Implementing Thermodynamic Cyclic Processes Using the DLR Thermofluid Stream Library

Peter Junglas*

PHWT-Institut, PHWT Vechta/Diepholz, Am Campus 2, 49356 Diepholz, Germany;
*peter@peter-junglas.de

SNE 33(4), 2023, 175-182, DOI: 10.11128/sne.33.sw.10665
Selected ASIM SST 2022 Postconf. Publication: 2023-08-15;
Rec. Revised Improved: 2023-11-23; Accepted: 2023-11-24
SNE - Simulation Notes Europe, ARGESIM Publisher Vienna
ISSN Print 2305-9974, Online 2306-0271, www.sne-journal.org

Abstract. Simulation programs modeling cyclic processes can be used in thermodynamics lectures to promote understanding. Modelica-based simulation environments are a good starting point for the development of such programs, but the handling of the corresponding thermo-fluid standard library is very difficult for non-experts. The recently presented DLR ThermoFluidStream Library is a good alternative, that is easier to use. It provides most components that are needed in typical cases and includes full access to the Modelica media library.
It will be shown, how to use the ThermoFluidStream Library to create examples ranging from the simple Otto and Diesel cycles over the basic Joule-Brayton and Ericsson processes to the water/steam based Clausius-Rankine cycle. Though the construction of concrete processes with given thermodynamic state values and mass flow still requires some effort, one can apply a systematic approach for working models for teaching purposes.

1 Introduction

Thermodynamics is a difficult subject for many engineering students, mainly because of its partly unintuitive nature using abstract notions like enthalpy and entropy. To promote understanding simulation programs can be used, which allow to "experiment" with state changes or complete cyclic processes, such as the collection of Java applets described in [1].

But the construction of such programs is a tedious and time-consuming task, especially if one wants to include examples that use more complex media than the simple ideal gas with constant specific heat capacity.

Instead of writing such programs from scratch, one could use a simulation environment to describe the example models, and leave the actual computation to its solver. Modelica [2] with its physical modeling approach seems to be a good starting point, especially since a comprehensive free model library is available that describes the thermodynamic behaviour of many useful media [3]. Therefore it will be used in the following to build models of the standard processes that are examined in most introductory thermodynamics courses: the Otto and Diesel processes for closed systems and the Joule-Brayton and Ericsson processes for open systems [4, 5]. These models should run on any Modelica platform, especially on the freely available OpenModelica environment [6], and can be employed directly in a thermodynamics course.

The Modelica Standard Library (MSL) already contains an elaborate thermo-fluid library that provides basic components for one-dimensional thermo-fluid flow in pipes, vessels or machines [7]. But due to its very general approach it is much too complicated for the simple didactical applications addressed here. Additionally, corresponding models recurrently fail to run for reasons that are hard to find for non-specialists [8].

The recently presented DLR ThermoFluidStream Library [9] ("ThermoDLR") seems to provide just the level of detail that is needed here: On the one hand it uses the full Modelica media library, on the other hand it offers components for vessels and machines that are much easier to handle than their MSL counterparts. And, most importantly, it uses a very clever, physically motivated scheme to achieve a high robustness [10] that should lead to models that generally run without delicate fine-tuning. This makes it a promising foundation for the construction of didactical examples.

A similar, but simpler and limited approach has been presented in [11], which also describes a Modelica library for thermodynamical examples ("ThermoSimT").

Since its focus is on teaching Modelica, it does not use the complex standard Media library, but a greatly simplified version. Especially its steam/water model only shows basic modeling principles, but is of no practical use. Nevertheless it allows to easily build models of all standard processes and will be used as a benchmark to assess the ease of use and versatility of the new ThermoDLR version.

In the following we will briefly describe the basic ideas of the ThermoSimT library, which provides models of cyclic processes for ideal gas with constant or temperature-dependent specific heat capacity and of the steam/water based Clausius-Rankine cycle. Then we will use the ThermoDLR library to implement similar models, utilizing the Media library to get valid practical results. Due to the didactical purpose and since we are only interested in equilibrium behaviour, the models have some unusual features: Heat transfer is done very fast, the characteristics of compressors or turbines don't really matter, and the values of the mass flow and several state variables are given in advance.

We will show, which problems appeared during the implementation, and present ways how to deal with them. This will help to produce similar models for own teaching purposes. As a starting point, all models described here can be downloaded freely from [12].

2 Cyclic Processes in ThermoSimT

Since [11] is a textbook on modeling and simulation, the main purpose of the ThermoSimT library is to teach the design and construction of a Modelica library. But thermo-fluid modeling is a very difficult task, therefore a lot of simplifying assumptions had to be made: The mass flow is constant and the flow has always the same direction, i. e. all connections are uniquely defined as input or output ports. The components have no states describing an internal change, but the thermodynamic variables just jump from the input to the output state. As a consequence, the described models are static, time changes can only by implemented by changing work or heat flows.

Since stream connectors [13] are much too advanced for an introductory textbook, the connector is based on the preliminary version of the thermo-fluid library described in [3]. ThermoSimT contains components for simple devices such as a cylinder, a heater, a pump and a turbine, together with source and sink components and a state measurement device that outputs all relevant thermodynamical variables.

Pump and turbine are identically modeled as simple turbo machines based on an isentropic state change with a simple linear characteristic

$$\dot{m} = K\omega.$$

The simple Media library covers the ideal gas with constant heat capacity ("simple air"), the NASA dry air model [14] and a simple model for steam and water, using ideal gas and ideal fluid equations together with a Clausius-Clapeyron based vapor pressure curve.

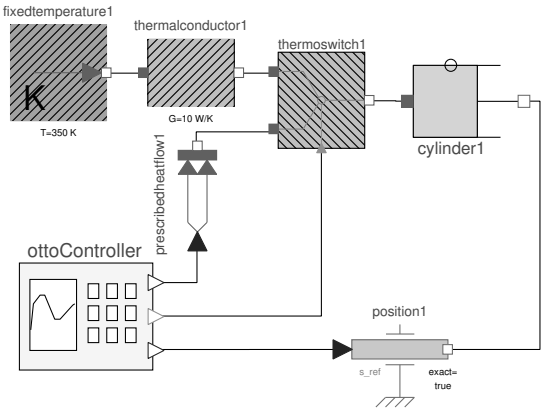

Figure 1: Otto cycle using ThermoSimT.

With these components a model of an Otto cycle can be built easily (cf. Figure 1). The thermodynamic computations are done in the cylinder component, while additional blocks provide a test stand defining the position of the piston and the amount of external heat. Models using simple air and dry air are provided, as well as a similar example for the Diesel process.

Much more interesting from a modeling perspective is the model of the Joule-Brayton cycle (cf. Figure 2), which describes the actual flow of the medium between components. The mechanical work for the compressor is provided by a constant torque block, while the external load at the turbine is modeled using a simple generator and a resistor. The ThermoSimT library does not work with a cyclic topology, but an additional cooler at least brings the state of the medium back to its initial state. Versions for simple air and dry air are provided, as well as an identically looking model using the simple steam/water medium, which actually makes it a Clausius-Rankine cycle.

The final example is the Ericsson cycle, which contains several compressors and coolers as well as turbines and heaters, to approximate an isothermal behaviour in the turbo machines. It can be modelled easily with ThermoSimT (cf. Figure 3).

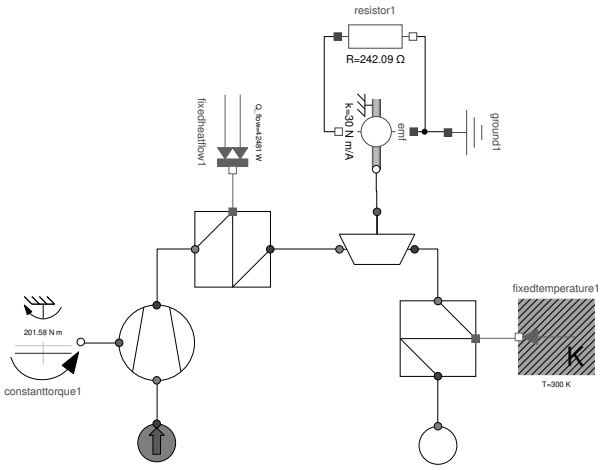

Figure 2: Joule-Brayton cycle using ThermoSimT.

3 Otto and Diesel Processes

Since a cylinder model, which is at the heart of the Otto cycle model, is not included yet in the ThermoDLR library, one has to build it oneself. Fortunately, a complete volume model already exists, together with several variants. They all inherit from the parent class `PartialVolume`, which provides most of the variables and equations needed, together with optional `HeatPort`, `Inlet` and `Outlet`. To construct a `CylinderVolume`, one simply extends `PartialVolume`, adds a mechanical `Flange` and provides simple equations for the definition of the volume, the force and the work at the flange. To simplify the drawing of a T-s diagram, an explicit variable for the entropy is added.

Exchanging the cylinder model in the ThermoSimT Otto cycle is all that remains to do. The new model works immediately with `DryAir`, for `SimpleAir` one has to extend its range of validity by defining

```
SimpleAir(T_min=200, T_max=2000)
```

The Diesel controller component needs the internal pressure to create an isobaric process, which can be supplied by a sensor at the optional `Outlet` of the `CylinderVolume`.

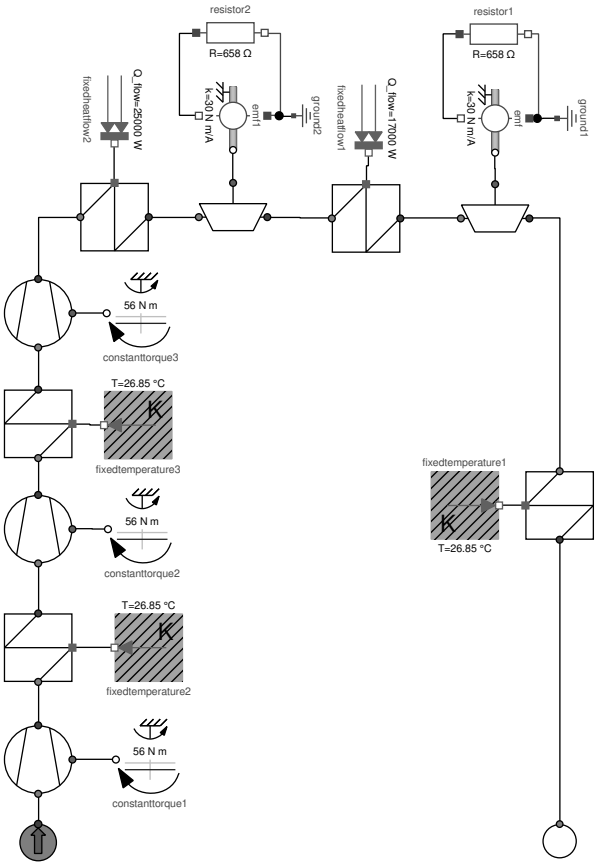

Figure 3: Ericsson cycle using ThermoSimT.

4 Joule-Brayton Process with Ideal Gas

The construction of a model for the Joule-Brayton cycle seems to be almost trivial. One starts by replacing the thermodynamic components from the ThermoSimT example with their counterparts from ThermoDLR: `Source`, `Sink`, `Compressor`, `Turbine`, and `ConductionElement` as replacement of the `Heater`. All parameters have their default values except for the initial pressure and temperature at the `Source`. That the mass flow can not be defined anywhere, is due to the different concept of ThermoDLR: $\dot{m}$ is a dynamic variable and is computed in the context of the whole model – here probably mainly depending on the compressor parameters. With the ideas from ThermoSimT in mind, this seems to be strange, and indeed: Starting the simulation results in the infamous error *"Failed to solve nonlinear system using Newton solver during initialization."*

To find the reason of this problem, one examines a simple test model consisting of a `Source`, a `Compressor` and a `Sink`. The `Source` defines starting pressure $p_1 = 1$ bar and temperature $T_1 = 300$ K, the `Sink` the final pressure $p_2 = 6$ bar. Using the same torque as in the ThermoSimT model, the simulation doesn't run, which is not surprising, since the compressor from ThermoDLR uses a completely different characteristic curve and default operating point.

To find an adequate torque value, one has to analyze the equations used in the `Compressor` component. Using default parameters and neglecting regularisations for small variable values, they are:

$$\begin{aligned} \frac{p_2}{p_1} &= \frac{\omega^2}{\omega_{ref}^2} - \frac{\dot{m}^2}{\dot{m}_{ref}^2} + 1 \\ w_t &= \frac{\kappa}{\kappa - 1} R T_1 \left(\left(\frac{p_2}{p_1} \right)^{\frac{\kappa-1}{\kappa}} - 1 \right) \\ &= \frac{\omega \tau}{\dot{m}} \end{aligned}$$

They define a quadratic pressure characteristic and compute the work using the explicit formula of an isentropic process for an ideal gas with constant heat capacity. Inserting the given state variables, the desired mass flow $\dot{m} = 0.1$ kg/s and the default parameter values, one easily computes values for ω and τ.

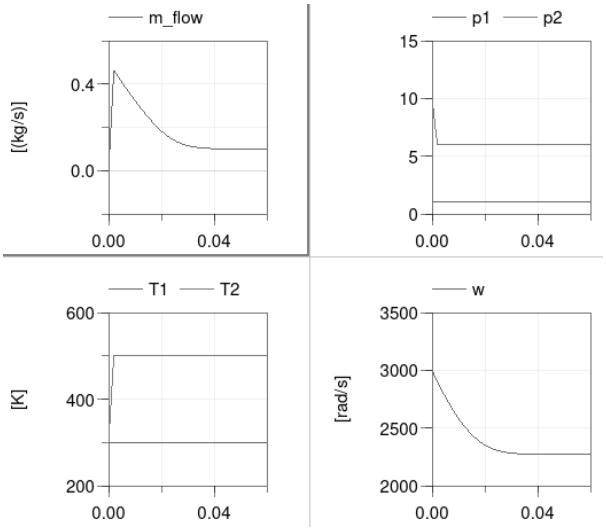

Figure 4: Results of the compressor test model.

Using this torque value for the constant torque and setting the initial value of the rotational velocity to ω (at least approximately), the test model runs and has the correct results (cf. Figure 4). In accordance to the philosophy of the ThermoDLR library, the initial value of $\dot{m}$ is chosen to be 0, so that the equilibrium is reached by simulating the powering up of the system.

A similar computation can be done for the turbine. Unfortunately, the identical characteristic curve of compressor and turbine contains a very strict regularisation, whenever the pressure drops (as in the turbine). But of course it's simple to create a copy of the curve model and use the quadratic characteristic for the turbine as well. Trying to compute the torque for the given state variable values, one gets a negative value under a square root. This problem can easily be fixed by changing the operation point, setting $\dot{m}_{ref} = 0.05$ kg/s. Finally one utilizes the linear characteristic of the simple generator to compute the resistance R, and gets the requested results.

For the heater, no new computations are necessary, since the needed heat is fixed by the thermodynamics. Combining the components (for a start without the final cooler) using the new parameter values, one gets a working Joule-Brayton process, which almost reproduces the required values. Only the pressure p_2 after the compressor is slightly higher in the combined model, maybe due to the effect of several regularisations.

Instead of going through the complete equations, the fine-tuning can easily be done with a few manual parameter changes: First one lowers the torque at the compressor, until p_2 reaches the desired value. Here, the `MultiSensor` components are very convenient, which display state variables directly in the graphical model. Now all pressure and temperature values are correct, only the mass flow is a bit too small. Better than fiddling with several parameters at once, one can use a simple scaling procedure: Increasing all works and heats by a constant factor q one can change only the mass flow, not the thermodynamical state. Setting $q = \dot{m}_{desired}/\dot{m}_{actual}$, increasing $\dot{m}_{ref}$, τ and $\dot{Q}$ by a factor q and dividing R by q, one eventually reproduces all ThermoSimT values.

Finally one adds another `ConductionElement` with a large heat transfer coefficient to bring the temperature down to the start value. With the ThermoDLR library, one can now make it a real cycle by deleting the `Sink` and `Source` components and closing the circle with an intermediary `Volume` element. The initial state, which had been defined by the `Source`, is now given as initial value of the `Volume`.

5 Ericsson Process with Ideal Gas

To simplify the construction of an Ericsson model, a subsystem is built for a line of three compressors and intermediate coolers (cf. Figure 5), as well as a similar model of two turbines with intermediate heater. Both components contain sensors to output the complete work and heat supplied to the line. This will be useful to compute the overall efficiency in the complete model.

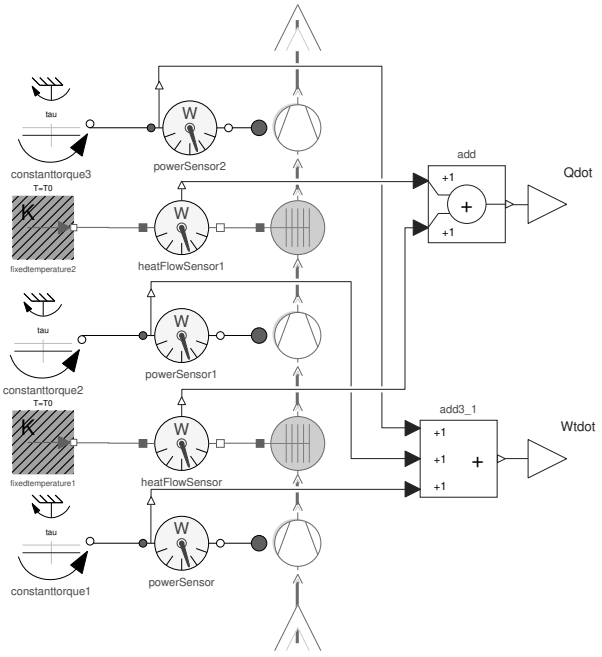

Figure 5: Line of compressors for the Ericsson cycle.

Providing correct parameter values, these compressor and turbine components show approximately isothermal behaviour. Using them in the Joule-Brayton model and guessing reasonable values for the work-related parameters τ and R, the Ericsson model runs – at least, if one lowers the intermediate heating temperature a bit. As before, one can use fine-tuning to get the given pressure values and scaling to reach the correct mass flow. The resulting model is more stable, so that one can raise the intermediate heating temperature to the incoming value to better approximate an isothermal process.

Now we can make good use of the additional possibilities that are supplied by ThermoDLR, and include a heat exchanger that utilizes excess heat after the turbine for preheating the gas before entering the heater (cf. Figure 6). To reach the same maximal temperature as before, one starts with a low heat transfer coefficient k_{NTU}, then gradually lowers the supplied heat, while rising k_{NTU}. The final Ericsson model is much more realistic than the previous version based on ThermoSimT.

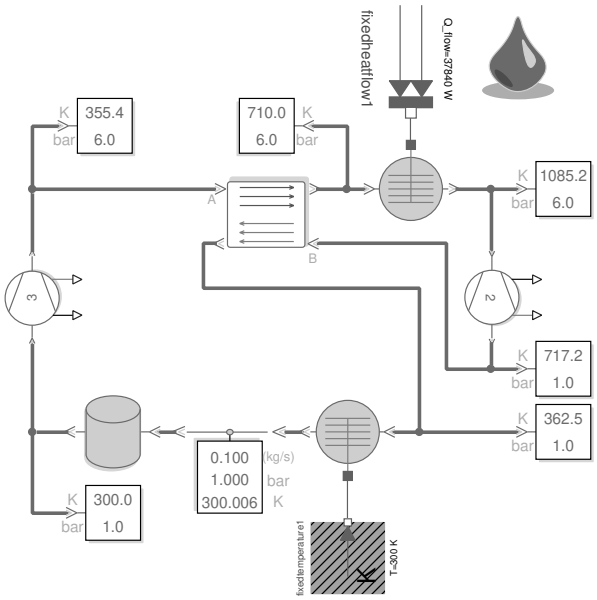

Figure 6: Ericsson cycle with heat exchanger using ThermoDLR.

6 Joule-Brayton Process with Dry Air

Building a Joule-Brayton example based on dry air as a medium should be done easily by simply changing `SimpleAir` to `DryAirNasa`, but unfortunately, the new model doesn't run: To compute the temperature from the given enthalpy, the function $h(T)$ has to be inverted – but the Dekker-Brent based solver is supplied with a start interval without a zero.

Looking closer at the `DryAirNasa` implementation, one finds another problem: The computation of the isentropic enthalpy is based on the approximation of constant heat capacity. This may be good enough for many practical applications, but in a teaching context one should present a correct solution. Since the utilised function `isentropicEnthalpy` is not declared as `replaceable`, one has to create a copy of the `DryAirNasa` class and its base class `SingleGasNasa` and supply a computation that

is based on the constant entropy. Additionally, the function `dp_tau_const_isentrop` that is used in ThermoDLR to define the compressor characteristics, is based on a constant heat capacity formula as well and has to be replaced by an exact version `dp_tau_const_isentrop_S`.

Using these more accurate methods, one is still faced with the original problem: The inversion gets a wrong start interval. This is due to the initial value $\dot{m}(0) = 0$: The direction of the flow is undefined initially and the algorithm tries values that are going into the wrong direction. Simply setting $\dot{m}(0) = 0.1$ kg/s, which is given anyhow, succeeds and the model runs – but the values of pressure and mass flow are not the required ones. Obviously, the seemingly small difference between `SimpleAir` and `DryAir` leads to larger deviations than expected.

To find correct parameter values, one again starts with a simple test model for the compressor. Either by a direct computation – which can be easily done for dry air with a small Matlab script – or by a few trial and error steps, one finds values that lead to the required pressure and mass flow. Due to the internal inversions, the model is less stable than its `SimpleAir` counterpart: Even if one sets the correct torque value, one still has to supply an initial value for ω that is large enough. For simplicity one can instead directly define ω at the flange. The corresponding computation for the turbine again shows that $\dot{m}_{ref}$ has to be lowered. After that, the computation leads to the correct value of the resistance R. Closing the cycle with a cooler and a volume element, one arrives at a Joule-Brayton model with dry air that – after a bit of the usual fine-tuning – reproduces the given state and mass flow values.

7 Clausius-Rankine Process with Standard Water

Basically, the Clausius-Rankine process is a Joule-Brayton process with water and steam as a medium: Though its technical realisation is much more complicated due to the phase transitions from water to steam and back, conceptually it consists of a pump, a heater, a turbine and a cooler. The media library includes `StandardWater`, a precise description of water and steam based on the IAPWS-IF97 formulation [15]. Yet, one cannot just use one of the Joule-Brayton models and change the medium, since the compressor and turbine components of ThermoDLR use explicit ideal gas relations. For incompressible fluids the `Pump` component can be used, a suitable turbine component is not provided in the library. Also, completely different values of the state variables than before will be employed: The pressure ranges from 0.1 bar to 60 bar, the highest temperature should be 500 °C and the mass flow 10 kg/s.

To find working parameters, one can start with a simple test model for the pump. The ThermoDLR `Pump` component provides two different characteristics: a centrifugal pump and a simpler nominal pump, which is used in the following. Setting reasonable nominal values and basic parameters, one can once again use the model equations and the given state and mass flow to compute a correct value of the rotational speed ω. Adding a standard heater (i. e. a `conductionElement`), one can reach the given temperature by first estimating, then fine-tuning the needed heat flow. The heating process includes the complete vaporisation of the water, succeeded by an overheating of the steam, but all this is automatically taken care of by the state equations used in IF97. Adding corresponding sensors, one can easily monitor the dryness fraction x everywhere in the cycle model.

A generic turbine model `TurbineG` that works for `StandardWater` (and any any other fluid medium) can be easily constructed: It inherits from the provided `PartialTurboComponent` and uses the explicit characteristic function `dp_tau_const_isentrop_S` that has already been defined for the `DryAir` example. Since the medium changes its phase from hot steam to wet steam inside the turbine, one can expect numerical difficulties, and in fact: Using default parameter values, the usual test model doesn't run, because the pressure reaches values below the triple point of water. To find working values, one initially starts with a simpler process, going from 60 bar to 30 bar instead of trying to reach 0.1 bar immediately. Changing ω_{ref} und $\dot{m}_{ref}$, one soon gets a model that at least runs for a very short time. This makes it possible to study its behaviour and find the reason of the problems. Adapting ω, the model finally runs to the end. Now one can use the usual fine-tuning to gradually lower the pressure to the requested value and to switch from the constant ω input to the simple consumer model. The needed resistance value is completely unrealistic, but can finally be scaled to a meaningful value by changing the transformation coefficient of the simple generator model.

Now a first version of a Claudius-Rankine cycle can be built by combining the pump-heater and turbine models. The model runs only for a short time, before the final pressure again drops below the triple point. Enlarging the value of R by a factor of 100, the model runs to the end, though the state values are far from the requested ones. Once again one goes through the fine-tuning process, changing ω, $\dot{Q}$ and R, to reach the requested state values. This is much more tedious than in the previous examples, because the very different properties of water and steam lead to a small range of working parameters. Additionally, their strong coupling sometimes results in a counterintuitive behaviour, as in the following scenario: One finds a too low temperature behind the heater and increases the heat flow, which leads to an increase of the mass flow and a falling temperature. Another problem is the instability of the standard DASSL solver that is used in the simulation program Dymola: Occasionally, the solver hangs for certain parameter values and has to be stopped. Changing to the Esdirk23a solver drastically reduced the frequency of such events. With a certain amount of perseverance, results within 1 % of the requested values could be reached.

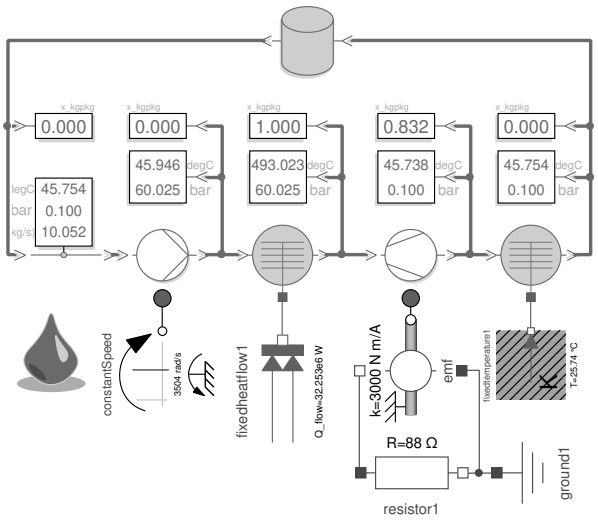

Figure 7: Clausius-Rankine cycle using ThermoDLR.

The next step is the addition of the cooler, which is supplied with a fixed t emperature. Choosing the obvious value of 45.80° C – the condensation temperature at 0.1 bar –, one doesn't reach $x = 0$, therefore one has to use a smaller value and to increase the heat transfer coefficient U c onsiderably. Finally adding the volume element to close the circle, one again gets very different results and has to tune the cooler parameters once more, as well as the initial temperature of the volume. But in the end one reaches a complete Clausius-Rankine cycle with the requested state values (cf. Figure 7).

8 Conclusions

All examples of the simple ThermoSimT library could be implemented in ThermoDLR, as well as more complex processes using a heat exchanger or an accurate water model. ThermoDLR is designed to create models that run immediately, nevertheless it was nontrivial to build models that reproduce the given state values and mass flows. A systematic approach often worked that was based on the following steps:

- Start with simple models, where initial and final states are defined by `Source` and `Sink` components.
- Change the operation points of turbo machines to values near the given state values.
- Use (simplified) versions of the component equations to get good starting points for external parameters.
- Gradually fine-tune parameters to approximate the given thermodynamic states. If necessary, change the mass flow by a scaling procedure.
- Combine partial models and reiterate the fine-tuning process.
- Finally close the cycle with a `Volume` component and properly define its initial values.

Sometimes it was difficult to find proper parameters to make the model run at all or to change the state into the required direction – especially for a complex medium such as water and steam. In such cases, it proved useful to set the initial value of the mass flow directly to the required value or to define operation points very close to the target values. Generally, such a model runs at least for a very short time. This is helpful, because seeing the results of small parameter changes directly can provide clues on what to do to finally reach a running model. Other helpful measures were the introduction of control valves to decouple parts and define intermediate states, or even to change the solver. When the model finally works as required, it is generally more

stable than the previous versions and allows for easy parameter changes.

The design of the ThermoDLR library, and especially its ability to start with zero mass flow and model a power-up situation, makes the modeling of thermofluid systems much simpler for non-specialists than the complex Modelica Thermo-Fluid library. On the other hand, the consistent use of the mass flow as a state variable sometimes leads to unexpected behaviour, when parameters are changed or submodels combined. ThermoDLR provides useful basic components that are easy to understand down to the equation level and are easy to extend due to a simple, but convenient inheritance hierarchy. The supplied `MultiSensor` components are very helpful during the fine-tuning phase. Some important elements are missing yet, such as a generic turbine with a corresponding characteristic function or a cylinder volume, and had to be added here.

On the whole, ThermoDLR proved to be a very useful tool for the construction of cyclic process models that can be used for demonstration purposes in thermodynamics lectures. Of course, it requires some effort to familiarise oneself with the library, but going in simple steps, as has been shown before, the learning curve is not very steep and can be mastered by lecturers, who are not experts in thermo-fluid modeling.

Acknowledgement

The author is grateful to Dirk Zimmer, Niels Weber and Michael Meißner for introducing him to the DLR Thermofluid Stream Library and for providing helpful hints during the construction of the described models.

References

[1] Junglas P. Simulation Programs for Teaching Thermodynamics. *Global J of Engng Educ*. 2006; 10(2):175–180.

[2] Modelica Association. *Modelica® – A Unified Object-Oriented Language for Systems Modeling, Language Specification Version 3.5*. URL `https://modelica.org/documents/MLS.pdf`

[3] Casella F, Otter M, Proelss K, Richter C, Tummescheit H. The Modelica Fluid and Media library for modeling of incompressible and compressible thermo-fluid pipe networks. In: *Proc. 5th Int. Modelica Conference*. 2006; pp. 631–640.

[4] Moran MJ, Shapiro HN, Boettner DD, Bailey MB. *Fundamentals of Engineering Thermodynamics*. Hoboken, NJ: John Wiley & Sons, 9th ed. 2020.

[5] Cerbe G, Wilhelms G. *Technische Thermodynamik*. München: Carl Hanser, 19th ed. 2021.

[6] Fritzson P, Pop A, et al. The OpenModelica Integrated Environment for Modeling, Simulation, and Model-Based Development. *Modeling, Identification and Control*. 2020;41(4):241–295.

[7] Franke R, Casella F, Sielemann M, Proelss K, Otter M. Standardization of thermo-fluid modeling in Modelica.Fluid. In: *Proc. 7th Int. Modelica Conference*. Como, Italy. 2009; pp. 122–131.

[8] Drente P, Junglas P. Simulating a simple pneumatics network using the Modelica Fluid library. *SNE Simulation Notes Europe*. 2015;25(2):85–92.

[9] Zimmer D, Weber N, Meißner M. The DLR ThermoFluidStream Library. In: *Proc. 14th Int. Modelica Conference*. Linköping, Sweden. 2021; pp. 225–234.

[10] Zimmer D. Robust object-oriented formulation of directed thermofluid stream networks. *Mathematical and Computer Modelling of Dynamical Systems*. 2020; 26(3):204–233.

[11] Junglas P. *Praxis der Simulationstechnik*. Haan-Gruiten: Verlag Europa-Lehrmittel. 2014.

[12] Junglas P. *Thermodynamic Cyclic Processes library in Modelica*. URL `http://www.peter-junglas.de/fh/simulation/thermocycle.html`

[13] Franke R, Casella F, Otter M, Sielemann M, Elmqvist H, Mattson SE, Olsson H. Stream connectors – an extension of Modelica for device-oriented modeling of convective transport phenomena. In: *Proc. 7th Int. Modelica Conference*. 2009; pp. 108–121.

[14] McBride BJ, Zehe MJ, Gordon S. NASA Glenn Coefficients for Calculating Thermodynamic Properties of Individual Species. *Nasa report tp-2002-211556*, NASA. 2002.

[15] Wagner W, Cooper JR, et al. The IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam. *J Eng Gas Turbines and Power*. 2000;122(1):150–182.