Variable Structure Modeling for Vehicle Refrigeration Applications

Imke Krüger   Alexandra Mehlhase   Gerhard Schmitz
Hamburg University of Technology, Department of Technical Thermodynamics
Denickestr. 17, 21075 Hamburg
TU Berlin, Department of Software Engineering and Theoretical Computer Science
Ernst-Reuter-Platz 7, 10587 Berlin

Abstract

A variable-structure approach for Modelica models is presented in this paper. Variable structure models enable the user to change the simulation model during runtime. This is not supported by common simulation environments and thus a Matlab script is used to control the run of the simulation. The script switches between the different models and sets the initial values to ensure smooth transients of the variables. The method is applied to a simplified model of a thermal management system for Lithium ion batteries in a hybrid vehicle. In this model some components do not need to be calculated through the complete simulation time and are removed from the model through the variable-structure approach. With this approach the simulation time can be reduces while the simulation accuracy is not affected negatively.

Keywords: vapour compression cycle; simulation speed; thermal management, variable-structure model

1 Introduction

How can the variable-structure method help to speed up simulations? In the case of battery thermal management, the branch to the battery cooling can be opened or closed with a valve such that the battery is only cooled when needed. So the general structure of the refrigeration cycle changes from a branched cycle to a single evaporator cycle. In simulation environments supporting Modelica it is not possible to change the set and causality of an equation system. In Modelica it is assumed that a model always has one set of equations and that the variables themselves do not change. For the refrigeration cycle it would be highly useful to be able to change the equation system because the equations for the unneeded branch could be turned off. This means that no unnecessary calculations have to be done and the simulation time could be reduced. To explain this approach the thermal management of HEV batteries will first be explained. Then the general approach for variable-structure models that was used in this paper is introduced. The presentation of a simple model and its preparation for the application of the variable-structure method is followed by the results for simulations with and without the variable-structure method.

2 Thermal Management of HEV batteries

The batteries of hybrid electric vehicles heat up due to inner heat generation. Thermal management is though essential to ensure safety and prevent ageing. The only reliable heat sink for the cells is the automotive refrigeration cycle. The cells can be cooled by evaporation of the refrigerant, therefor a cold plate is put in parallel to the ordinary evaporator (see fig.1).

Cooling of the cells is only necessary when their upper temperature limit is reached. Only then the valve to the battery cooling branch opens, e.g. there is no refrigerant flow as long as the cells are cool enough. The necessary cooling power depends on the drive cycle and the surrounding temperatures.

System simulation plays an important role in the design of vehicle air conditioning. It enables the user to test various system architectures as well as providing values that cannot be measured in real life test rigs. As the development becomes faster and additional tasks like battery cooling emerge, accelerating the simulations becomes necessary. Additional components and more complex system designs raise the dimensions of the resulting nonlinear equation systems. During the evaluation process of a cooling system, a lot of simulations for various climatic conditions and heat loads

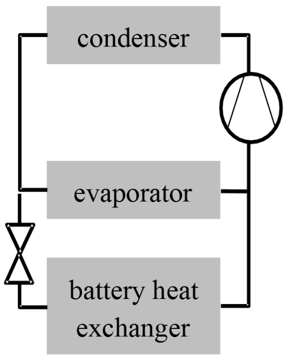

Figure 1: System architecture of A/C Refrigerant Cycle with Battery Cooling

are necessary to evaluate the additional energy consumption of the refrigeration cycle.

In the current Modelica models with static structure, the refrigerant mass flow in the battery branch cannot be set to exactly zero. The resulting very small refrigerant mass flows and pressure losses in the control volumes slow down the simulation. In addition, the mass flow might change its sign, causing further deceleration of the simulation. The equations for the closed branch have to be solved during the whole simulation although they are not needed most of the simulation time. The CPU time needed for simulations becomes too large and the number of possible simulation runs is limited by the available time. Very simple (and less exact) models have to be used, making the results less reliable.

Calculation time could be radically reduced if the obsolete equations could be switched off when the battery branch is closed. The time span during which the valve is closed can make up large parts of a driving cycle (see [1]) so there is a large potential to reduce the time for a simulation run. Currently there is no possibility to deactivate equations in Dymola/Modelica during runtime.

3 Variable-structure modeling with Modelica/Dymola

A variable-structure model consists of different modes whereas each mode itself is a model and has a set of equations and variables. The model can switch from one mode to another triggered through a switching condition. When a switching condition occurs the mode switch takes place and the end values of the simulation results are used to initialize the next mode. The modeler has to define which end values to use to initialize the next mode.

As explained above such a change of an equation system is needed to model the thermal management of batteries more efficiently. But neither Dymola nor other simulation environments e.g. OpenModelica and SimulationX support the change of a set of equations of a Modelica model during a simulation run. Therefore, a scripting approach with Matlab is used as introduced in [2]. This approach allows a user to model their models in a chosen simulation environment or language and use Matlab to switch from one mode (and therefore to another set of equations) to the next.

The general idea is to create a new modeling layer where the structural change is described and which handles the actual change. The simulation models are implemented in a simulation environment chosen by the user. It is important that the simulation environment can be controlled through Matlab so a model can be compiled and a simulation run can be started using Matlab.

Figure 2 illustrates the sequence of operations in a Matlab script that handles the change of a set of equations of a model. In this example the variable-structure model has two modes, which means we have two models whereas each model has its own set of equations. For this example we use Dymola as a simulation environment, but other environments could be integrated as well.

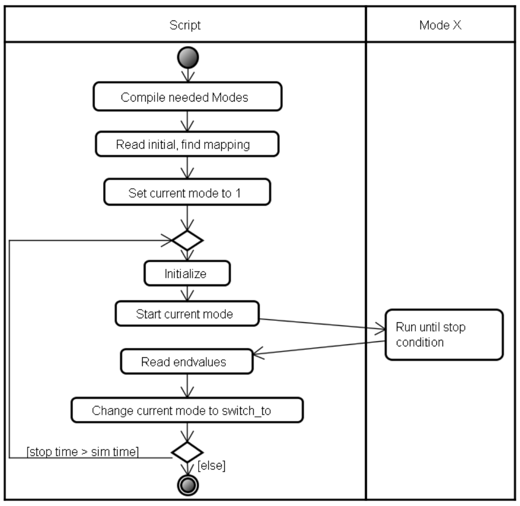

Figure 2: Course of events of a Matlab script to change a set of equations of a Dymola model

As a first step all needed models are compiled, which means an executable model 'dymosim.exe' with an initialization file 'dsin.txt' is created which can be started through Matlab. Afterwards the simulations parameters (start time, stop time, solver, etc.) are set.

Then a mapping of variables takes place. In this mapping process the initialization files of all modes are loaded. These files contain all variables and parameters with their startvalues. An equivalent file is created by Dymola at the end of each simulation of a model (called 'dsfinal.txt'), containing the endvalues of variables. When loading such a file we get an array with all variable names and an array with all initial (or end) values of a simulation.

For a mode transition between two modes it is necessary to map user defined variables from the end-array to the initial-array of the next mode. Therefore, a mapping matrix is created for each transition. This matrix holds the indices of the values to be read from the end-array in the first column and the indizees of the values to be overwritten in the initial-array in the second column.

This mapping matrix is created at the beginning because it saves simulation time when a transition is needed more than once, for instance switching from mode 1-> 2 -> 1 -> 2 would mean that the mapping matrix from mode 1 to mode 2 can be used twice. After this preparation phase is done the simulation of the first mode is started. The script uses the dymosim.exe which is created when compiling a Dymola model to simulate the model.

When a defined stop condition is reached, which is implemented in the model itself, a terminate command will stop the simulation. As soon as the simulation terminates, the end values of the simulation are read from the 'dsfinal.txt' file. The earlier created mapping matrix for this transition is then used to map the simulation results to the initial data in the dsin.txt file for the next mode.

The script then starts the dymosim.exe of the second model. This simulation runs until the stop time or another terminate condition is reached. Then the script again processes the simulation data and either the stop time of the simulation is reached which stops the simulation completely or the script changes back to the first mode via a transition and the mapping matrix in this transition.

With this simple approach Dymola can be used to simulate variable-structure models even though Dymola on its own does not support these kind of models. This means that existing Modelica models can be reused for variable-structure models and that they do not have to be remodeled in other tools or languages as SOL [3], MOSILAB [4] or Hydra [5] which do support variable-structure modeling to a certain extent. The problem with these approaches is, that SOL is an experimental language and does not support index reduction and solvers in the extent that Dymola does. MOSILAB does not support index reduction at all and is not freely available for it is still under development. Hydra is based on functional programming languages and is therefore not as easy to learn for modelers. All the existing approaches would mean a remodeling of the existing air condition models.

4 Evaluation

4.1 Model

All models in this use case are based on the AirConditioning Library by Modelon [6], based on the AC lib [7].

A simple test case was created to evaluate the variable-structure approach. It reduces the complex model of a refrigeration cycle with thermal management of the battery to the main components that are affected when the battery cooling branch is closed. The model consists of an evaporator with a discretized pipe in parallel, the branch to the pipe can be closed with a valve. The valve has a variable Kv-value that can be set by an input source. The original model with all initial equations activated (Figure 3) serves also as the initial mode for the variable-structure model. For the

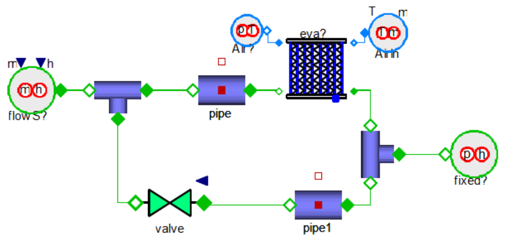

Figure 3: model for mode 1 with two branches

second mode (Figure 4), only the evaporator branch remains. Two new component models were created: splitResistance and junctionResistance. They represent the pressure loss in the corresponding components from the first mode. Using the same names for the components eases the mapping of the parameters and start values when a switch occures. This means at the beginning of the Matlab script where the mapping takes place, all components and their variables which have the same name (e.g. evaporator) are mapped in the mapping matrix of the transition.

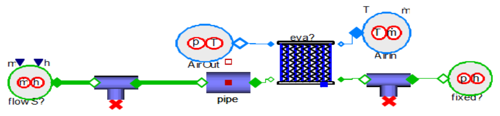

Figure 4: Modelica model for mode 2 with only one branch

The simulations are carried out as follows: The valve is closed with a ramp function, beginning at 30s simulation time. After 10s, the valve is closed. At 50s (10s after complete closure) the variable-structure model switches to mode 2. At 150s the variable-structure model switches back to mode 1, the valve opens again at 160s. Figure 5 shows the sequence for the kv-Value of the valve.

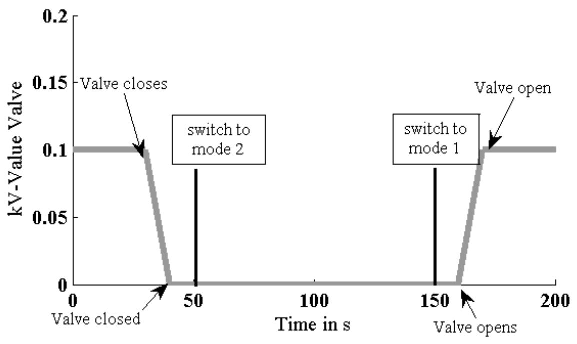

Figure 5: Simulation sequence for the kv-value of the valve

4.2 Preparation of the modes

As described above the mode 1 model has more components than the mode 2 model but each component from mode 2 has been in the first model, too. Therefore, it is known that the end values of the first model can be used to initialize all components of the second model. To make the identification of the components which exist in both modes easier, they were called exactly the same. Therefore, the mapping function called at the beginning of the Matlab script can create a mapping matrix which maps all variables existing in both modes to each other. To create the mapping matrix the script takes the lists of the initial names of both modes and searches through this list to match the names. This is necessary because the order of the variables might not be the same, even (sub-)variables of a component such as (evaporator.p[1], evaporator.[T]) might be in different order. The first mode has about 7 400 variables and the second mode about 6 900 variables which makes this mapping process time consuming. A better mapping algorithm is planned for future work.

If the variables and components are not called the same, the modeler can define which variables and components belong together. For instance the modeler can define that all variables from a component 'a' from mode one have to be matched to all variables from a component 'b' in model two. In this case all variables inside these components are matched and the mapping is saved in the mapping matrix.

To be able to initialize the second model through the Matlab script the model needs to be prepared for a script initialization. Many components in the AirConditioning Library are per default set to initialization through parameters and initial equations. This leads to the problem that an initialization from outside is not per default possible. For instance look at the following model:

```
model init
  Real T1(start = 100);
  Real T2;
initial equation
  T2 = T1-10;
  ...
end init;
```

In this example T1 has a start value of 100. But T2 can only be initialized through T1 and is 10 smaller than T1. With such a model we are not able to initialize the T's separately because we cannot ignore the initial equation even though this might be necessary if this model is the second mode and different values are needed. In the AirConditioning library such cases can be handled by setting the initType of the components to 'noInit'. If a component cannot be initialized externally and does not have such an initType provided the user has to change the model to use it in a variable-structure model. Often these initial equations are deep down in the model hierarchy, e.g. temperature of the evaporator wall and therefore it is quite difficult to locate all needed changes. This does not mean that initial equations are not allowed for variable-structure models, it just means that a modeler has to know what his model is doing and if the initial equations hinder an initialization from the outside.

An easy way to test if the model can be initialized though an extern file is to simulate a model for a period of time and use the end values from the 'dsfinal.txt' as initial file and restart the simulation of the same model.

If the simulation results are smooth around the mode change it is usually save to assume that the initialization worked.

A problem with the initialization through Matlab is that when using the given Matlab methods to handle Dymola the initialization does not always work. For instance, it is possible to use a method dymosim.m which gets as parameters the name of the Dymola model and the initial values (and some other data). But this method does not seem to write the dsin.txt (more precisely the dsin.mat as it is called from Matlab) correctly. This means the initialization does not work correctly and the simulation results are wrong.

First it was assumed that the initial equations in the AirCondition model were the problem but it was discovered that the given Matlab function seems to be the problem. Therefore a new initialization method in Matlab was written. This method uses the mapping matrix of the transition and creates a new initialization matrix for the new mode, which only holds the end data of the old mode and user defined values. All other data is not included in this initialization matrix. This new initialization matrix is then saved in the models dsin file. When switching back from the second mode to the first not all necessary data is known to initialize the mode. Therefore, the modeler has to specify the additional values separately in the Matlab script.

4.3 Results

The results of the variable structure model are conform to a large extent with the results of the static structure model, which calculates the unnecessary branch through the whole simulation time. The mass flow in and out of the split can be seen in figure 6. The valve starts to close at 30s, the mass flow changes according to the changing pressure drops in the branches. When the valve is completely closed, the mass flow into the battery branch is almost zero but shows still little variations around zero for the static structure model.

The variation of the mass flow results in variations of the pressure drop in the junction. Figure 7 and 8 show the inlet and outlet pressure of the evaporator. The little variations are thus propagated to all the components of the model, to the complex ones (in this example the evaporator), too.

The needed CPU time for the simulation runs is plotted in figure 9. The CPU time is given through Dymola and is the time from calling the dymosim.exe until exiting the simulation. Until the first switch of the variable-structure model, the CPU times rise with the same speed for both simulations. The second mode

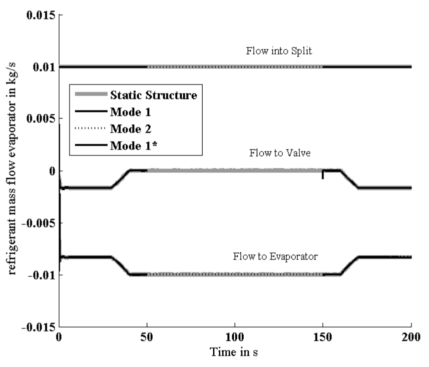

Figure 6: Refrigeration mass flow in and out of the split for static and variable structure model

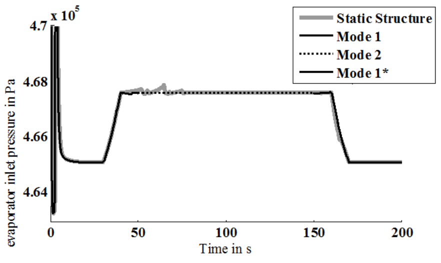

Figure 7: Refrigeration pressure at evaporator inlet

of the variable-structure model is calculated rather fast and does only need a short simulation time. While the variable-structure model does only need a short simulation time during this phase where only one branch of the model is simulated the static structure model needs a lot of simulation time. The opening of the valve at 160s lets the CPU time of the static model rise almost vertically. Whereas the variable-structure models CPU time does not rise that high. This already shows that while only simulating a short period of time (200 seconds), it is already possible to save a lot of simulation time through the variable-structure approach. In this example both modes where simulated for the same period of time.

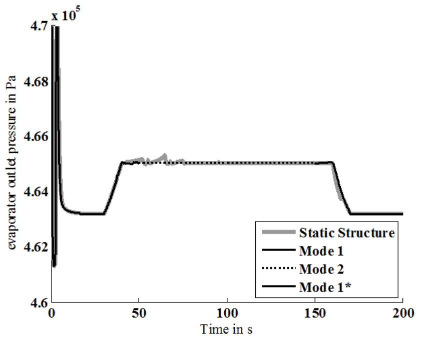

Figure 8: Refrigeration pressure at evaporator outlet

If, as was mentioned in the introduction, the two branched model is only needed for short periods during a drive cycle a lot of simulation time can be saved. But it also has to be considered that using a variable-structure model also means that a switching procedure is necessary and that each mode needs to be compiled. Compiling the two necessary modes of the variable-structure models takes about 22 seconds whereas the compilation of the static structure only takes about 13 seconds. Creating the mapping matrix for each transition at the beginning of the script take about 10 seconds – the search algorithm is not optimized yet and the time could be significantly reduced with a better algorithm.

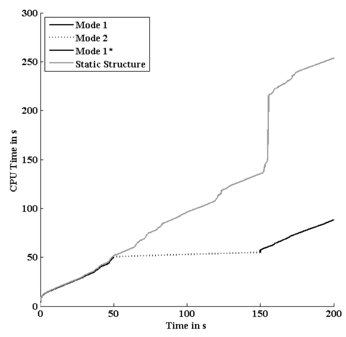

Figure 9: CPU time needed for simulation

The switching from one mode to the next with loading the end values and setting the initial values takes about 0.5 seconds per switch. This means that when looking at the overall simulation time of the presented example the variable-structure model needs about 121 seconds whereas the static structure models takes about 270 seconds. This means that even with the necessary overhead the simulation with the variable-structure model is still faster.

4.4 Restrictions

As already mentioned throughout the paper the variable-structure approach has some restrictions and the modeler has to regard certain points. At first the initialization needs to be mentioned, the modeler needs to define how the end values of one mode are used to initialize the next mode. If the old mode does not provide enough data for the initialization the modeler either has to provide the missing data (by concrete values or calculations) or the variable-structure approach might not be feasible. Furthermore, the models used as modes need to be initialized from the outside, so the modeler might need to adapt the models to fulfill this requirement.

As it also is with conventional modeling it is with variable-structure modeling, too, that one should not use it just because one can. For instance in conventional modeling a model with few equations might suffice even though one could model it more accurately which might result in solver problems or time problems. So it is with variable-structure modeling. In some cases the approach might be usable but not feasible, because the switch does not lead to a significant positive effect. For instance the simulation time is not reduced and the accuracy is the same. Another possibility is that the switch is done at the wrong time, e.g. switching to the second mode of the refrigeration application when the valve is not closed yet. This will lead to a model with inconsistent results. This means that, as in conventional modeling, the modeler needs to know his models and what he wants to do to use the variable-structure approach feasible and sensible.

5 Summary

System simulation for refrigeration cycle models in vehicle refrigeration applications is time consuming. The variable structure method presented in this paper can help to reduce the needed CPU time and the overall simulation time for such a model.

With the help of a Matlab script, the user can switch between several representations of the same model.

It takes some time to prepare and test the models. If the given advices are already considered during modeling the method can be easily used to speed up simulations.

The method can be applied to other applications with variable-structure with more than two modes, too. A Python framework which guides the user through the steps to describe a variable-structure model is currently worked on. This will enable the user to describe the models more easily and to use a free software (Python) instead of Matlab. Furthermore, more simulation environments will be integrated so the user is not limited to Dymola.

It is planned to investigate the advantages of variable-structure models more thoroughly and with more complicated models. With these researches it will be possible to find out when variable-structure models can be used sensibly and when it is more useful to use a static structure model.

References

[1] Krüger,I. Energy Consumption Of Battery Cooling In Hybrid Electric Vehicles. In: Proceedings of 14th International Refrigeration and Air Conditioning Conference, Purdue, USA, 16-19 July 2012 (to be published).

[2] Mehlhase A. Varying the level of detail during simulation. In: Proceedings of ASIM 2011, Symposium Simulationstechnik, Winterthur, Swiss, 7-9 September 2011.

[3] Zimmer D. Equation-Based Modeling of Variable-Structure Systems. Ph.D. thesis, Swiss Federal Institute of Technology, 2010.

[4] Nytsch-Geusen, C., Ernst, T., Nordwig, A., and et al. (2005). Mosilab: Development of a modelica based generic simulation tool supporting model structural dynamics. In: G. Schmitz (ed.), Proceedings of the 4th International Modelica Conference, Hamburg, March 7-8, 2005.

[5] Nilsson, H., Giorgidze, G. (2010). Exploiting structural dynamism in Functional Hybrid Modelling for simulation of ideal diodes. In: Proceedings of the 7th EUROSIM Congress on Modelling and Simulation. Czech Technical University Publishing House, Prague, Czech Republic, 2010.

[6] Tummescheid, H., Eborn, J., Prölß, K., AirConditioning - a Modelica library for dynamic simulation of AC systems, In: G. Schmitz (ed.), Proceedings of the 4th International Modelica Conference, Hamburg, March 7-8, 2005. vol. 1, p. 185-192.

[7] Pfafferoth, T., Schmitz, G., Modeling and transient simulation of CO2-refrigeration systems with Modelica, International Journal of Refrigeration, 2004, Vol. 24, no. 1, p.42-52.