# Steady State Initialization of Vapor Compression Cycles Using the Homotopy Operator

Christian Schulze[1] Andreas Varchmin[1] Wilhelm Tegethoff[2]

[1]TLK-Thermo GmbH, Germany, {c.schulze,a.varchmin}@tlk-thermo.com
[2]Institut für Thermodynamik, University of Braunschweig, Germany, w.tegethoff@tu-braunschweig.de

## Abstract

This paper presents a concept how a hybrid DAE of a vapor compression cycle can be initialized in steady state using the homotopy method. A simplified equation system for a vapor compression cycle is described and its computational causality explained. It is discussed how additional boundary conditions can be applied to the simplified equation system, which do not apply to the actual equation system. The robustness and CPU time for different cases is examined and discussed based on transition plots.

*Keywords: Vapor Compression Cycle, Homotopy, TIL, ThermalSystems*

## 1 Introduction

Modelica is nowadays widely used in industry and research for object oriented modelling and transient simulation of cyber physical systems. Several Modelica compilers are available and the compatibility between them is continuously improving.

Although Modelica is used for transient simulation of dynamic models, the user is often only interested in the steady state results. And even if the transient simulation is wanted, the initial state of the model is preferred to be in steady state. These arguments particularly apply to vapor compression cycles, because they are computationally very expensive.

Models which were implemented for steady state simulation are fundamentally different from transient models, because simplifications or analytic solutions such as the NTU method (see Verein deutscher Ingenieure (2013)) can be applied. From a user's perspective it would be most convenient to have just one model for both cases. If it is not possible to merge both models, then the two different models should provide the same level of detail and precision.

A typical simple model optimized for steady state simulation of a vapor compression cycle could have two (algebraic) state variables. A dynamic model using finite volume method may have more than 100 (continuous time) state variables, which have to be brought into a steady state. So the dynamic models used for transient simulation are structurally more complex than models which have been optimized for steady state calculations.

If a dynamic model is initialized in steady state, all continuous time states and other initial unknowns (e.g. `fixed =false` parameters) are calculated from an nonlinear systems of equations of the initialization problem. This initialization equation system is a result of the initial equation `der()=0` for all continuous time states.

Most Modelica tools translate the hybrid DAE described by the Modelica equations to an explicit hybrid ODE to solve it. Nonlinear sub-systems are solved inline e.g. using Newton's method. Some tools also provide a DAE Solver to handle both, the differential equations and the algebraic equations. However, the focus of these solving methods is the transient integration rather than solving large nonlinear systems. Often the nonlinear solvers fail to solve the nonlinear equation system of the initialization problem.

One solution to improve the chance for a convergence of the nonlinear system of the initialization problem would be to improve the root finding method. Another solution would be to simplify the model. The homotopy operator is targeted at the second option. The nonlinear equation system of the initialization problem can be simplified to make sure that also less sophisticated solvers find a solution.

Vapor compression cycles are computationally very expensive. The fluid properties used in these models are highly nonlinear and based on complex equations (multiparameter equations of state). The fluid properties also have a very limited numerical range of validity, e.g. evaluating these properties for a negative pressure, temperature or density is impossible. As the equations have been estimated to describe measurement data, they also have an even more restrictive physical range of validity. So it is essential that the system state always is within the range of validity.

## 2 Homotopy Operator

### 2.1 Rationale

If a dynamic model is initialized in steady state, many start values are required. From an engineer's perspective, reasonable start values are either obvious and easily defined, or they are almost impossible to provide, because they are actually the desired result of the model. The solving procedure is obscure because it is intellectual property of the tool vendor, and often it is not clear if the solving process has failed because of physical or numerical problems.

Usually the engineer can observe that parts of the system seem to diverge to problematic working conditions, but there is no way to influence the solving procedure.

The larger algebraic nonlinear systems are, the harder it is to trace back the reason for convergence problems of nonlinear systems. It is very difficult to provide good feedback to the user about the underlying problem.

Homotopy is a concept to increase the robustness and simplify the solving procedure for algebraic nonlinear systems. The idea is to use a simplified model (to replace complex dependencies by simplified ones), to calculate a first guess for the result of the actual equation system, and to make use of the similarity of the systems during the transition from the simple to the complex equation system.

For simple nonlinear systems a common Newton solver will be more efficient than using homotopy method. The larger the nonlinear systems are, the more effort has to be invested into the numeric root finding method. The homotopy method enables the engineers to find better ways to define start values, and to focus the solving procedure on relevant aspects. So in other words, it enables to describe the aspects which are obvious to the engineer with a physical understanding of the system. An engineer always crosschecks the plausibility of the calculated system states - the solver cannot do that.

An engineer would also try to generate guess values based on a levelled approach. So first the focus should be on the top level, to set general working conditions including guess values for interface values between subsystems. After that each subsystem has to be processed using the interface values. Interdependencies between subsystems can and have to be broken completely at the interfaces between the subsystems.

The homotopy method requires a simple equation system, which can be used to calculate the same state variables that are calculated from the actual equation system. Additionally it is essential that the resulting solution of the state variables only changes continuously during the transition from the simple to the actual equation system. So the causality must be compatible, and the transition must be continuous. If these two requirements are fulfilled, then the homotopy method can be applied.

## 2.2 Homotopy in Modelica

The Modelica implementation of homotopy as presented in Sielemann et al. (2011) uses one global dimensionless transition factor $\lambda$ which is zero for the simple equation system and one for the actual equation system.

The homotopy operator looks like a function in Modelica. It outputs a linear combination of the two inputs:

```
function homotopy
  input Real actual;
  input Real simplified;
  output Real val;
end homotopy;
```

The homotopy function switches term-wise between actual and simple equations. A small example could look like this:

$$y = \mathrm{Term}_{\mathrm{actual}} \cdot \lambda + \mathrm{Term}_{\mathrm{simple}} \cdot (1-\lambda) \quad (1)$$

$$z = y^2 \quad (2)$$

Even though the term for the calculation of $y$ is switched linearly in eq. 1, the eq. 2 to calculate $z$ is switched nonlinearly because it is nonlinearly dependent on $y$.

In general all derivatives have to be set to zero, to initialize a system in steady state, consequently there will be a large initialization nonlinear system. The homotopy function can be used anywhere in the model equations or initial equations. The function is intended to be used to support solving nonlinear systems and it will be ignored in other cases.

In Dymola there is a separate symbolic analysis of the simplified equation system ($\lambda = 0$) that may have a different computational causality and structure. E.g. the former iteration variables could be calculated explicitly. There is also a symbolic analysis for the initialization nonlinear equation system dependent on the parameter $\lambda$ which represents the `der()=0` initial equations. The latter equation system has the same structure, dimension and computational causality as if the homotopy method was not used, although the homotopy function calls were inlined.

The solving procedure often implements a few basic steps:

1. $\lambda$ is set to 0
2. The (separately analysed) simplified equation system is solved
3. The initialization equation system with the current $\lambda$ is solved using a common root finding method with start values from the last evaluation
4. $\lambda$ is increased
5. if $\lambda < 1$, repeat steps 3-5, else final execution of 3

Instead of solving one equation system, an equation system has to be solved for each $\lambda$-value. So by design the homotopy method is more computationally expensive than solving the actual equation system directly. However, comparing the CPU time is difficult, because the start values have a huge impact, and the chance of convergence for different boundary conditions is also important. Common root finding method usually fail to initialize a larger system in steady state without using homotopy method. Considering robustness and the lack of good start values, it is worth investigating the homotopy method.

Dymola usually uses a common root finding method for nonlinear systems and it is required to activate the homotopy method use from the beginning (this can be done with an Advanced-flag, or an annotation in the model). By default, the initial lambda step size is 0.1 and it remains unchanged until one evaluation fails. If that happens, the step size is reduced and the solver will try again. But the step size will not be reset to 0.1, it remains reduced for the rest of the solving procedure. This sometimes has a negative effect on the computation time, if the transition is highly nonlinear for low $\lambda$-values.

Initialization in a partially steady state is usually not meaningful. Parts of a model which are not initialized in steady state can be seen as a dynamic boundary condition to the connected other model part which shall be initialized in steady state. If the time derivative of all states in a subsystem are zero given dynamic boundary conditions, then usually that subsystem is not in a steady state. The time derivative will be zero at the simulation start, but will be unequal to zero as soon as $t > t_{\text{start}}$.

The Modelica homotopy operator is supported by different Modelica compilers such as OpenModelica, Dymola, and SimulationX. But currently not many libraries extensively use this method. E.g. the ThermoPower library (Casella and Leva, 2006) provides models with homotopy as discussed in Casella et al. (2011). The ClaRa library (Gottelt et al., 2017) also applies the homotopy operator, but this feature is not implemented completely. The ThermoCycle (Quoilin et al., 2014) uses homotopy. The TIL library (Gräber et al., 2010), which is also known as ThermalSystem library, does not yet support homotopy, but is the basis for this publication.

## 2.3 Simple Example

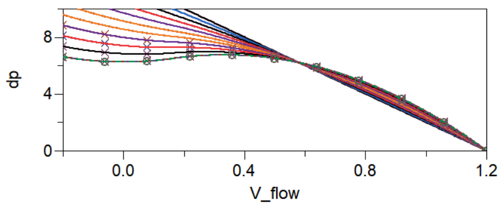

**Figure 1.** Nonlinear pump characteristic with transition to a simplified solution. The simplified model is linear, can therefore be solved symbolically and has only one solution.

In figure 1 the transition between a nonlinear pump characteristic and linear approximation equation is shown. The Modelica code is listed in section A.

These nonlinear pump characteristics can cause problems because algebraic equation systems based on them may have several solutions. E.g. if the pressure difference is known and not dependent on the volume flow rate, there might be three possible volume flow rates (e.g. at $dp = 6.5$). In fact this is also a common problem for real systems. Homotopy can help to find the wanted (rightmost) solution without giving a start value.

First the simple equation system can be rearranged symbolically to calculate the mass flow rate because it is a linear relationship. Then this result will be used to solve the nonlinear equation system $6.5 = dp(V_{\text{flow}})$ with $\lambda = 0$. So the residual should be equal to zero for the calculated start values. Subsequently lambda is increased (e.g. $\lambda = 0.1$) and the equation system is solved again. But since the new equation system is almost the same as the last one, it is easy for the root finding method to find a solution close to the start value.

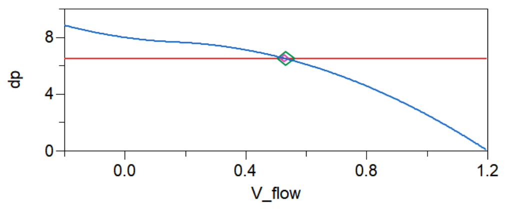

**Figure 2.** Solving the pump characteristic for $dp = 6.5$ at $\lambda = 0.7$. The last result of the nonlinear system is used as start value (large diamond marker) to find the next result (small diamond marker).

In figure 2 the curve for $\lambda = 0.7$ is shown. The large diamond marker represents the start value from the last solution for $\lambda = 0.6$. The small diamond marker represents the result of the equation system. The start value and the solution are very close to each other, and by reducing the $\lambda$ step size, the values will be even closer.

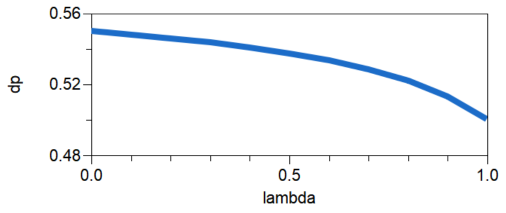

**Figure 3.** Solutions of the equation system plotted over $\lambda$. The transition is continuous and describes the transition from the simple equation system to the actual one.

If the transition is continuous, then solutions calculated for different lambda form a continuous solution path from the simple to the actual equation system. The transition of the resulting algebraic state is shown in figure 3. There must not be a discontinuity or pole in this $\lambda$-plot. The $\lambda$-plot could also be considered as a root locus plot - the solutions for different $\lambda$-values are connected to a line.

## 3 Initializing Vapor Compression Cycles with Homotopy

We are focusing on finite volume models with balance equations for mass, energy, and momentum. Some components such as the valve and compressor have steady state balance equations. The dynamic heat exchangers are dicretized one-dimensionally. In contrast to that the separator model is a 0-D model with dynamic mass and energy balance. For more details see Schulze (2013). If the dynamic component models shall be initialized in steady state, then an additional initial equation has to be added to set the time derivative of the continuous time state to zero. The continuous time state variables are pressure and specific enthalpy.

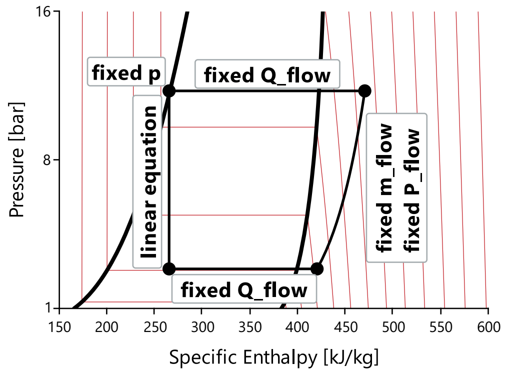

**Figure 4.** User-defined boundary conditions for the simplified equation system. Heat flow rates in each component are predefined, mass flow rate, separator pressure, and a linear valve characteristic.

In figure 4 the boundary conditions used for this work are shown. The basic concept for the simple equation system is to describe the state of a whole vapor compression cycle using a number of simple conditions:

- Fixed heat flow rates in each heat exchanger
- Fixed mass flow rate and power in the compressor
- Fixed pressure and filling level in the separator
- Linear characteristic in the valve

For this simplified nominal working state, all mass flow rates, enthalpy and pressure states can be calculated. Due to the steady state mass balance all mass flow rates are equal to the one set in the compressor. The high pressure is set by the user in the separator. The low pressure is calculated from the mass flow rate, the high pressure, and linear valve characteristic. Starting from the separator outlet enthalpy in a saturated state (which is known due to the fixed pressure), all enthalpies can be calculated. In a discretized heat exchanger the heat flow rate density is assumed to be constant, so if the control volumes have the same size, the enthalpy difference between two neighboring volumes is constant.

The valve characteristic is a linear equation which connects high pressure, low pressure, mass flow rate, and valve opening area. The valve opening area is usually controlled.The valve opening area must not be removed from the simple equation system, because the controller state will be calculated from it.

The above described simplified equation system does not require the calculation of fluid properties. Usually the heat flow rate is calculated from a temperature difference, but as the heat flow rates are predefined, the temperature has no influence on the result of the simplified equation system. If in contrast a complex heat transfer model would be replaced only by a constant heat transfer coefficient, then the temperature-enthalpy relation is not broken (Casella et al., 2011).

During the transition between the simplified and actual equation system the states must stay within the range of validity of the fluid properties. So the definition of the simple system and the transition to the actual equation system are the most important challenges when using homotopy.

Because the simple equation system is analysed separately, it is solved symbolically and no start values are required. This is one of the main advantages when using homotopy. The user only has to provide the values listed above and nothing more. Usually the user knows how to choose this nominal working state.

This concept also works for more complex vapor compression cycles. In case there are more than two pressure levels, there has to be an additional separator or a component such as an ejector. An additional separator would set the pressure, and an ejector would set the mass flow rate ratio between suction inlet and driving inlet. Also internal heat exchangers do not cause problems, because the predefined heat flow rate decouples the two fluid pipes. If a system consists of multiple connected vapor compression cycles and/or heating/cooling liquid cycles, this simple equation system decouples the cycles and the above shown computational causality can be applied to each cycle.

The level of abstraction of the presented approach is higher than in other publications such as (Casella et al., 2011), many relations have been replaced completely by predefined values, not only by linear relations.

## 4 Loop Breaker Component

A vapor compression cycle consists of at least 4 components connected to a cycle. If dynamic models are used, then it is no problem to connect these components to a cycle. If only steady state models are used, then the mass balance causes circular dependencies. Each component has a mass balance that basically sets the outlet mass flow rate equal to the (negative) inlet mass flow rate:

$$\dot{m}_\mathrm{A} = \dot{m}_\mathrm{B} \tag{3}$$
$$\dot{m}_\mathrm{B} = \dot{m}_\mathrm{C} \tag{4}$$
$$\dot{m}_\mathrm{C} = \dot{m}_\mathrm{D} \tag{5}$$
$$\dot{m}_\mathrm{D} = \dot{m}_\mathrm{A} \tag{6}$$

This equation system is singular. The equation $\dot{m}_\mathrm{A} = \dot{m}_\mathrm{A}$ can be derived, and no value is set.

To solve this problem using pure steady state models, an additional component called loop breaker is used. This component does not have a mass balance and is therefore underconstrained. The circular dependency is no longer a problem. However, when initializing a dynamic model in steady state, the mass balance cannot be removed. Only for the initialization phase the circular dependency has to be broken.

The locally overdetermined equation system for the mass flow rates can be brought into a balanced form, if one degree of freedom is added to the this part of the initialization problem. This degree of freedom has to be set by the mass balance equations. There are only a few ways to add a degree of freedom to the initialization equation system without changing the continuous time equation system:

1. A parameter with `fixed=false`
2. A discrete state variable
3. A continuous time state variable with `der()=0` as continuous time definition

One possible implementation for a loop breaker can be interpreted as a junction model. Starting point is a component model with one inlet connector and one outlet connector. In this model an additional mass flow rate `mDot_loopbrk` is added to the mass balance. If this component is integrated to the closed cycle of steady state components, `mDot_loopbrk` can be calculated. As no mass is added in the other components, no mass will leave the loop breaker component, therefore `mDot_loopbrk` is equal to zero:

```
  parameter Real mDot_loopbrk(fixed=false);
initial equation
  der(density)=0;
equation
  mDot_in + mDot_out + mDot_loopbrk =
    volume*der(density);
```

It is important to notice, that the whole initialization problem is balanced. It is only locally overdetermined. So by adding a degree of freedom another initial equation can be added.

The presented approach is similar to the one used by (Casella et al., 2011).

## 5 Separator Model

The separator model has to be treated different from the other models. Generally the separator has two main purposes. First, it separates liquid from vapor in a normal operating condition. Second, it is used to store refrigerant without changing the state of the system. This component is also special because its state cannot be calculated from the constraint `der()=0`, because in normal operating condition, the filling level of this component does not influence the outlet state (pure liquid or vapor). An additional information about the initial filling level or the total mass in the system is required. The additional degree of freedom added for the mass balance loop breaker is used for this purpose. So actually there is a `der(h)=0` initial equation, and a `fillingLevel=initialFillingLevel` initial equation.

The above presented approach to use predefined heat flow rates may lead to another problem: The sum of the predefined heat flow rates and powers might not sum up to 0. This is by definition not a steady state, since more (or less) heat is put into than taken of the cycle. This circular dependency is not properly detected by Dymola, rather the problem is solved numerically. If the energy balance is fulfilled numerically, then the solution can be found. If the energy balance is not fulfilled, then evaluating the simple solution fails. Similar to the mass balance, the system of equations is not in a locally balanced state.

To overcome this overdetermined energy balance and to smooth the transition from simple to actual solution, an additional degree of freedom is added to the separator model: An energy balance loop breaker. Similar to the mass balance loop breaker, an additional variable is added to the balance equation, and it is calculated from all initial conditions - namely the `der(h)=0`. So if the heat flow rates sum up to zero, then the simple equation system result for this energy flow is zero. But if the heat flow rates are unbalanced, then this additional energy flow is equal to the energy balance error for the simple equation system.

For this additional degree of freedom an additional initial equation has to be added. But this initial equation should not be used to define this degree of freedom, but rather set something else which had not been defined yet: the total pressure level. Up to now the pressure is not yet defined, only pressure difference due to the valve characteristic. Of course in the actual equation system the energy balance loop breaker variable should be zero.

The following initial equation has been chosen:

```
homotopy(deltah_loopbrk, k*(p-pInitial))=0;
    k=1e-2;
```

The loop breaker variable `deltah_loopbrk` with the unit $[J/kg]$ is set to zero as actual solution, and for the simplified solution the pressure is set to a fixed value. $k$ is used to define the transition shape between pressure difference and energy boundary condition. As a result of this the energy balance does not have to be fulfilled for the prefixed user values in the simple equation system. However, in the actual equation system the loop breaker enthalpy difference is zero.

## 6 Specific Enthalpy Breaker Models

Similar to the separator model, it is useful to add additional breaker models, to define the fluid state i.e. specific enthalpy at certain positions. This is possible by adding more degrees of freedom to the initialization problem which disappear at simulation time.

For example the superheat after the evaporator is often controlled to a constant value. So assuming the controller will be successful, the fluid state at that position is known and can be set to a constant value for the simple solution. The enthalpy difference `delta_h` in the model has to be 0 for the actual solution, and for the simple solution it has to be calculated from the constraint `portB.h_outflow= superheatedEnthalpy`. `superheatedEnthalpy` is the enthalpy difference to the saturated enthalpy. So one possible implementation is:

```
Real superheatedEnthalpy=...;
```

```
  parameter Real delta_h(fixed=false);
initial equation
  0 = homotopy(delta_h,
      portB.h_outflow-superheatedEnthalpy);
equation
  portB.h_outflow = inStream(
      portA.h_outflow) + delta_h;
```

## 7 Controller

In the presented simplified solution, the superheating after the evaporator depends on the mass flow rate, but the outlet specific enthalpy does not. So the solver will try to find a low pressure value, that has the desired superheating temperature for a given specific enthalpy. This equation system is hard to solve and often has zero or two solutions.

However, the controlled variables can be changed for the simple solution. Instead of passing the actual superheating temperature to the controller, a pressure difference to a desired low pressure value can be passed to it (of course the setpoint of the controller has to be added to this difference). A simple model to modify the measured signal could look like this:

```
  parameter Real replacement_desired = ...;
  parameter Real original_setpoint = ...;
  parameter Real k = 1e-6;
  RealInput u "original value";
  RealInput replacement_measured;
  RealOutput y = homotopy(u,
      original_setpoint + (
      replacement_measured -
      replacement_desired)*k);
```

The `replacement_desired` is the desired low pressure, `replacement_measured` is connected to the current low pressure. `k` is used to relate the order of magnitude of a pressure difference to a temperature difference.

Similar to this the capacity controller in the system has to be modified. The controller usually modifies the compressor displacement and consequently the mass flow rate. The cooling capacity is fixed for the simple solution and so is the air outlet temperature. If the controller is not modified, then the output will be limited and the integral part will may be defined by the anti-windup implementation. This issue can be fixed similarly by replacing the measured air outlet temperature with an homotopy term that depends on the controller output itself. So using the above described model the `replacement_desired=0.9` is the desired relative displacement, `replacement_measured` has to be connected to the current relative displacement. `k` is set to 10.

## 8 Software Experimental Results

In the following a common R-134a automotive vapor compression cycle is examined. The system is shown in figure 5. The cycle has a valve, an evaporator, a compressor, and a condenser with separator and build-in subcooling section. The superheating after the evaporator is used to control the expansion valve (superheating setpoint 7 K). The evaporator air outlet temperature is used to control the relative displacement of the compressor (air temperature setpoint 3°C). The whole system including the controllers is initialized in steady state. The evaporator air inlet temperature and the condenser inlet air temperature are 30°C. The compressor speed is set to 50 Hz (=3000 rpm). All results have been calculated using Dymola 2019.

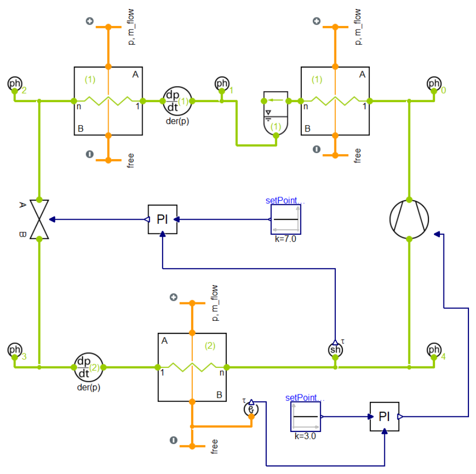

**Figure 5.** Automotive air conditioning cycle used for simulative experiments, based on TIL. The condenser is implemented using two separate heat exchangers.

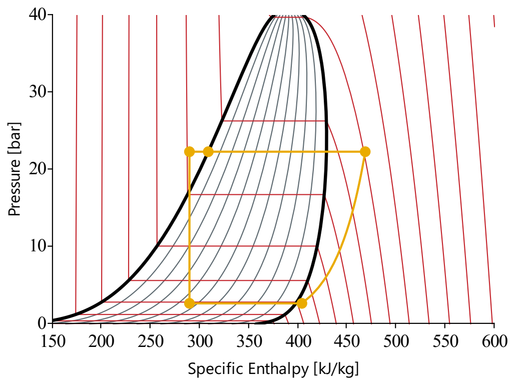

**Figure 6.** Steady state of the automotive air conditioning cycle used for simulative experiments shown in the ph diagram.

The predefined values in the simplified equation system are:

- High pressure: 25 bar
- Mass flow rate: 0.05 kg/s,
- Condenser heat flow rate: 7000 W condensation + 1000 W subcooling
- Evaporator heat flow rate: 6000 W
- Compressor power: 2000 W

- Nominal linear valve characteristic: mass flow rate = 0.05 kg/s at 24 bar pressure difference
- Setpoint for superheat controller replacement (low press.) = 1 bar
- Setpoint for capacity controller replacement (rel. disp.)= 0.9

The predefined values are enough to replace all initial and start values in the model, if the system is initialized in steady state.

The robustness of the solving procedure is influenced by two aspects:

1. Plausibility of the simplified nominal working state.
2. Similarity between the simplified and nominal operating condition.

## 8.1 Transition from Simplified Nominal System State to Actual System State

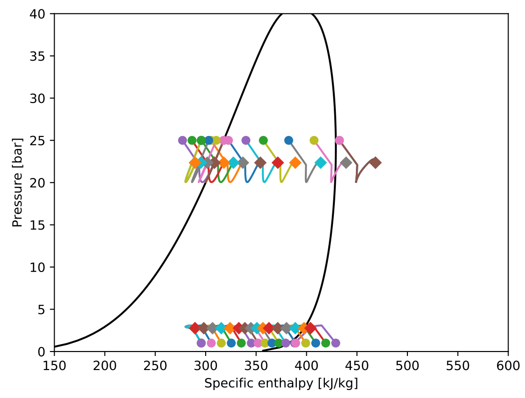

**Figure 7.** Transition from simplified nominal system state to actual system state. Each marker represents a thermodynamic state in a control volume of the heat exchanger. Circles mark the simplified state. diamonds mark the actual state.

Figure 7 shows the transition between the simplified and the actual control volume states in a ph-diagram. It is clearly visible that the transition of the states is continuous, but the shape of the transition is not linear. There is no simple explanation for transition form. In the simple equation system the pressures are predefined, in the actual equation system the pressures are defined by the refrigerant mass in the components, the temperatures, and heat transfer to the other medium.

As mentioned before, it is important that all enthalpy and pressure states remain in a reasonable range during the transition. Otherwise the fluid properties would cause problems. E.g. it is not possible to provide reasonable property data for a negative pressure or if temperature are below the triple temperature. As can be seen the transition stays well within a reasonable range. Pressures stay below the critical point and above the triple point. The specific

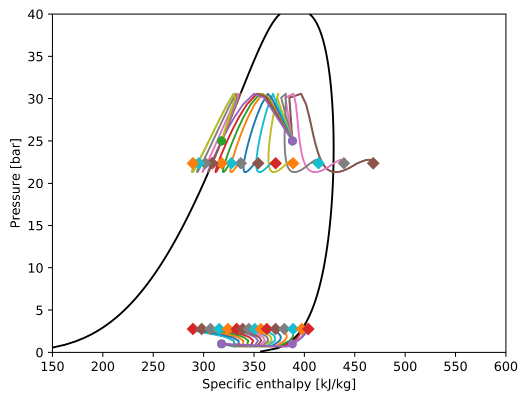

**Figure 8.** Transition from simplified nominal system state with zero heat flow rate to actual system state.

enthalpies stay around the two phase region. If the simplified nominal working state is not close to the actual state, this does not seem to be a problem.

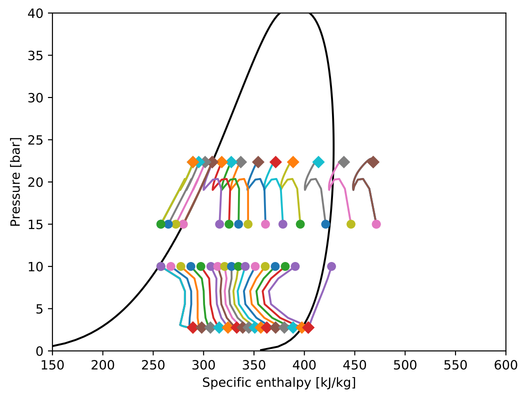

**Figure 9.** Transition from simplified nomnial system state with low pressure difference to actual nominal system state.

Even the zero heat flow rate use case shown in figure 8 is working fine, although the transition is nonlinear. The pressure levels of the simplified nominal working state seem to be very important for the plausibility of the system state as is visible in the nonlinear transition in figure 9.

## 8.2 Robustness against Operating State

Often a normal transient simulation from an arbitrary initial state is done to find the steady state of a dynamic model. Homotopy method is an alternative to that. To enable a fair comparison, the simulation times for both cases have been tested. In this section the results of a batch run for different boundary conditions are discussed. The operating states of the system are shown to illustrate the robustness. The following conditions have been varied:

- compressor speed: 10 Hz to 50 Hz (600 to 3000 rpm)
- condenser air inlet temperature: 10°C to 60°C
- evaporator air inlet temperature: 10°C to 50°C

All the dynamic simulations start from the same initial state. The simulation stop time was set to 1000 s, DASSL was used with a tolerance of 1e-4. To increase the chance of convergence, the supheating controller has a comparably low gain value. The results calculated with homotopy method are all based on the same nominal state ("normal"). The model was only initialized with homotopy at $t = 0$s, but not simulated. The total number of cases is 60.

To examine how these results depend on the model size, the same batch run has been done with different levels of dicretization. "2 x volumes" indicates, that the number of control volumes in the heat exchangers has been doubled.

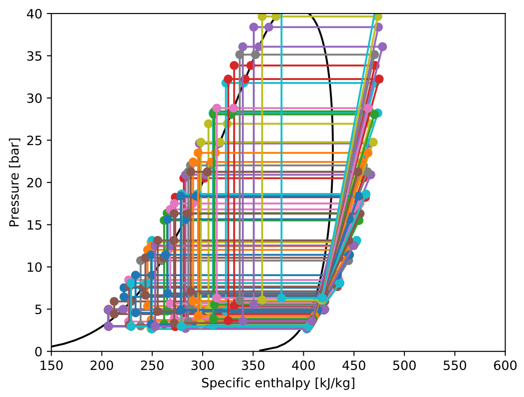

**Figure 10.** Steady state results of the cycle of the parameter variation. The results vary from very low pressures and temperatures up to the critical point.

Figure 10 shows the steady state results of the batch run. In all cases the superheating was reached, and the filling level of the separator was around 40 to 50%. So the system was always in a normal operating condition, which simplifies the simulation. The pressures are ranging from 2.5 bar to 43 bar.

For a wide range of operating states the above presented homotopy method is capable of finding a steady state. It proved to be very robust and easy to parametrize. However, if the controller limits would be active, the picture might change.

## 8.3 Computational Effort

The batch run discussed in section 8.2 is now examined regarding its CPU time and function evaluations. The measurements were taken on an i7 (2. gen), each calculation as been executed 5 times to get an average execution time.

In figure 11 the CPU times for the different cases are shown. For the three variations (three levels of dicretization) of the system homotopy method is computationally less expensive than a simulation by a factor of 5-10.

Since the CPU time is difficult to measure precisely and is highly dependent on the CPU an additional indicator was chosen: For the dynamic simulations the number of F-evaluations (evaluations of the RHS of the hybrid ODE), and for the homotopy method the number of evaluations of the residual of the initialization problem. These measures

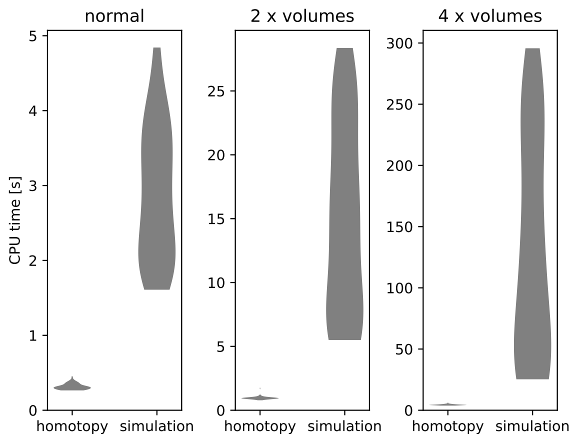

**Figure 11.** CPU time of a parameter variation compared between simulation with Dassl (tolerance = 1e-4) and homotopy initialization. "2 x volumes" indicates that the number of control volumes has been doubled compared to the normal example. The thickness of this violin plot indicates the density of occurrence.

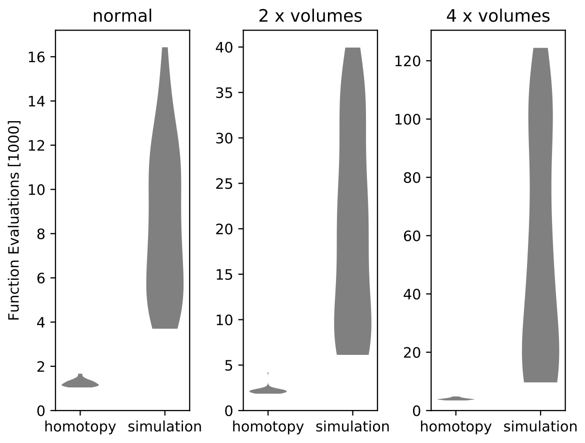

**Figure 12.** Number of function evaluations of a parameter variation compared between simulation with Dassl (tolerance = 1e-4) and homotopy initialization. "2 x volumes" indicates that the number of control volumes has been doubled compared to the normal example. The thickness of this violin plot indicates the density of occurrence.

usually are proportional to the CPU time. The results are shown in figure 12.

The two figures look very similar, so the CPU time has been measured with a sufficient precision. But comparing the simulation with the homotopy shows that a RHS evaluation of the ODE is not equivalent to a residual evaluation of the intialization problem. The F-evaluation is computationally more expensive.

However, larger parameter studies and user tests have to be done to clearly evaluate the benefits from using homotopy for different initial states, simplified nominal working states, and larger systems.

## 9 Conclusion

The simplified equation system to describe a vapor compression cycle that was presented in this paper is easy to parametrize, and defines a reasonable system state. The simple model is very abstract but it particularly enables separation of different flow paths and different cycles, so it is potentially able to handle large scale problems, even though this still has to be proven.

The mass and energy balance require a loop breaker to handle the different causality of the simplified model. The energy balance loop breaker turned out to have a positive influence on the convergence. Measured values for controllers have to be modified the define the integral part.

The experiments show that initialization using the presented approach is very robust, and neither the operating state of the system, nor the boundary conditions have to be close to the simplified solution.

Homotopy method lead to a reduction of the computational effort. The transition of the system state including the controllers have a huge impact on the result. A bad homotopy implementation is likely to fail or be computationally more expensive. More time has to be invested to evaluate the user-friendliness, robustness, and computational speed also for other cycles.

## 10 Acknowlegdements

This work has been partially supported by the BMBF in the VEOTOP project (FKZ: 01LY1809A).

## A Appendix

Code for a simple homopty example:

```
model HomotopyPumpLine
  Real V_flow(start=0);
  parameter Real dp0=6;
initial equation
  der(V_flow) = 0;
equation
  der(V_flow) =
    1*(
      6.5 - homotopy(
        actual = (dp0 + 3*V_flow - (3*
            V_flow-0.5)^2*sign(3*V_flow-0.5
            )),
        simplified = (2*dp0 - 10*V_flow)
        )
      );
end HomotopyPumpLine;
```

In Dymola the flag `Advanced.OnlyUseHomotopyMethod` has to be activated to find the rightmost solution. If it is not activated, homotopy is only used if the default algebraic solver fails. If the flag `Advanced.DebugHomotopy` is activated, a csv file with the `V_flow` over lambda will be generated in the current working directory.

## References

Francesco Casella and Alberto Leva. Modelling of thermo-hydraulic power generation processes using modelica. *Mathematical and Computer Modelling of Dynamical Systems*, 12 (1):19–33, 2006. doi:10.1080/13873950500071082.

Francesco Casella, Michael Sielemann, and Luca Savoldelli. Steady-state initialization of object-oriented thermo-fluid models by homotopy methods. 2011. doi:10.3384/ecp1106386.

Friedrich Gottelt, Timm Hoppe, and Lasse Nielsen. Applying the power plant library clara for control optimisation. In *Proceedings of the 12th International Modelica Conference, Prague, Czech Republic, May 15-17, 2017*, number 132, pages 867–877. Linköping University Electronic Press, Linköpings universitet, 2017. doi:10.3384/ecp17132867.

M. Gräber, K. Kosowski, C. Richter, and W. Tegethoff. Modelling of heat pumps with an object-oriented model library for thermodynamic systems. *Mathematical and Computer Modelling of Dynamical Systems*, 16(3):195–209, 2010. doi:10.1080/13873954.2010.506799.

Sylvain Quoilin, Adriano Desideri, Jorrit Wronski, Ian H. Bell, and Vincent Lemort. A modelica library for the simulation of thermodynamic systems. 2014. doi:10.3384/ECP14096683.

Christian Schulze. *A Contribution to Numerically Efficient Modelling of Thermodynamic Systems*. PhD thesis, Dec 2013. URL `https://publikationsserver.tu-braunschweig.de/receive/dbbs_mods_00057492`.

Michael Sielemann, Francesco Casella, Martin Otter, C. Clauss, Jonas Eborn, Sven Erik Mattsson, and Hans Olsson. Robust initialization of differential-algebraic equations using homotopy. 2011. doi:10.3384/ecp1106375.

Verein deutscher Ingenieure. *VDI-Wärmeatlas*. VDI-Buch. Springer, Berlin Heidelberg, 11 edition, 2013. ISBN 978-3-642-19981-3. doi:10.1007/978-3-642-19981-3.