APPENDIX H

Simulation performance and other contributions

Abstract

This appendix gives a short overview and description of the work performed by Modelon in the OpSys project. Modelon contributed in defining and setting up a pyFMI interface for communication between a real heat pump and a house model. Modelon also provided a first principles heat pump model written in Modelica, but which proved to be too slow for seasonal simulation and controls. In addition to minor modeling support within the project, Modelon focused latter part of the project on simulation performance improvement of the mentioned heat pump model and came up with a spline based method for thermodynamic property functions.

Jens Pålsson, Katrin Prölss, Lixiang Li, Pieter Dermont
Modelon

Table of Contents

1 Introducing pyFMI

pyFMI is an open-source package intended for the simulation of FMU's (Functional Mock-up Unit) in python. FMU are dynamic models created through the open standard FMI (Functional Mock-up Interface, http://fmi-standard.org). Many tools provide FMU simulation capabilities, however pyFMI is a license free framework and was preferred over a commercial tool due to its lightweight and reliable capabilities for model simulation at each communication point, over extended time periods.

The FMU represents a dynamic house model, about which more information can be found in Appendixes C and D. It is a co-simulation FMU, i.e. the solver is included with the model in the FMU. The pyFMI script links the house model – through a Modbus interface – to the OPSYS test rig with a physical heat pump (Appendix A). The HiL (Hardware-in-the-Loop) system created permits to assess the real 'seasonal' heat pump COP as opposed to the steady state COP provided by the manufacturer.

The work carried out consisted of scripting a robust simulation and communication framework. The script starts with requesting the inputs to the model from the heat pump, followed by a model simulation, the outputs of which are subsequently communicated back to the heat pump. This includes sufficient and appropriate error handling capabilities. The set-up with pyFMI showed to be technically feasible and all communication was handled as required.

2 Requirement specifications

The Vapor Cycle Library, a model library written in Modelica by Modelon targets the area of vapor compression and Organic Rankine cycles for system and control design as well as component integration. It was used to set up a heat pump model based on first principles for the system investigated in Appendix D. The model turned out to be too complex for the targeted simulation times slowing down the overall system simulation significantly. Especially the following aspects were identified to require further work to use the heat pump model in the proposed time horizon including controls;

1. Refrigerant medium properties. A lower accuracy in favor of higher performance is desirable. See also the following section X.3
2. The heat exchanger models (evaporator and condenser) ask for geometry parameter input by the user, such as heat transfer area, hydraulic diameters and solid masses. This information is often not available when developing a general control strategy or looking at system architecture. Further work is needed on heat exchanger with low discretization which supports parameterization with a given performance specification instead of actual design parameters.
3. The heat pump model needs to simulate fast even when the compressor is turned off and the refrigerant flow goes to zero eventually. The model approach in the library suffer often from bad damping at low or zero flow rates and are therefore sensitive against numerical disturbances.

3 Medium properties in vapor compression cycles

In addition to models describing the dynamic behavior of the components in the process the Vapor Cycle Library contains also fluid property models for the used refrigerants and working fluids. The calculations are based on the Helmholtz function, which is usually formulated as high order polynomials, implemented in Modelica as well. The component models define pressure and enthalpy as continuous time states, which means that they are given as inputs to the property calls. Most computationally expensive are iterations required to transform the numerical states to inputs of the **Helmholtz** equation, density and temperature.

A number of nested function calls are necessary to deliver all required thermodynamic and transport properties including phase boundary information. The initial implementation of the fluid properties present in version 1.2 of the Vapor Cycle Library (VCL) yielded a rather slow performance, which was continuously analyzed and improved during this effort, targeting mainly the order of execution and iteration start values. It was possible to decrease CPU time by a factor 3 with later version of VCL as can be seen in Figure 1.

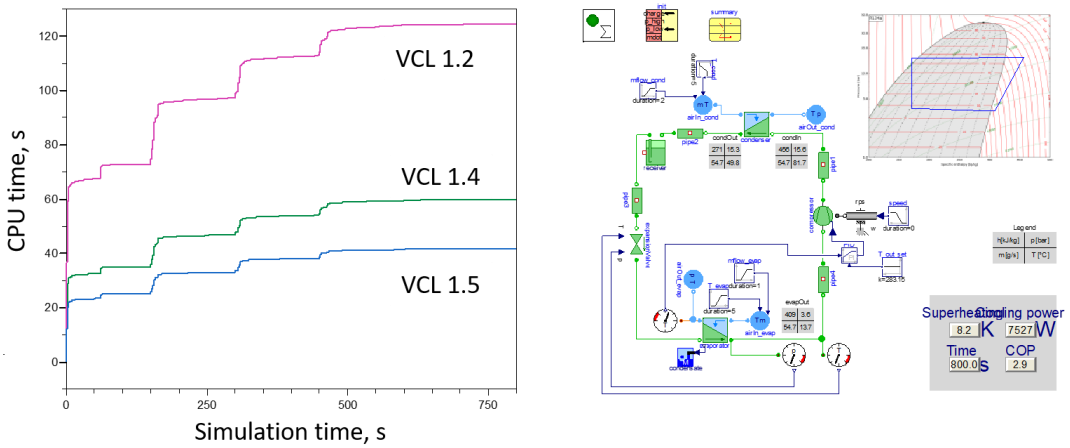

Figure 1: CPU time reduction with improvements in the fluid property routines, here for R134a and different VCL versions. The left plot shows the CPU time on a standard laptop vs. simulation time for the refrigeration system to the right with transients imposed at the system boundary.

A faster property calculation method, based on lower order piece-wise splines, is a neccessity when facing more complicated cycles and more advanced control/optimization development. Such method should satisfy the following requirements:

1) (At least) continuous 1st derivatives
2) Consistent forward and backward (inverse) calculations
3) Analytic inverse function to avoid iteration
4) Consistency across the phase boundary to avoid chattering
5) Analytic Jacobians to speed-up solution of non-linear systems

A new spline-based table look-up (SBTL) method [1] satisfies all the above attributes and has already been adopted as part of the International Association for the Properties of Water and Steam (IAPWS) guidelines. Significant speed-up by application of this method has been observed in both CFD simulations for steam turbine and also dynamic simulations of different thermodynamic cycles. We developed a fast property model for two-phase fluids based on the SBTL method and summerized our development effort and results in this chapter.

3.1. Methodology

The SBTL method uses piece-wise quadratic splines (bi-quadratic for 2D) [2] [3] to approximate the properties derived from equations of state. Spline coefficients, instead of raw data, are stored in files and retrieved for evaluation. In the models we developed, the Helmholtz based models are used to generate the source data. Since the spline is a second order one, it's fast to evaluate and it can be inverted analytically if the function is monotonic. This is usually the case for what we need in thermodynamic cycle simulations (pairs of properties like temperature and enthalpy at fixed pressure, entropy and enthalpy at fixed pressure etc.). The interpolation scheme also ensures C1 continuity.

Another key feature of the SBTL method is the usage of an equidistant grid that avoid searching when evaluating the spline. The original publication divided the whole domain into several sub-domains to achieve high accuracy for water and steam with minimum number of grid points. For our purposes, one equidistant grid without sub-division is good enough. Note that while having more grid points to improve accuracy does not increase the cost of computation, it does make the data file larger (for 2D splines, file size $\sim O(n_{grid}^2)$) and takes more time to load at initialization.

In thermodynamic cycles simulations, chattering behaviors can happen if property calculation is consistent across the phase boundary. In our work, the phase boundary was defined by $T_s(p)$, a spline for saturation temperature as a function of pressure. Temperature, density and entropy were interpolated as 2D splines as functions of pressure and enthalpy in the single-phase region. Proper extrapolation was done for these properties to make sure that they are smooth near the phase boundary.

To implement the SBTL scheme, property data was generated using Helmholtz based model from the Modelon Base Library (a base library containing common features, components and functions). A Python script, developed based on [2] and [3], then solved for the spline coefficients and stored them in files. The script also evaluated the accuracy of the generated splines and checked if the grid size was

appropriate. Finally, the coefficient files were used by the thermodynamic property model written in Modelica, via an external object in C. Figure 2 shows the structure of the property model.

Figure 2 Structure of a thermodynamic property function call in the SBTL model in Modelica

More theoretical details about the STBL method can be found in [1] and computer algorithms to solve for the spline coefficients are provided in [2] and [3] in FORTRAN code.

3.2. Results

In this section, comparison was made between a Helmholtz-based model and the SBTL model for R134a. We first looked at CPU time of a single function call, then we used the medium in an evaporator simulation. The Helmholtz based model was taken from the Modelon Base Library (*Modelon.Media.PreDefined.TwoPhase.R134aRef*). Computer configuration is shown in Table 1.

Table 1 Configuration of the computer used for testing

Model	Dell Precision M2800 Laptop
Processor	Intel® Core™ i7-4810MQ CPU @ 2.8 GHz
RAM	16.0 GB
System	64-bit, x64 based, Windows 10 Pro
Software	Dymola 2018
C compiler	Visual Studio 2012 Express Edition (11.0)

Few functions were tested individually and results are listed in Table 2. In dew enthalpy test, pressure input ramped from 0.3 bar to 39.5 bar. For temperature and density derivatives, the enthalpy ramped from 150kJ/kg to 500KJ/kg with pressure fixed at 0.3, 0.5, 1, 2, 5, 10, 20 and 39.5 bar.

Table 2 Results of individual function tests

Property	Deviation	CPU time Helmholtz	CPU time SBTL	CPU Time reduced
Dew Enthalpy	<0.2%	3.6e-5s	6e-7s	98.3%
Temperature	<0.3%	>2.7e-5s	<3.1e-6s	>88.5%
$\partial\rho/\partial h\|_p$	N/A[1]	>2.8e-5s	<3.5e-6s	>87.5%

[1] Phase boundary locations are slightly different in the two models, makes it hard to compare the results in terms of percentage deviation.

The dew enthalpy test is representative of saturation properties, which are splines both in the Helmholtz-based model and the SBTL model. The temperature function shows the speed of computation for density, entropy etc. The test for partial derivative of density demonstrates the speed-up of calculating partial derivatives, e.g. specific heat capacity, isobaric expansion coefficient etc. Overall, we see a significant improvement in computational performance by using the SBTL model with only a small deviation in the results.

System models for heat pumps usually consist of thousands of equations with complicated numerical structures produced by symbolic manipulations. Hence, further proof of concept, beyond property function tests, is required to evaluate the performance of the SBTL model in system level simulations. In a full cycle, the heat exchangers usually have the highest number of property function calls and take up a large part of the computational cost. Hence, a heat exchanger simulation is a great test case to justify the effort to develop a complete SBTL model for heat pump systems.

The test model is directly taken from Modelon's Heat Exchanger Library (*HeatExchanger. HeatExchangers.FlatTube.Experiments.Evaporator*), see Fig. 3. It's a test bench for discretized cross-flow evaporator models. Downstream pressure of the air is atmospheric pressure and that of the refrigerant (R143a) is 4.85 bar. Inlet mass flow rate of air is 0.3kg/s while that of the refrigerant is 0.04 kg/s. The air inlet temperature to evaporator is 30° C, whereas the refrigerant enters at an enthalpy of 270 kJ/kg.

Dynamics of the simulation was driven by the changes in the boundary conditions: air mass flow rate ramped from 0.3kg/s to 0.5 kg/s at t=10-20s and the refrigerant mass flow rate went from 0.04kg/s to 0.05kg/s at t=20-25s. We ran the simulation for 100s to get to a steady state.

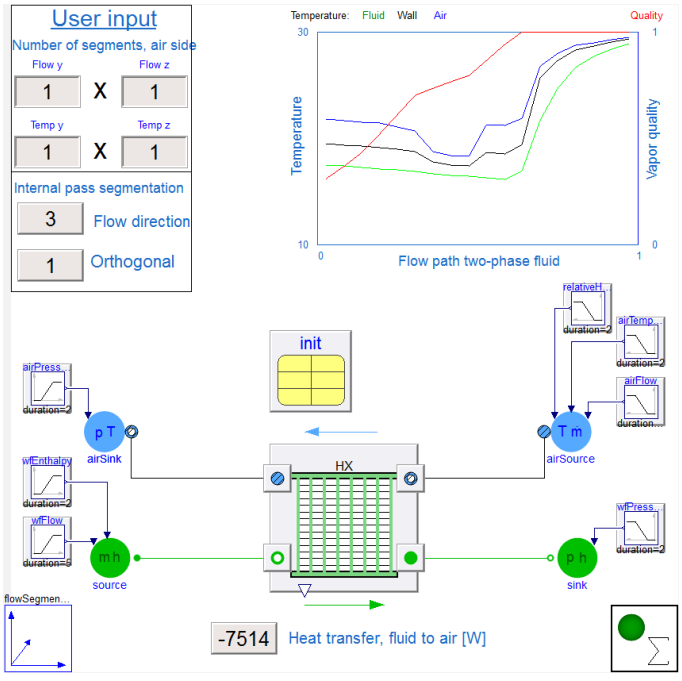

Figure 3 Evaporator test bench for comparison of different medium.

Figure 4 shows the comparison of cooling power and air outlet temperature from the two medium models. The new STBL model matched the Helmholtz based model very well for these two key variables.

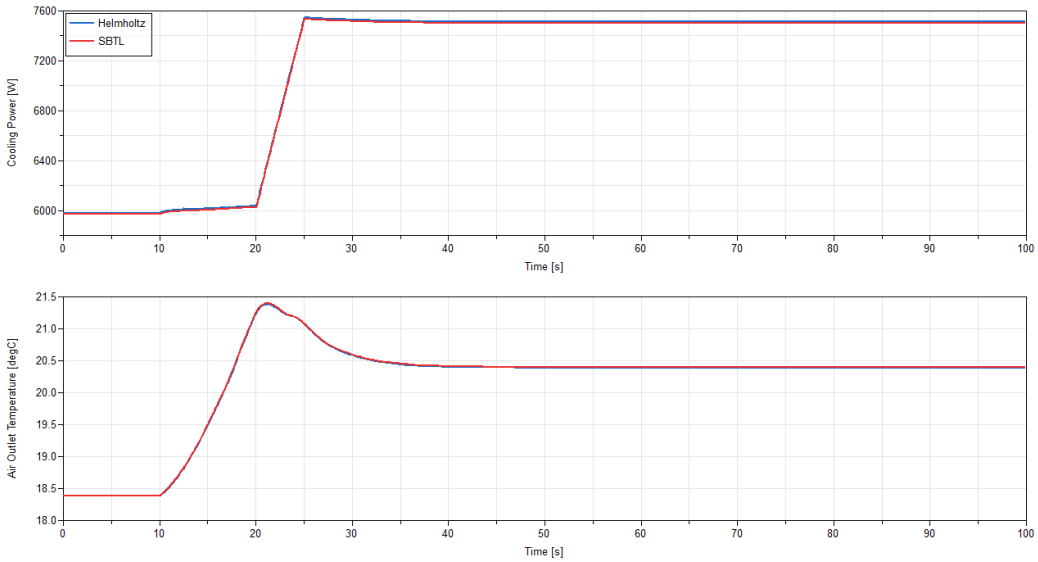

Figure 4 Cooling power and air outlet temperature of the evaporator

The CPU time of the evaporator tests is plotted in Figure 5. The run time after initialization, shown by the red curves, was reduced by 30% using the SBTL model. The initialization time was cut by more than 40% despite the fact that about 1s was spent on loading coefficient data.

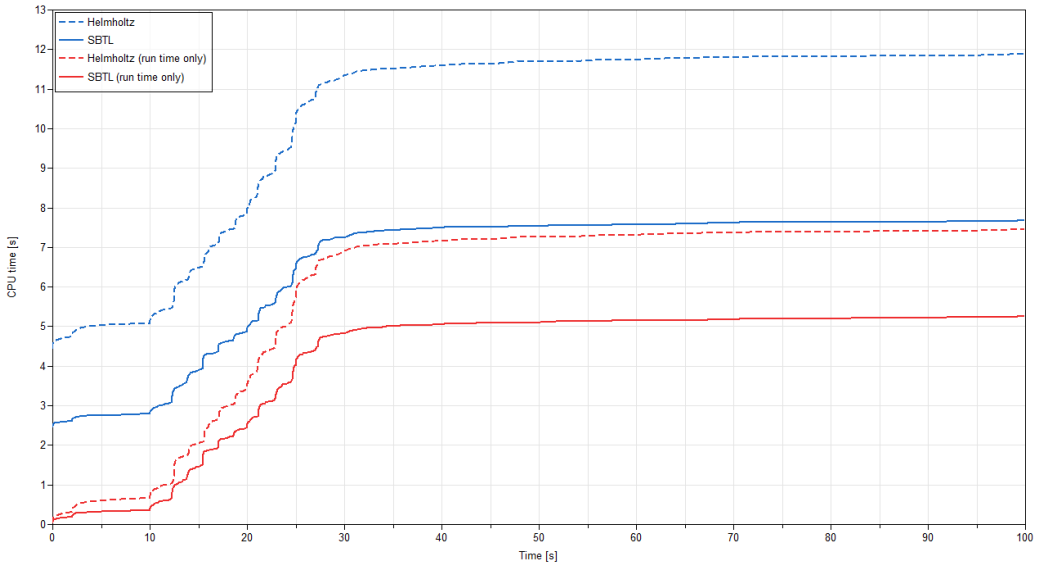

Figure 5 CPU time of the evaporator tests using Helmholtz based model and SBTL model

In both the individual function tests and the evaporator component tests, we observed a significant speed-up by using the SBTL property model with the same accuracy of traditional Helmholtz based model. For the next step, an inverse function, namely enthalpy calculated from entropy and pressure, needs to be implemented for the compressor in the heat pump system. More tests should be performed on different full thermodynamic cycles to further validate the accuracy, computational performance and robustness of the SBTL property model.

4. References

[1] Kunick, M., and H. J. Kretzschmar. *Guideline on the fast calculation of steam and water properties with the spline-based table look-up method (sbtl)*. tech. rep., The International Association for the Properties of Water and Steam, Moscow, Russia, 2015.

[2] Späth, H. *One dimensional spline interpolation algorithms*. AK Peters/CRC Press, 1995.

[3] Späth, H. *Two-dimensional spline interpolation algorithms*. AK Peters, Ltd., 1995.