Varying the level of detail during simulation

Alexandra Mehlhase
a.mehlhase@tu-berlin.de
TU Berlin
Ernst-Reuter-Platz 7 / 10587 Berlin

Abstract

The quality of a simulation model depends on the selected level of detail. If an inadequate level is chosen, the simulation will either lack accuracy or simulation performance. Often, it is not possible to find the optimal level of detail and a trade-off is necessary. To avoid such a trade-off it would be necessary to change the level of detail during the simulation. Such a change is not possible in modeling tools such as Matlab/Simulink or Dymola, because a change of the equation system would be required during simulation. In this paper we present a script that enables the user to specify a switch from one level of detail to another. With this approach a model of a diesel combustion engine can be simulated in less simulation time without significant accuracy losses.

1 Introduction

Models of physical systems need to become more detailed to fulfill the growing need for accuracy. Such detailed models usually take up more simulation time which is often not feasible. Often, a level of detail which satisfies the accuracy requirements and still simulates in an adequate time cannot be found thus compromises are necessary.

We regard variable-structure models as a possible solution to avoid such compromises. With a variable-structure model the physical equations that describe the behavior of the model can change during runtime. This means the model can switch from one mode with one set of equations to another mode with a different set of equations. Each mode itself is therefore represented by a model with a set of equations. There are two different application areas for variable-structure models:

- The behavior of the system changes: *variable-behavior model*
- The level of detail changes: *variable-detail model*

An example for a variable-behavior model is a pendulum that becomes a free-falling mass (see **Figure 1**).

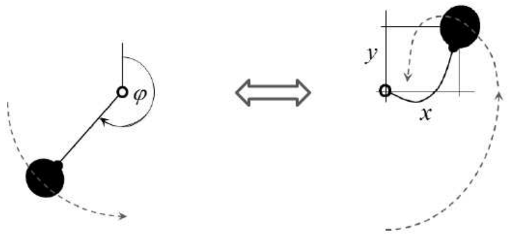

Figure 1: Pendulum becoming a free-falling mass

An example of a variable-detail model is a beam which can be modeled with two different levels of detail. When crossing a certain bending point (decision of the modeler) the model needs to be calculated more accurately (see **Figure 2)**.

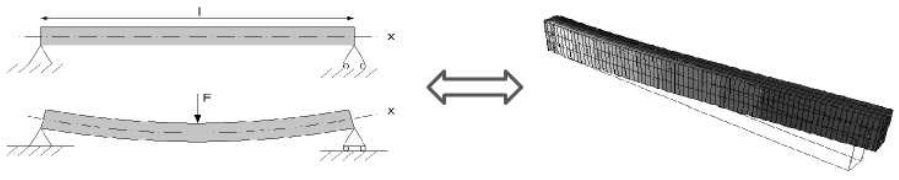

Figure 2: Bending beam with different level of detail

A variable-detail model could be simulated in more detailed form in critical situations and in less detailed form in less significant regions. Consequently, the variable-detail approach can make a simulation faster and more accurate.
In the available modeling tools such as Matlab/Simulink® [4] (here called Simulink) and Dymola® [2] it is currently not possible to change the equation system during runtime. Researches on the topic of variable-structure models have been done [3, 5, 7] which resulted in e.g. the Modelica-based tool Mosilab [6] and the experimental language SOL [9]. All approaches present means to create variable-structure models but the model needs to be created in the specific tool or language. In contrast, our approach supports the standard tools Simulink and Dymola.
The general approach of starting, stopping and initializing different modes is equivalent for variable-behavior and variable-detail models. Therefore, Section 2 and Section 3 use the pendulum variable-behavior model to introduce two different approaches to model variable-structure models. While the first introduced way is feasible for small models only, the second way scales with large models. The mode switches in the second approach are realized though a programmed script that initializes and starts the modes. The scripting method is then used to create a variable-detail model of a diesel combustion engine. In the presented example, the simulation time of the variable-detail model was reduced about 50% without significant losses in accuracy compared to a one level of detail model. Based on the findings of the experiments, Section 4 gives an outlook on requirements for modeling variable-structure models. Finally Section 5 presents the conclusions of the paper.

2 Variable-structure models in Simulink and Dymola without scripting

To illustrate how a variable-structure model can be modeled in Simulink and Dymola the pendulum example from the introduction is used. The model has two modes:

- Mode 1: pendulum
- Mode 2: free-falling mass

The simulation starts in the pendulum mode and goes over to the free-falling mass mode when the centrifugal force becomes less than zero. As soon as the mass reaches the rope length again the mode switches back to the pendulum mode. The movement of the pendulum with the two mode switches is illustrated in **Figure 3**. In the following sections, this example is implemented in Simulink and Dymola without using any other means than the ones the modeling tool offers.

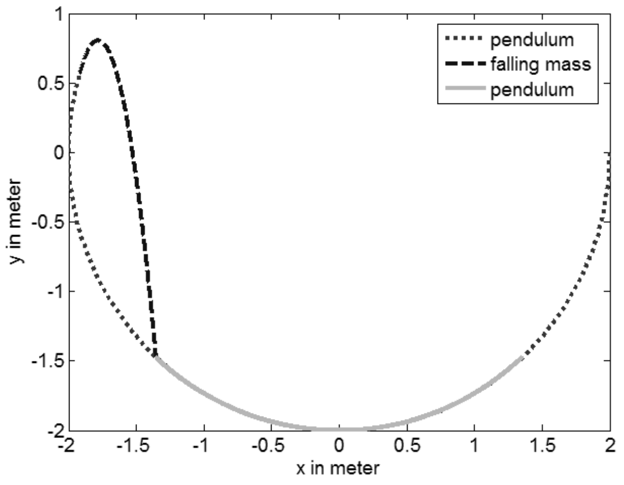

Figure 3: Pendulum simulation results

2.1 Simulink

To implement the pendulum in Simulink enabled subsystems for each mode were used. These modes have to run exclusively and the end values of the old mode need to be the initial values for the new mode. This results in a complicate switching procedure presented in **Figure 4**. Depending on the mode the model is in, different switching conditions have to be checked. This is done through the switch block on the right hand side (mode test). Furthermore, the pendulum starts once with start values given at the beginning of the simulation and once with the end values of the free-falling mass (switch occurred).

This implementation is not recommended when more than two modes are required for the needed switching conditions would become more complicated and the model might not be understandable anymore.

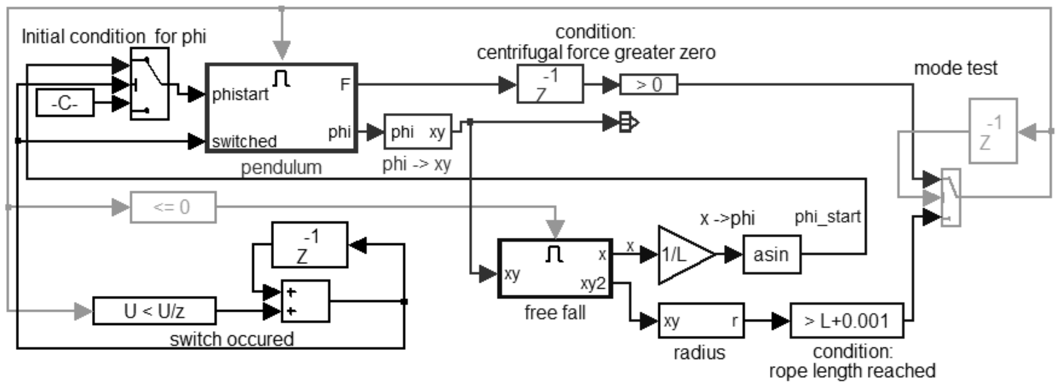

Figure 4: Pendulum in Simulink

2.2 Dymola

In Dymola it is not possible to run the modes exclusively as in Simulink. In Dymola both modes need to run simultaneously and the coordinates for the pendulum have to be chosen according to the mode the model should be in. Therefore, unnecessary calculations are done and more simulation time is required. The shortened code of the Modelica model is presented in **Listings 1**.

```
algorithm
    when (F<=0 and fall <=0) then             // ball equations
     // initialize ball ...                       ...
    end when;                                     r^2 =bx^2+by^2;

    when (r>L and fall>=1) then               // choosing the right coordinates
     // initialize pendulum ...                  if (fall>=1) then
    end when;                                      x = bx;
                                                   y = by;
   equation                                      else
    // pendulum equations ...                      x = px;
       px = sin(phi)*L;                            y = py;
       py = -cos(phi)*L;                         end if;
```

Listings 1: Shortened Modelica-script of a variable-structure pendulum

2.3 Rating of the approaches

The two approaches to model variable-structure systems are in our opinion not recommendable when modeling complex systems. In both tools it is more a workaround than a feasible method. In Simulink the switching procedure gets more complicated for more complex systems and will not be manageable for large systems. The Dymola model has even more deficiencies, because the simulation time will rise with an increasing number of modes due to the simultaneous calculation of the modes. A better approach to model variable-structure systems in Simulink and Dymola is presented in the next section.

3 Variable-structure models with scripting

The switching from one mode to another is done with a script which initializes and starts the new mode. For this approach each mode of the model needs to be implemented as separate model: in Simulink as separate model files (mdl-file) and in Dymola as separate classes. In each model, which represents a mode, a switching condition is required which stops the simulation of the mode. The script then takes control and starts the new mode with initial values calculated from the old mode. **Figure 5** illustrates the course of events when switching back and forth between two modes independent of the modeling tool.

To introduce the scripting approach the pendulum example from Section 2 is used. Three different scripts are presented and discussed:

- Matlab-script with a Simulink model
- MOS-script with a Dymola model
- Matlab-script with a Dymola model

Afterwards, the same three scripts are used to implement a variable-detail combustion engine and the pros and cons regarding the simulation times are discussed.

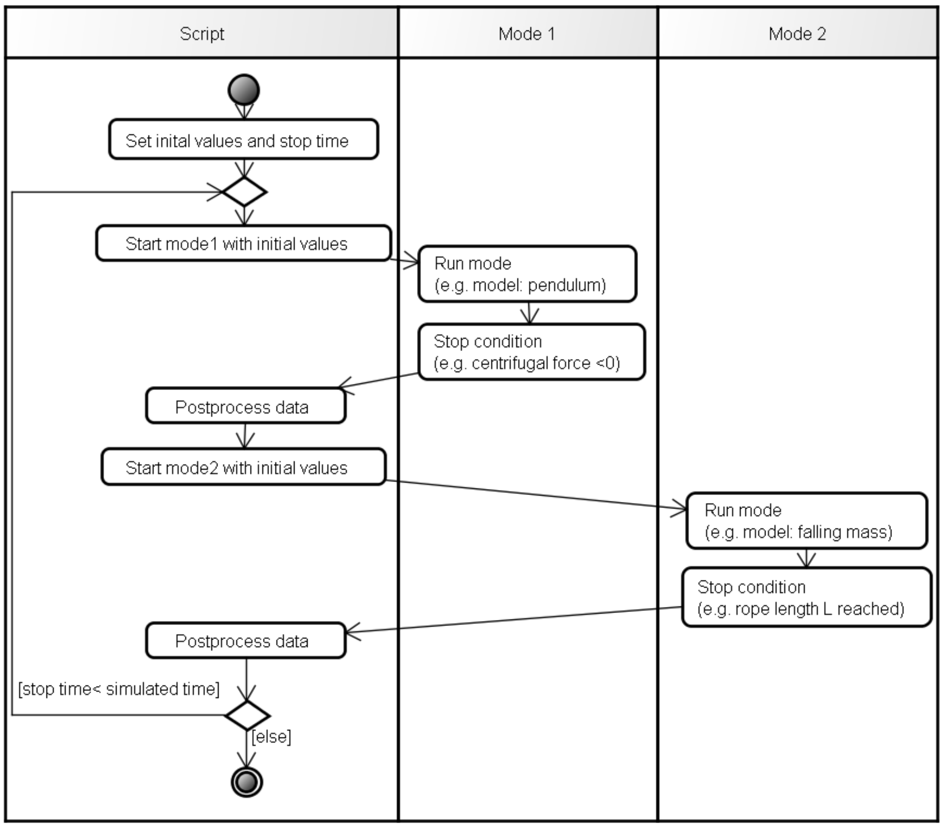
Script
Mode 1
Mode 2
Set inital values and stop time
Start mode1 with initial values
Run mode (e.g. model: pendulum)
Stop condition (e.g. centrifugal force <0)
Postprocess data
Start mode2 with initial values
Run mode (e.g. model: falling mass)
Stop condition (e.g. rope length L reached)
Postprocess data
[stop time< simulated time]
[else]

Figure 5: Course of events for a structural change with scripting

3.1 Pendulum example

To model the pendulum example in Simulink with the scripting approach, the enabled subsystems of the Simulink model from Section 2 are saved in two different modeling files. In Dymola a ball and a pendulum model is implemented in one package but as separate classes.

Simulink with Matlab-script

The Matlab-script for the pendulum derived from **Figure 5** is shown in **Listings 1**. The scripting approach provides the means to run the modes exclusively and the switching condition can be stored directly in the mode. Therefore, the script is rather simple because it only sets the initial values and starts the new mode.

```
sim('pendel.mdl');
  xy = [pendel(end,1);pendel(end,2), pendel(end,3);pendel(end,4)]  % initial value
sim('ball.mdl');

  start_phi = asin(1/L*ballXY(end,1)); % initial value
sim('pendel.mdl');
```

Listings 1: Matlab-script for the mode switch of the pendulum

Dymola with MOS-script

For the Dymola model a MOS-script is implemented which performs the necessary switches (see **Listings 2)**.

```
st =5;
simulateExtendedModel("script.pendulum",stopTime=st,
    initialNames={"phi"}, initialValues={2},finalNames = {"x"});
  x=readTrajectory(fileName,{"x","y", "dx","dy","L"},n);

simulateExtendedModel("script.ball(L=2)",stopTime=5,
    initialNames={"x","y","vx","vy"}, initialValues=x[1:4,n], finalNames = {"x"});
  phi=readTrajectory(fileName1,{"phi"},n);

simulateExtendedModel("script.pendulum",stopTime=st,
    initialNames={"phi"}, initialValues={phi},finalNames = {"x"});
```

Listing 2: Script for the pendulum model

A drawback of using MOS-scripts is that Dymola can only deal with one compiled model at a time. Therefore, each time a switch occurs the model for the next mode needs to be compiled. This means, the pendulum mode of the example needs to be compiled twice. But it should only be necessary to compile each mode once because the equation system of a mode does not change. A model which was already compiled should be reused instead of recompiled. An approach where only necessary compilations are done is presented in the next section.

Dymola with Matlab-script

To overcome the drawback of MOS-scripts that a recompilation is needed for each mode switch we make use of the fact that each time a model is compiled an executable file ‘dymosim.exe’ is created. Matlab provides the functionality to create this executable through Dymola and start it with defined initial values and parameters.

A Java, Pascal or batch-script could be used instead of Matlab but the needed functionality to compile and run a model would have to be implemented first. All modes of the variable-detail model can now be compiled once at the beginning of the script. Afterwards, the script only runs the executable files and sets initial and parameter values (see **Figure 5**).

3.2 Variable-detail engine model

To illustrate the advantages of variable-detail models a diesel combustion engine is modeled. The engine has a manifold which is connected to the environment through a throttle. At the beginning of the simulation, the environment has a different pressure than the manifold. The pressure in the manifold changes as long as there is a pressure difference between the environment and the manifold. When the difference is less than 0.1bar the manifold and throttle can be replaced by an environment with a constant pressure. The switch back to the first mode takes place when the difference between the environment pressure and the initial pressure is greater than a defined threshold. A schematic view of this model is presented in **Figure 6**.

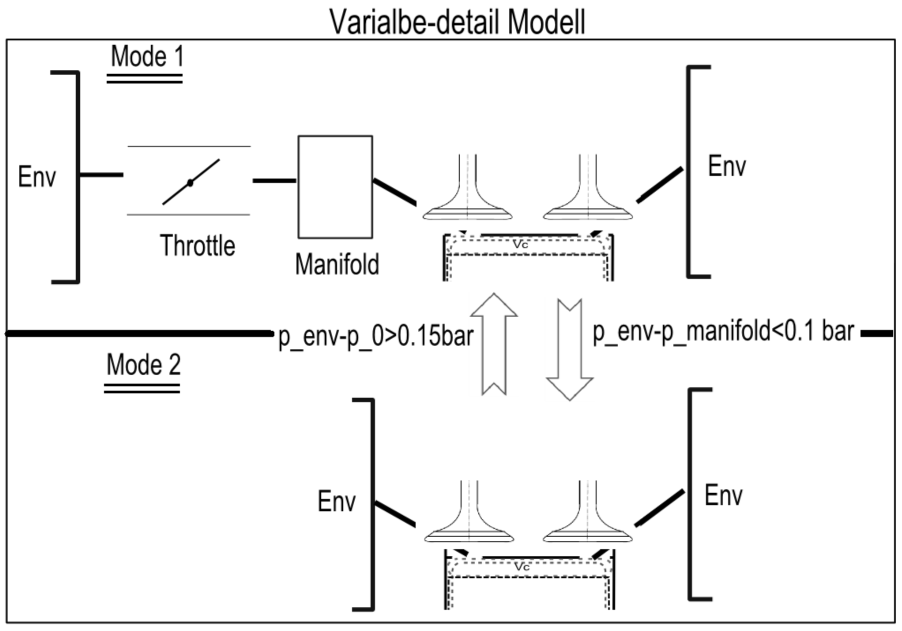

Figure 6: Icon view of the two levels of detail in Dymola

For testing more than one mode switch the environment pressure changes every 5 seconds between 1 and 2 bar and the simulation runs for 20 seconds.

Each time the environment pressure changes the detailed model is used to model the dynamic behavior in the manifold. When a steady-state is reached again the less detailed model is used. This results in seven mode switches.
The model was implemented in Simulink and Dymola and three scripts were written: one in Matlab for the Simulink model, one MOS-File and one Matlab-script for the Dymola model. All scripts work as was presented in **Figure 5** only the used syntax is different dependent on the scripting language and simulation tool. The simulated cylinder temperature from a variable-detail and a one level of detail model is shown in **Figure 7**. The difference between the two models is so marginal that it can be disregarded. When looking at the timestamp of the switch (vertical line) it can be seen that the temperature goes from one mode to the other without visible discontinuities. All performed simulations have equal results and are therefore not presented in separate figures.

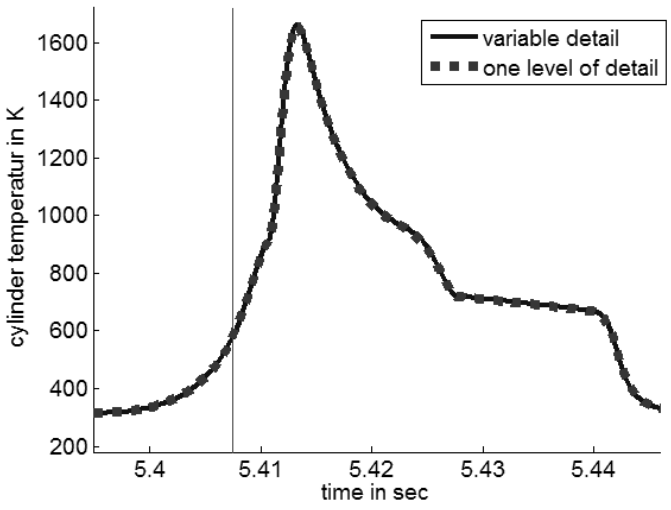

Figure 7: Cylinder temperature of a variable-detail model and a one level of detail model

In **Table 1** measured simulation times are presented for different simulations. The total time is the sum of the simulation time, of the compile time and of the residual time. The compile time is the result of the multiplication of the number of needed compilations and the time needed for each of these compilations. The residual time is the required time for the pre- and postprocessing of all mode switches.
The Dymola variable-detail model needs only one third of the simulation time compared to the one level of detail model. The Dymola model with the Matlab-script is also faster regarding the overall time but the advantage is lessened due to the two compilations. The residual time results from calling the executable and reading the simulation data after the simulation. For both simulations the residual time is equivalent and thus the calculation of the new initial values is so fast that it does not have an effect. The simulation times of the Dymola model with the MOS-script and Matlab-script is equivalent.

But due to the unnecessary compilations for each mode switch and a slower postprocessing, the total time of the MOS-script model is longer compared to the total time of the one level of detail model.

In Simulink the advantages of the variable-detail model are almost non-existing. This is the case because the less detailed mode needs almost the same simulation time as the more detailed mode. The switching of modes works in Simulink but to gain time advantages modes with larger time differences are required. Although, the equations of the Simulink model and the Dymola model are the same, the Simulink model needs a lot more simulation time. Regarding the simulation speed, the Dymola variable-detail model with the Matlab-script yields the best results.

Stop time 20sec/ 7 Switches	**One detail Dymola**	**Variable-detail Dymola/Matlab**	**Variable-detail Dymola/MOS**	**One detail Simulink**	**Variable-detail Simulink**
Simulation time	45	15	16	180	174
Compilation time	1* 5sec	2*5sec= 10sec	7*5sec = 35sec	-	-
Residual time	7	7	19	0	2
Total time	**57**	**32**	**70**	**180**	**176**

Table 1: Table with simulation times through scripting

4 Drawbacks and future work

In this section an overview of the drawbacks of the scripting approach and ideas to overcome these is given. The scripting approach will become rather complicated when many modes are necessary. Therefore, a more convenient approach would be to describe the switching in the modeling language directly and let the compiler create the needed script. Furthermore, a method is needed where variable-structure models can be created independent of a specific tool. A possible syntax for such an approach might be SysML [8]. In such methods additional information should be added to the models, such as for which experiments the model is valid [1]. With the knowledge whether a model is valid for a certain experiment, necessary mode switches can be detected and the compatibility of subsystems can be proven. We are currently working on these ideas.

Some rules concerning variable-detailed models can be learned from the simulation results:

- Switches should not occur too often, because they might consume all performance gains. Analyses about how switching conditions should be laid out are necessary.
- The switching time needs to be compensated by the speed advantages of the less detailed model – when the goal is saving simulation time.
- Using Dymola with MOS-scripts is not feasible as long as models need to be compiled for each mode switch.

5 Conclusion

In this paper an approach to simulate variable-structure models with scripts was presented. With this approach the model can be implemented in common modeling tools such as Simulink or Dymola.

We think that the scripting approach is an easy way to handle variable-structure models without having to work with complicated language constructs and dynamic creation/destructions of objects. By creating one model for each mode in Simulink or Dymola the mode switching can be done with a script. With a Dymola variable-detail model of a diesel combustion engine the simulation time was reduced significantly compared to a one level of detail model.

Simulink models have some drawbacks when creating variable-detail models. Firstly maintaining models can become quite difficult if a submodel is contained in more than one model. Secondly, the simulation speed of the diesel combustion engine model in Simulink compared to the equivalent Dymola model is much slower. We think that Dymola models are in advantage regarding variable-structure models because the object oriented approach makes it quite easy to create different modes and maintain them. The scripting approach with Dymola models enables the modeler to create variable-detail models from existing models and thus benefit from the speed and accuracy advantages.

6 References

[1] *Cellier, F.*: Continuous System Modeling. Springer-Verlag New York, 1991.

[2] *Dynasim.* Dymola version 7.4. http://www.dymola.com [Last access: July 22, 2011]

[3] *Friesen, V.*: Objektorientierte Spezifikation hybrider Systeme, PhD Thesis TU Berlin, 1997

[4] *Mathworks*: Matlab/Simulink version 2011a. http://www.mathworks.de [Last access: July 22, 1011]

[5] *Nordwig, A.* Integration von Sichten für die objektorientierte Modellierung hybrider Systestem, PhD Thesis, TU Berlin, 2003

[6] *Nytsch-Geusen, C.*: Advanced modeling and simulation techniques in MOSILAB. A system development case study, In proceedings, 5th International Modelica Conference TU Hamburg-Harburg, 2006.

[7] Pepper,P., Mehlhase, A. Scholz, L., Höger, C.: Modelica-style variable-structure modeling., In Equation-Based Object-Oriented Modeling Languages and Tools, Zürich, Switzerland (to appear), 2011

[8] *Weilkiens, T.*: System Engineering with SysML. Morgan Kaufmann, 2008.

[9] *Zimmer, D,* Equation-Based Modeling of Variable-Structure Systems. PhD Thesis, Swiss Federal Institute of Technology, 2010.