

Hand-in for the Module “Modelling” in the course TME102 Vehicle Dynamics Advanced

0 Preparations

0.1 Install Modelica tool

See https://chalmers.instructure.com/courses/23432/pages/how-to-access-tools?module_item_id=350933

Install one of the Modelica tools:

- Dymola
- OpenModelica, see <https://openmodelica.org/>

Dymola is recommended for the course and Chalmers has license for education and research. OpenModelica is an open and free tool, so you can use it also after you leave Chalmers.

0.2 Arranging folder, files, package, models, etc

- Create a working directory on your computer, e.g., `C:\...\Module_Modelling\HandIn`
- Download the Modelica file `package_HandOut.mo` from the course web. Save the file as `C:\...\Module_Modelling\Hand-in\package_YourName.mo`
- Open by double-clicking the file `package_YourName.mo`
- Rename the package to same name as your file, `package_YourName`
- `Ctrl+S` to save
- Find the model `HelloWorld`.
 - Check that you can see the model code in the “Text tab”.
 - Click “Check”. It should lead to:
 - The model has the same number of unknowns and equations: 1
 - Check ... successful
 - Move to the “Simulation tab”
 - Click “Simulate”, it should lead to:
 - ... Integration terminated successfully at $T = 1$...
 - Open a plot window. Select some variable to plot, for instance `x`. It should show `x` dropping exponential with time, starting from `x=1`.
- Exit the Modelica tool. You have now prepared well for the Tasks in this Hand-in.

Modelica and FMI tools typically saves their generated files (C files, result files, command scripts `*.mos`, FMU files `*.fmu`, ...) in the Working Directory. If you open a package by double-clicking, you will directly get `C:\...\Module_Modelling\HandIn` as Working Directory, which is the recommended way of working for this hand-in.

The Modelica book: <https://mbe.modelica.university/> can be useful while working with the hand-in. Find it and book-mark it. Browse https://mbe.modelica.university/behavior/equations/first_order/ and <https://mbe.modelica.university/behavior/equations/physical/>.

0.3 Tasks overview

The tasks are presented below, in an order which is suitable to carry them out. So, if you only aim at pass, you are recommended to do only Task 1. Task 1 gives up to 3 points, so you have a reasonable margin to reach the 2.4 points required for pass. See also course memorandum.

The hand-in is defined as a “Canvas Assignment”, so you should submit via Canvas. There is a due date in Canvas. You do not need to do a proper report, but some kind of **document** (*.pdf, *.word and, *.ppt) and at least **one Modelica file** (*.mo).

There are sections called “What to hand-in” in the end of each task or subtask.

1 Task 1 (3p)

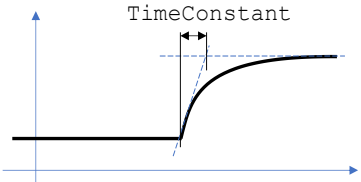

There are 3 subtasks: 1.1, 1.2, 1.3, and 1.4.

The main work and time should be on 1.1. However, 1.2, 1.3, and 1.4 are there to encourage/force you to see 1.1 in an “vehicle engineering context”.

1.1 Add physical phenomena to the model

You shall add the 3 physical phenomena in the Table 1-1.

Table 1-1: Three physical phenomena to be added

Phenomenon	Pictogram
<p>Add a (continuous) 1st order time delay of the torque generation of the Prime Mover.</p> <p>Use parameter $TimeConstant = 0.05$ [s]</p>	 $\text{der}(T) = (T_PmReq - T) / \text{TimeConstant};$
<p>Add Drive Shafts stiffness (change it from rigid to compliant).</p> <p>This addition introduces the design parameter $c_{DriveShafts}$.</p> <p>Use parameter $c_{DriveShafts} = 23$ [kNm/rad] (This is the reference design parameter value.)</p>	 $\text{der}(T) = \dots;$
<p>Add (linear and unsaturated) Tyre Slip.</p> <p>Note: Model is prepared, so you only need to change from</p> <pre>parameter Boolean ModelTyreSlip = false; // =true; //</pre> <p>to</p> <pre>parameter Boolean ModelTyreSlip = true; // =false; //</pre> <p>(So, the model is prepared but you are recommended to reflect over that this really shows that Modelica is not a <u>programme</u> language but a <u>modelling</u> language. The different branches in the if statement are not executed as in a traditional programming language. Instead, the declare different equations, which makes the manipulation to <i>Explicit form</i>, of the whole model, very different.)</p>	

1.1.1 Recommended work steps

- In the handed out Modelica file there is a package “package_HandOuts”. It includes the model “_1D”, which is the same model as presented at the lecture. Test simulate it and plot some results. (You can plot with selecting variables in the variable list or with the handed-out plot-script “plot_1D_HandOut.mos”. (The script language is not standardised, so the script file only works in the tool Dymola.) Such a test simulation should result in plots approximately as Figure 1-2.
- Do a **Physical model** (drawing) first and use it for doing the Mathematical model. Good modelling starts from understanding, which is best reached by drawing. For instance, the drawing will be the place where you define your new variables.

- Now it is time to make the **Mathematical model**, i.e., to change/add text/code to represent the phenomena.
 - Copy the handed-out model and rename it to, e.g., “_1D_Task1”.
 - Now change/add the text/code. As you see in Table 1-1, the modelling declarations and equations are not exactly stipulated, so you will need to invent some own variable and parameter names.
 - A recommendation is to check frequently, both by “check button” and “simulate button”. Then it is easier to find and correct errors.

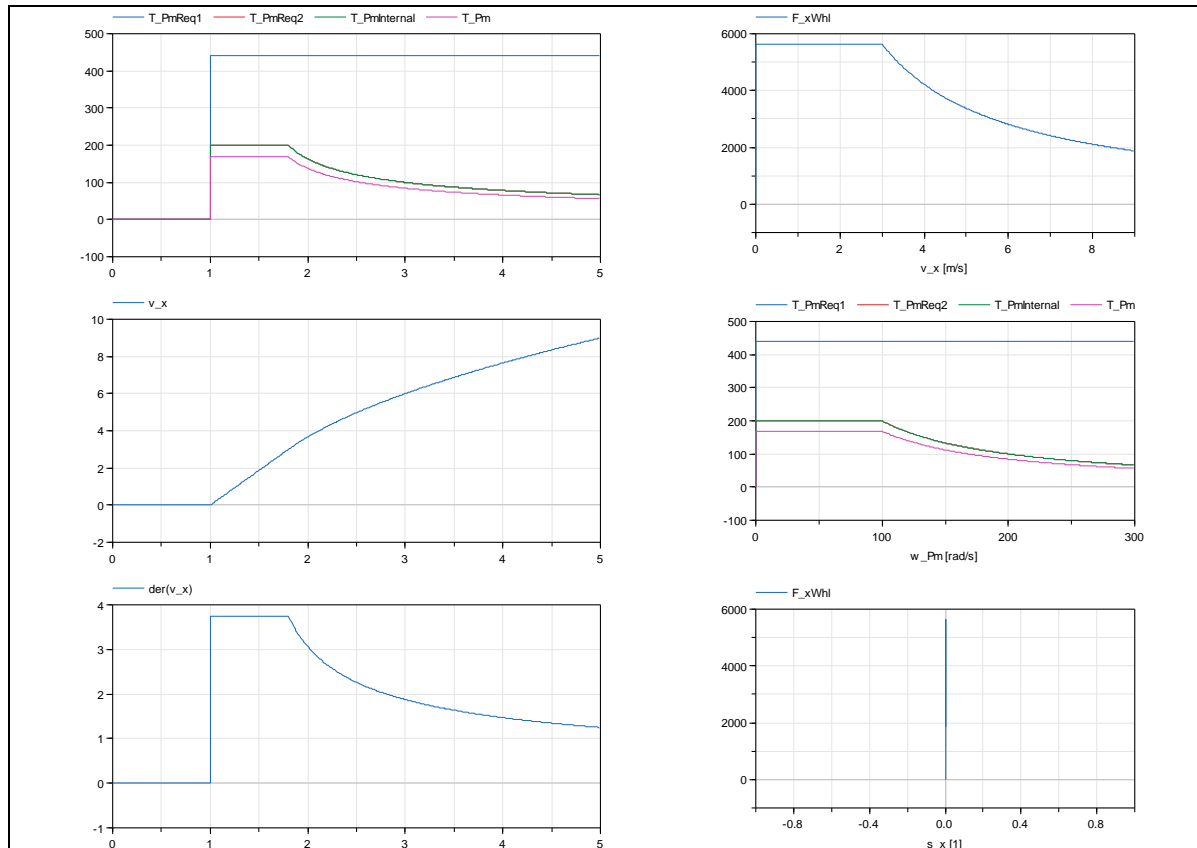


Figure 1-2: Result of the handed-out plot script

1.1.2 What to hand-in

- A **Physical model**, in the form of a drawing, FBD, etc e.g., in manual drawing, power point or similar). If you want to do it in power point, there is some graphical material as starting point in slide 2 in the power point file in the “tools folder”.
- A **Mathematical model**, in the form of Modelica code, suitably as a model in the package in the file named “package_YourName.mo”.

1.2 Judge model validity

Before using a model to engineering/design, one must judge the **model validity**.

1.2.1 Recommended work steps

- Since you do not have any measurement data, your judgement must be based on your intuition/experience. You can at least compare it with the handed-out model, without the added phenomena. For this it is useful to plot one (or some) variables from the model without and with added phenomena, see Figure 1-3. Interesting variables to plot could be vehicle longitudinal velocity and acceleration. Maybe also tyre longitudinal and vertical force.
- Reflect on:

- Is the difference expected and explainable?
- Which are the risks to use the model outside its validity?
- Are there any *other* physical phenomena that you think would be valuable to add? Or is there any which can be *removed* to simplify model?

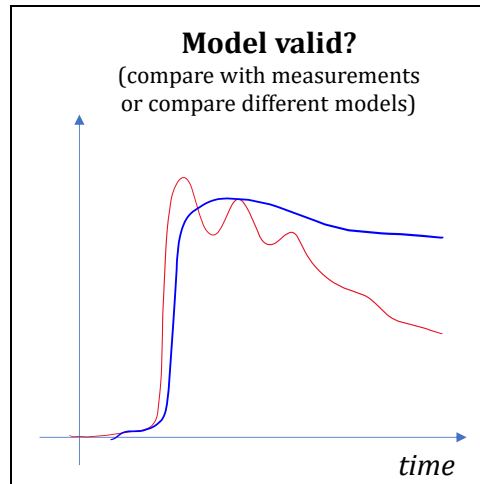


Figure 1-3: A typical diagram to judge the model validity.

1.2.2 What to hand-in

- A **plot with two curves** of a variable as function of time, one curve with the original model and one with the model with added phenomena, something like Figure 1-3.
- Some sentences, maybe 2-4, from your **reflections**.

1.3 Preparations for Virtual Verification/Development

1.3.1 Recommended work steps

Read the following. If you agree, just take it as it is. You can also change/add design parameter, attribute, manoeuvre, function, and requirements.

Take $c_{DriveShafts}$ as the *design parameter*. With the *attribute* "User Experience" in mind, one can create *manoeuvre* with *functions* and *requirements* as follows:

- The **manoeuvre** is a maximum acceleration on flat ground with friction peak of 0.6.
- Here are relevant **function measures**, connected to the manoeuvre. See also Figure 1-4:
 - Time $t_{0..30}$ in [s] for acceleration 0-30 km/h
 - Overshoot in acceleration as fraction of mean acceleration $\Delta a_x/a_{x0}$ in $[(m/s^2)/(m/s^2)]$
- Here are relevant (quantified complete vehicle) **requirements**, connected to these measures are:
 - The vehicle shall do the acceleration in $t_{0..30} < 4$ [s]
 - The vehicle acceleration fraction shall overshoot with $\Delta a_x/a_{x0} < 0.25 [(m/s^2)/(m/s^2)]$

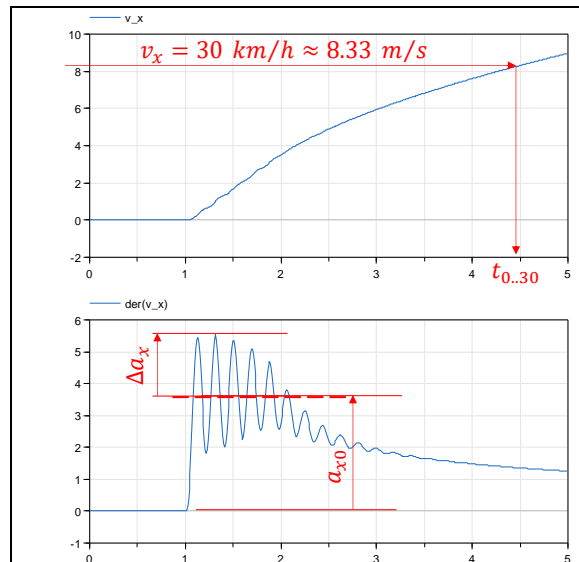


Figure 1-4: Definition of function measures

1.3.2 What to hand-in

You do not have to hand-in anything from 1.3. (Unless you made changes/additions in 1.3.1. Then define those changes/additions.)

1.4 Virtual Verification/Development

So far, you have only compared different models. You have not been asked to compare the different designs. Such comparison should now be made. It will be enough to compare two values of the design parameter $c_{DriveShafts}$.

1.4.1 Recommended work steps

One of the tests can be without any changes in parameters, i.e., you can use the simulation from 1.2.

In the other test, you change to another value of $c_{DriveShafts}$. A proposal is to change it 50% up or down, depending on your curiosity.

To compare different values on the design parameters, one need to compute the achieved **function measures** for each value of the design parameters. These can be put in a table as Figure 1-5 them in a table for comparison. In the same context, one can add the requirement. Such a table is typically what you need to **compare** the design parameters and propose a value on it.

Best design? (compare different design parameter values)			
	Design parameter		Require- ment
	7	8	
Function measure	30	35	>32

Figure 1-5: A way to diagram to propose a design parameter.

If we need to re-design, we will also need to understand how the design parameters influence the physics. For this purpose, it is typically good to plot some **time trajectories** for different values of the design parameters, such as Figure 1-6.

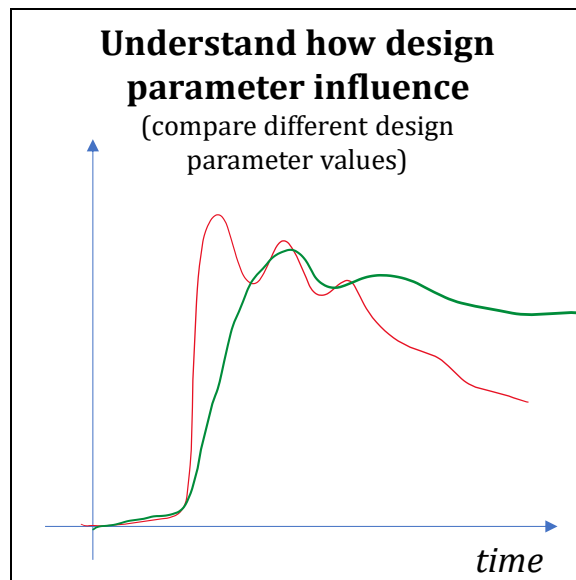


Figure 1-6: A typical diagram to understand the influence of a design parameter.

Write 3-5 sentences about a possible next design loop. E.g., discuss:

- Are there *other values* on the design parameter $c_{DriveShafts}$ which you would propose to try? Why? Which?
- Would you recommend varying *other design parameters*? Why? Which?
- Which *operational parameters* (defining the manoeuvre) could be interesting to vary?
- Are the *function measures* suitable?
- Are the required *values* of the requirements suitable?

1.4.2 What to hand-in

- A **table** with $2 \times 2 = 4$ values: Your 2 function measures ($t_{0..30}$ and $\Delta a_x/a_{x0}$) for your 2 tested value on $c_{DriveShafts}$. Something like Figure 1-5.
- Propose a value on $c_{DriveShafts}$. Or, conclude that none of the tested values fulfil the requirements.
- A **plot** with 2 curves of a variable as function of time: for your 2 tested value on $c_{DriveShafts}$. Something like Figure 1-6.
- A proposal of which 3rd value on $c_{DriveShafts}$ to test if one should continue the design iteration.

2 Task 2 (3p)

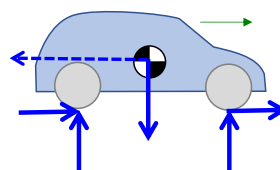
This task contains subtasks where more physical phenomena are added to the model.

2.1 Load Transfer

Add Longitudinal Load Transfer (longitudinal transfer of vertical force between the axles) using a rigid suspension model. You can assume a distribution of propulsion; only on one of the axles or distributed in some other way.

Use:

- $l_f = 1.25 [m]$
- $l_r = 1.5 [m]$
- $h = 0.4 [m]$

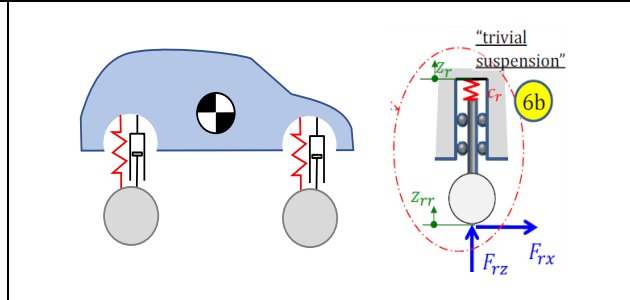


2.2 Suspension

Add ("Trivial") Suspension compliance (=1/stiffness) and damping.

Use:

- $c_{fz} = c_{rz} = 30 \text{ [kN/m]}$
- $d_{fz} = d_{rz} = 2.5 \text{ [kN/(m/s)]}$
- $J_y = 3000 \text{ [kg m}^2\text{]}$



2.3 Recommended work steps

For each of the subtasks in 2.1 and 2.2 you can probably work approximately as in 1.1 and 1.2.

2.4 What to hand-in

For each of the subtasks in 2.1 and 2.2 you should hand in:

- Physical model
- Mathematical model (in Modelica format, e.g., one model in file `package_YourName.mo`)
- Plots of some time trajectories from a simulation. It is suitable to use the reference value of the design parameter $c_{DriveShafts}$ also used in 1.2.
- (To keep the workload of this hand-in limited, you do not need to write any analysis of the simulation or plot. But feel free to ask questions and Bengt will try to answer them while correcting the hand-in.)

3 Voluntary Task 3

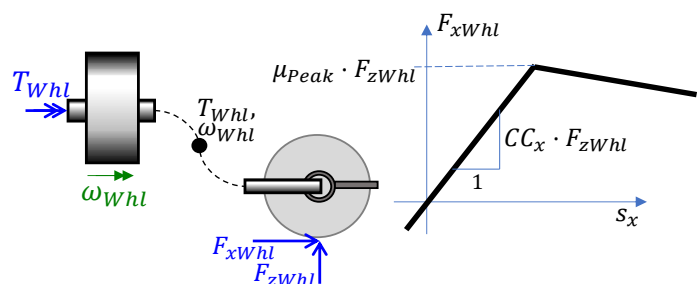
The two subtasks 3.1 and 3.2 was, during course development, included as two additional subtasks in task 2. But then moved to this separate voluntary Task 3, to reduce risk of a too demanding Hand-ins. They were not deleted from this document, since they are relevant and should at least be made available for the course participants.

You can:

- Simply disregard the whole "3 Voluntary Task 3",
- Read the instructions for your own curiosity, or,
- Do one or two of subtasks 3.1 and 3.2. However, they will not give you any points, unless you have less than maximum 3+3=6 points from Task 1 and 2: $Points_{HandIn} = \max(Points_{Task1} + Points_{Task2} + Points_{Task3}, 6)$

3.1 Wheel Inertia and Tyre Friction Saturation

Add wheel rotational inertia of the wheel and saturation of the tyre friction force.



Note: Model is prepared, so you only need to change from

```
parameter Boolean ModelWheelInertia = false; // =true; //
parameter Boolean ModelTyreFrictionSaturation = false; // =true; //
```

to

```
parameter Boolean ModelWheelInertia =true; // =false; //
parameter Boolean ModelTyreFrictionSaturation =true; // =false; //
```

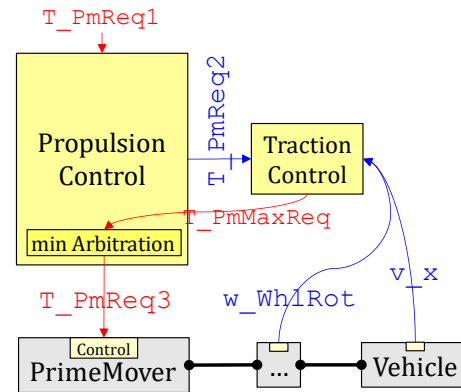
3.2 Mechatronics

Add **Traction Control** (or Wheel Slip Control or Wheel Slip Max Limitation) that limits the slip by reducing the Prime mover torque.

Model it with time sampling, e.g. *SampleTime* = 0.02 [s].

The performance of the controller is not important in this hand-in as long as it works conceptually correct. One way can be to simply cut the Prime Mover torque generation to zero if slip is above a certain limit.

Hints of how you can draw and think is given in figure and on next table row. It proposes to have two sampled algorithms, *PropCtrl* and *TractionControl*.

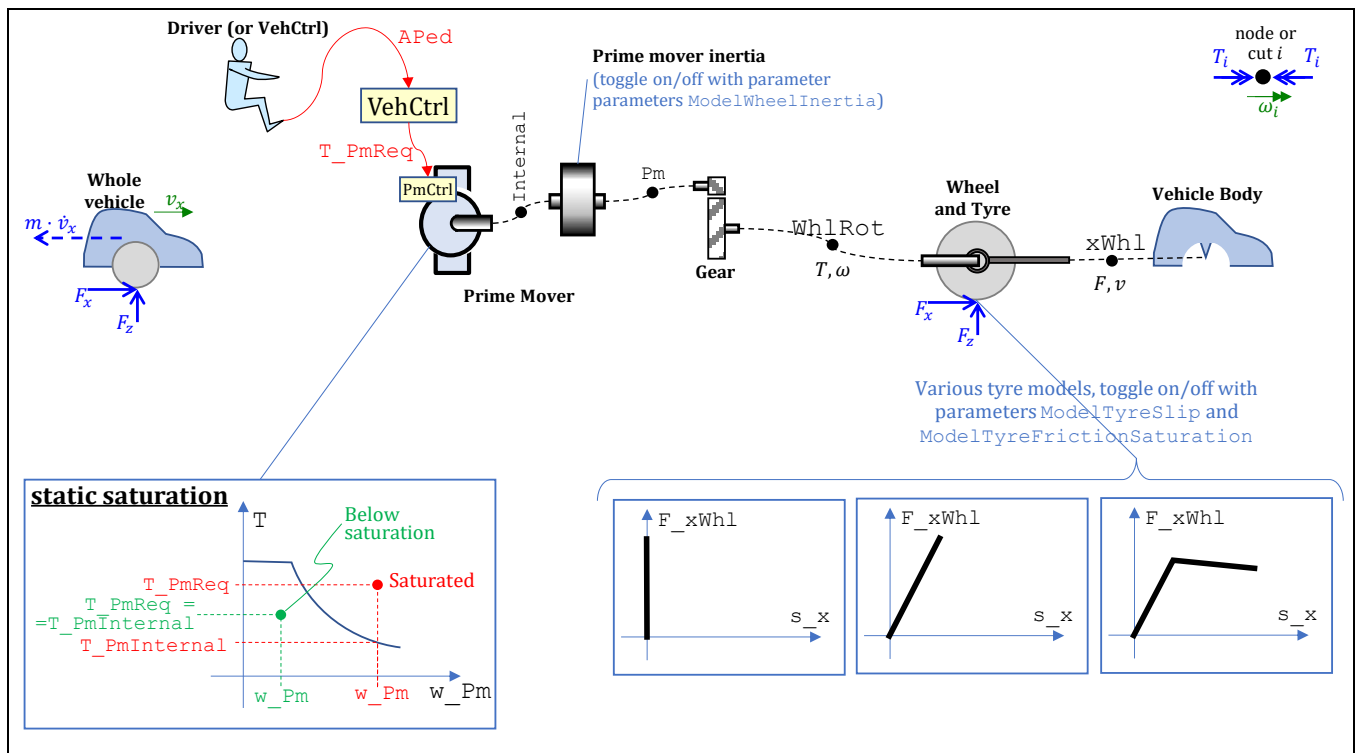


```
//PropCtrl (part of VehCtrl):
when sample(0, SampleTime_Prop) then
  T_PmReq2=...T_PmReq1...T_PmMax...P_PmMax...;
  T_PmReq3=min(T_PmReq2, pre(T_PmMaxReq));
end when;
```

```
//TractionControl:
when sample(0, SampleTime_TC) then
  s_x=...;
  if s_x<s_xMaxReq then
    T_PmMaxReq= pre(T_PmReq2);
  else
    T_PmMaxReq=0;
  end if;
end when;
```

4 Appendix: Model description

Beside the handed out Modelica code, the below Physical model (drawing) describes the handed out model.



5 Appendix: FAQ

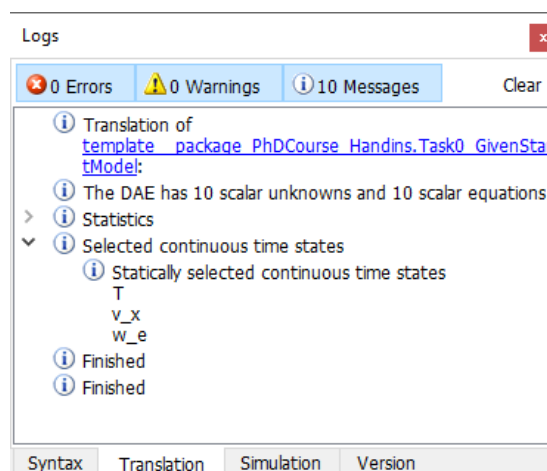
This section includes questions and answers which have occurred for course participants. **If new issues appear during the course, this section will be updated.**

5.1.1 Problem with C-compiler used by Dymola.

One possible C-compiler is recommended at <https://www.3ds.com/products-services/catia/products/dymola/c-compiler/>, look under “GCC compilers”. Download and install C-compiler “64-bit MinGW”.

5.1.2 How to see which state variables that Dymola selects?

In the “Logs” window, under “Translation”, under “Selected ...”:



5.1.3 Documentation of Modelica commands

A kind of “help function” is to do search as follows:

The screenshot shows the Modelica documentation interface. The 'Package Browser' on the left lists various operators, with 'sample()' selected and highlighted. The main panel displays the documentation for 'sample()', including its information, syntax, description, and examples. A graph at the bottom right illustrates the step-like output of the sample operator over time.

sample()

Information
Trigger time events

Syntax
`sample(start, interval)`

Description
Returns true and triggers time events at time instants "start + i*interval" (1, ...). During continuous integration the operator returns always false. The start time "start" and the sample interval "interval" need to be parameter expressions and need to be a subtype of Real or Integer.

Examples

```
model Sampling
  Integer i;
equation
  when sample(1, 0.1) then
    i = pre(i) + 1;
  end when;
```

The graph shows a step function. The x-axis represents time, and the y-axis represents the value of the sample operator. The function is 0 for most of the time, but it jumps to 1 at regular intervals (0.1 units), creating a staircase pattern. The y-axis ranges from 4 to 12, and the x-axis ranges from 0 to 10.