abruptly and kept the solving close to the solution throughout convergence.

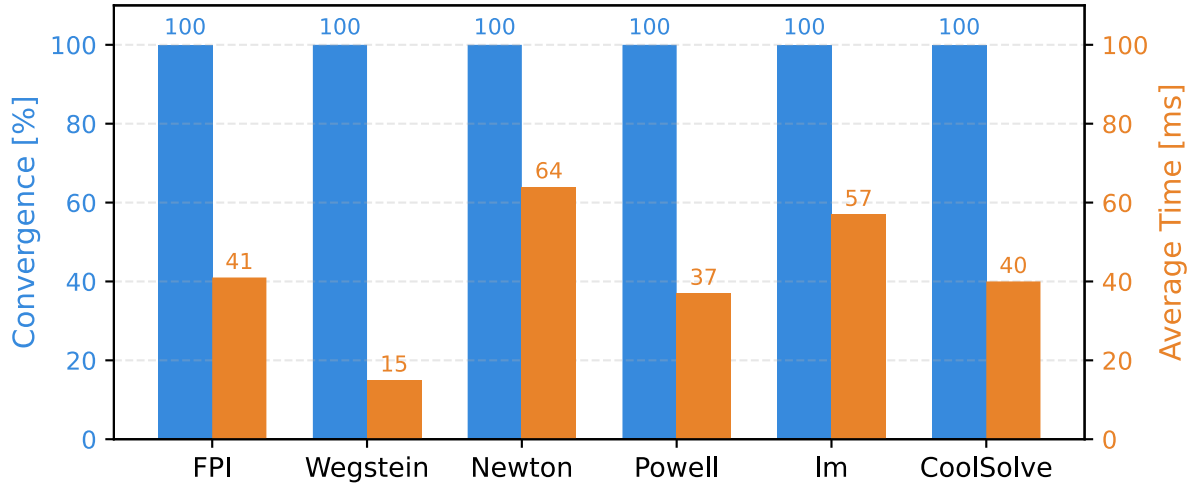

Figure. 4. *Numerical method benchmark results for the heat pump case.*

4.3. ORC Simulation

In contrast to the results from the previous section, all numerical methods did not converge in all cases, as can be seen in Figure 5. The main difference in convergence robustness is between the CoolSolve and the LaboThApPy environments. In this case, the equation-based frame converged for only 51% of the test points compared to between 86 and 100% for the sequential approach. This finding aligns with the results from Winkler et al. [4], who concluded that a higher number of unknowns tends to limit the convergence capability of a solving environment.

Concerning the LaboThApPy methods, Figure 5 shows that FPI methods have a higher convergence rate compared to Newton methods. This is due to early setup iterations showing very small changes in residuals, constructing a badly conditioned Jacobian. Once the constructed Jacobian is used to update the iteration variables, the guesses are updated to physically inconsistent values that corrupt the simulation. Concerning the convergence speed, the discussion is similar to the heat pump section. FPI methods prove to be faster by requiring less iterations to declare convergence.

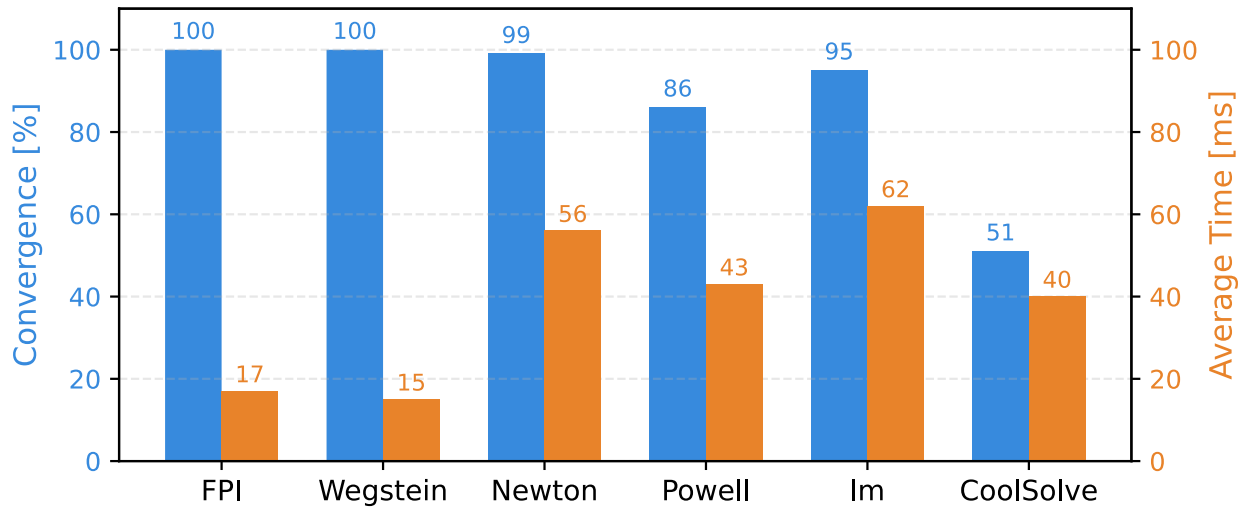

Figure. 5. *Numerical method benchmark results for the ORC case.*

5. Conclusion

This work presented and compared two solving environments for steady-state thermodynamic system simulation: CoolSolve, featuring an equation-based solving approach coupled with a Newton-Raphson solving approach, and LaboThApPy, using a sequential strategy coupled with several numerical solving methods from fixed-point iteration and Newton-based families. The test consisted in the variation of design parameters of thermodynamic component models, resulting in a 10000-simulation test matrix for each of two test systems: a high-temperature heat pump and a preheated-recuperated ORC. The results show that:

- Sequential methods exhibit higher convergence rates than equation-based approaches, due to their reduced number of unknowns. In terms of computation time, the equation-based method from CoolSolve partially compensates for its large system size through the implementation of an automatic analytical Jacobian computation, remaining competitive with a solving time of about 40ms.
- Focusing on LaboThApPy environment, fixed-point iteration methods, and especially the Wegstein method have a faster convergence time (15-40ms) than Newton-based approaches (37-64ms), despite their less favourable mathematical convergence properties.

The latter result can be explained by thermodynamic component models finding their solution in a low number of iterations. Since the fixed-point methods do not externally modify the iteration variables, convergence is declared faster. In contrast, Newton-based methods perform additional iterations for the computation of a Jacobian using finite differences before declaring convergence.

These results suggest that fixed-point methods are the most suited for steady-state design-point thermodynamic system simulation. Future work would consist in extending the comparison to off-design and dynamic simulations frameworks.

Nomenclature

Acronyms

FPI	Fixed-Point Iteration
HTHP	High-Temperature Heat Pump
HX	Heat Exchanger
IHX	Internal Heat Exchanger
ORC	Organic Rankine Cycle
PCM	Phase Change Material

Variables

c	specific heat, J/(kg K)
h	Specific enthalpy, J/kg
$\dot{m}$	Mass flow rate, kg/s
$\dot{Q}$	Heat rate, W
T	Temperature, K

Greek symbols

Δ	Difference
ε	Effectiveness
η	Efficiency
λ	Eigenvalue
∇	Gradient

Subscripts

c	Cold side
cd	Condenser
cp	Compressor
ev	Evaporator
ex	Exhaust
exp	Expander
h	Hot side
is	Isentropic

n	Iteration number
pp	Pump or Pinch-Point
sat	Saturation
SC	Subcooling
sf	Secondary Fluid
SH	Superheating
su	Supply

References

[1] Murthy, A.A., K. Praveen, S., Gopal, K., Ishwaragouda, S. P., Gangadharan K. V., Eshwar Reddy, C., Advancements in dynamic simulation techniques for refrigeration cycles: A comprehensive review. Energy 360 2024;1;100007.

[2] Moradi, R, Cioccolanti, L., Modelling approaches of micro and small-scale organic Rankine cycle systems: A critical review. Applied Thermal Engineering 2024;236;121505.

[3] Witte, F., Tuschy, I., TESPy: Thermal Engineering Systems in Python. JOSS 2020;5(49);2178.

[4] Winkler, J, Aute, V., Radermacher, R, Comprehensive investigation of numerical methods in simulating a steady-state vapor compression system. International Journal of Refrigeration 2008;31(5);930-42.

[5] Thermal Energy System Specialists. LLC : TRNSYS – Available at https://www.trnsys.com/ [accessed 29.3.2026].

[6] Modelica Language. Modelica Specification – Available at https://specification.modelica.org/ [accessed 29.3.2026].

[7] F-chart software. EES – Available at https://fchartsoftware.com/ [accessed 29.3.2026].

[8] MathWorks. MATLAB and Simulink for Engineering Systems – Available at https://www.mathworks.com/ [accessed 29.3.2026].

[9] AspenTech. Aspen Plus – Available at https://www.aspentech.com/en/products/engineering/aspen-plus [accessed 29.3.2026].

[10] Ian H. Bell, Jorrit Wronski, Sylvain Quoilin, Vincent Lemort, Pure and Pseudo-pure Fluid Thermophysical Property Evaluation and the Open-Source Thermophysical Property Library CoolProp. Industrial & Engineering Chemistry Research 2014;53(6); 2498-2508.

[11] Rémi Dickes, Olivier Dumont, Ludovic Guillaume, Sylvain Quoilin, Vincent Lemort, 2018, Charge-sensitive modelling of organic Rankine cycle power systems for off-design performance simulation. Applied Energy;212;1262-81.