CHALMERS
UNIVERSITY OF TECHNOLOGY

Towards a simulation environment for Operating Cycle Analysis of Road Transports

Master’s Thesis in in the Master´s programme Automotive engineering

STEFAN VENBRANT

Department of Applied Mechanics
Division of Vehicle Engineering and Autonomous Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2015

MASTER’S THESIS IN AUTOMOTIVE ENGINEERING

Towards a simulation environment for Operating Cycle Analysis of Road Transports

STEFAN VENBRANT

Department of Applied Mechanics
Division of Vehicle Engineering and Autonomous Systems
Vehicle Dynamics group
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2015

Towards a simulation environment for Operating Cycle Analysis of Road Transports

STEFAN VENBRANT

Master's Thesis 2015:70
ISSN 1652-8557
Department of Applied Mechanics
Division of Vehicle Engineering and Autonomous Systems
Vehicle Dynamics group
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone: + 46 (0)31-772 1000

Cover:

Vikström, Fredrik Diits. 2012. Nya tester med Volvos fordonståg. *ViBilägare*. January 25. http://www.vibilagare.se/nyheter/nya-tester-med-volvos-fordonstag-36293 (downloaded 2015-08-12)

Chalmers Reproservice/ Department of Applied Mechanics
Göteborg, Sweden 2015

Towards a simulation environment for Operating Cycle Analysis of Road Transports
Master's thesis in Master's programme Automotive Engineering
STEFAN VENBRANT
Department of Applied Mechanics
Division of Vehicle Engineering and Autonomous Systems
Vehicle Dynamics group
Chalmers University of Technology

Abstract

One of the biggest challengers in today's vehicle industry is to reduce fuel consumption, reduce emissions and lower energy consumption. There is several ways that the car manufacturers engage these challenges, like downsizing engines, reducing aerodynamic resistance etc.

Another way is to investigate how the vehicles are being used and configure them based on that. This means that e.g. a truck intended for city driving could be configured for this already when ordered by the logistic company. For this to occur simulations performed in simulation softwares must be improved and operating cycles used in these simulations has to be more accurate.

There are two main aims with this thesis: the first is to investigate and propose two simulation environments that could be used by stockholders interested of performing their own simulations on Volvo vehicles and others. One should be open source software for stockholders who cant afford to buy expensive softwares.

The second aim is to be a start-up phase for a larger project called OCEAN [1].

This is conducted as follows: a number of simulation softwares are investigated and two are chosen, these are Simulink and OpenModelica.

A standard called FMI (Functional Mock-up Interface) is investigated and later used. FMI allows models to be exchanged between different softwares called S-models meaning that parameters can be changed but the equations behind are hidden.

Four simulation combinations of a truck are built in the following combinations and softwares (one model was made in OpenModelica but because of problems in OpenModelica Dymola had to be used instead, see conclusions if more interested):

1. Simulink model built and simulated in Simulink
2. Model modelled in a Modelica tool and simulated in Dymola
3. Simulink model with FMUs generated in a Modelica tool
4. Model modelled in a Modelica tool with FMUs generated in Simulink and simulated in Dymola

Result of actual speed v_is, accelerator and brake pedal position, simulation time and fuel consumption are recorded and compared, this to see if a vehicle built in different softwares but with the same parameters give the same result. Conclusions are made and recommendation on softwares and future work is made.

Key words: Simulation softwares, Operating Cycles, FMI

Contents

Preface

This report is the result of a master thesis work carried out at the Department of Applied Mechanics at Chalmers University of Technology in collaboration with the department System Engineering and Integration at Volvo Group Trucks Technology.

This thesis is a start-up for the larger project OCEAN where Pär Petterson is the Ph.D student.
I would like to thank my supervior Sixten Berglund at Volvo GTT for the support and advises during this thesis.
I also want to thank Pär Petterson for support and help when I needed.
A big thank you to Bengt Jacobson who was my examinator and also gave me good advises and support during the thesis work.

The help support at Modelon and especially Erik Åberg have been very helpful every time I turned to them with questions about FMI toolbox and how to get it to work. Thank you.

Finally I want to express my gratitude to the support at Maplesim/Maplesoft as well as the support at OpenModelica for answering my questions whenever I had any.

This thesis work has been a great experience and have given me a first insight of how simulation tool works, how important and helpful these are for vehicle design and some of the problems that can occour when using these tool. This will be very helpful for my future professional career.

Göteborg June 2015

STEFAN VENBRANT

1 Introduction

1.1 Background

A big challenge in todays vehicle industry is the reduction of fuel consumption. The fuel price continues to increase and is expected to keep doing so in the future. Trucks represent a big part of the consumption because of their use in transportation of cargo. Several ways of reducing fuel consumption is being investigated, e.g. downsizing of engines, aerodynamic performance etc. Another way is to investigate how the vehicles are being used and configure them based on that.
Through getting more knowledge in this area it would be possible for a logistic company, when ordering a number of trucks, to tell the truck manufacturer how the trucks are intended to be used, eg a truck intended for city driving. The manufacturer could optimize the truck configuration for this and possibly even tune some design parameters for the specific condition.
In order for the manufacturer to know how to tune the vehicles for different driving conditions simulations and tests must be conducted. Tests are done by driving test trucks equipped with sensors and other information gathering equipment in different operating conditions while simulations can be done with computers and simulation softwares.
Simulation softwares are a big help when developing new vehicles. The use of softwares makes the testing phase cheaper and faster. In the simulation softwares operating cycles are used. Different operating cycles are used depending on what situation is of interest, e.g. city cycle for city driving, highway cycle for highway, etc.

There is also a need for a driving cycle that covers how trucks are driven in loading terminals when being loaded/ unloaded. Even if the trucks are turned off most of the time when being in the terminals the cool down period is quite long, and this needs to be simulated in a driving cycle. Other driving cycles that there is a need for is how busses are driven in bus terminals, vehicles in battery charging areas etc.

Today there are a number of simulation tools that can be used to estimate fuel efficiency, energy consumption and other parameters. Volvo Group use Global Simulation Platform (GSP) including Global Simulation Tool (GST) when running simulations to estimate fuel consumption and similar data. In order to do this a number of different submodels (like engines, transmissions etc.) from vehicles (trucks, busses, etc.) but also for boats, generator-sets and construction equipment has been added to GST, together with a couple of different driver behaviors and operating cycles. In the subparts it is easy to change properties like number of gears, chassis type, drag coefficient, etc., so different vehicles can be simulated.

1.2 Problem motivating the project

Because GSP is developed and owned by Volvo Group, other stakeholders (companies, universities, organizations, etc.) do not have permission to use this program. If looking at a larger perspective the same goes for other companies around the world, they also use their own developed softwares that others outside the companies cant access.
In order to be able to solve tomorrow's challenges in this area it is of importance to let other being able to do their own simulations and investigations with Volvo parts/vehicles as well as other companies designs.

For this to happen there is a need for research in other simulation tools that can be used by the stakeholders. Also it should be possible to easily save between GSP and the other simulation tools. This to make it easier to save “washed” parts from Volvos libraries to the new simulation programs and the other way around.

1.3 Aim

This thesis have two main purposes.

- One purpose of the thesis work is to prepare for an open source library from where stakeholders, who want to do their own vehicle simulations, can import the sub parts needed and by using the simulation programs investigated in this thesis obtaining their own reliable data. There will also be driving cycles and driver models in the data base.
- The second main purpose of this thesis work is to be the start-up phase for a larger project called OCEAN [1].

1.4 Research questions

The question that is answered in this thesis work is:

- What simulation formats and simulation tools should be used to build a open modeling and simulation library for a simulation environment for operating cycle analysis of road transports?

1.5 Method

1. Investigate different simulations tools that exist, and choose 2-3 to continue with.
2. A vehicle is chosen to be modelled in the simulation softwares together with a operating cycle and driver. This model is then simulated in the chosen tools.
3. Simulation results are compared and discussed.

1.6 Limitations

- It exist quite a few simulation tools but in this thesis work some that might prove to be a good candidate to use will be discarded, for example if there is no access to free trial version, etc.

- Cost will also be a limitation when investigating simulation tools. It is not possible to buy all simulation tools just to test and see if they fulfill requirements.

- Only traditional driving cycle concept will be used in simulations, e.g., $v_{set}(t)$ (speed over a certain time) or $v_{set}(s)$ (speed over a certain distance).

- In chapter 1.4 both simulation formats and simulation tools are mentioned. However only simulation tools will be investigated. Formats will be chosen depending on what tools that are chosen.

2 Simulation Tools

Today there are a large number of simulation tools being used in a wide number of areas. Because of the time limit of this thesis only a few of these can be chosen for a closer investigation in order to decide which ones to use in this thesis work. Seven promising tools were chosen to be investigated closer. The tools were "weighted" from a number of important parameters, like usability, modeling capability, stability, etc. The Pugh matrix used for this will be shown in chapter 2.6. The tools chosen were: Matlab/Simulink, Dymola, MapleSim, OpenModelica, Scicos, Scilab and Python.
Tools with the possibility to use graphical interface modeling is preferred before a text based modeling software, this mainly because of the complexity of modeling a vehicle; there are a big number of equations that is needed in text based modeling. Python was however included in the initial investigation because of its popularity and low cost and usability when learned.

Also a quick explanation of the difference between simulation tools and formats is needed. Easy explained simulation tools are the software we build and simulate our models in. Simulation format is the language the simulation tool uses. E.g. Simulink is both the tool used to build models and the format but in e.g. OpenModelica the simulation tool is OM but the format/language is Modelica.

2.1 Early overall decision about 2 tools

1. Matlab/Simulink was chosen from on an early stage, this because of a number of reason. It is widely used software known by many companies and universities.
2. At least one of the chosen tools should be open source. This because smaller companies, research facilities or universities might not be able to buy an expensive license for a software to do their own simulations.

A feature that enables exchange between the selected tools was also early decided would be a advantage. This would allow the possibility to exchange models/ sub parts between tools and thus saving the time of modeling from scratch in each simulation tool. This feature would also add two extra modeling & simulation examples in order to evaluate the result.

2.2 Examples

When the simulation softwares were evaluated a model was built in each of the softwares. In the simulation softwares Matlab/Simulink, Dymola, MapleSim and OpenModelica a simple DC Engine was modelled. In Scicos and Scilab a Continous function was used.
In- and out power of a DC engine can simply be described as follow:

In/Out power

$P_{in}= I*V$	where I is input current and V is input voltage
$P_{out}=T*\omega$	where T is torque and ω is angular velocity

With the in and out power the efficiency is calculated:
$\eta = P_{out}/P_{in}$

2.3 GSP as reference software

Global Simulation Platform is a software developed and owned by Volvo Group designed to evaluate longitudinal vehicle dynamics, fuel consumption and other performances of vehicles. It can also be used to develop powertrain componets. It is built on Simulink and includes several libraries with operating cycles, driver behaviors, truck and bus set-ups, etc as well as for boats, generator-sets and construction equipment.

2.4 Investigated Softwares

2.4.1 Matlab/Simulink

Matlab is a language developed by Mathworks. It can be used to run simulations, optimize, calculations, analyzing etc. Simulink is a toolbox for Matlab, a block diagram environment that can be used for simulation of multi-dynamic systems. Simulink includes a large area of blocks in many areas. It is also possible to include Matlab functions into Simulink, which is a useful feature. Below a picture of a DC motor modeled in Simulink is shown. To run a simulation the green button with a play arrow is used [2].

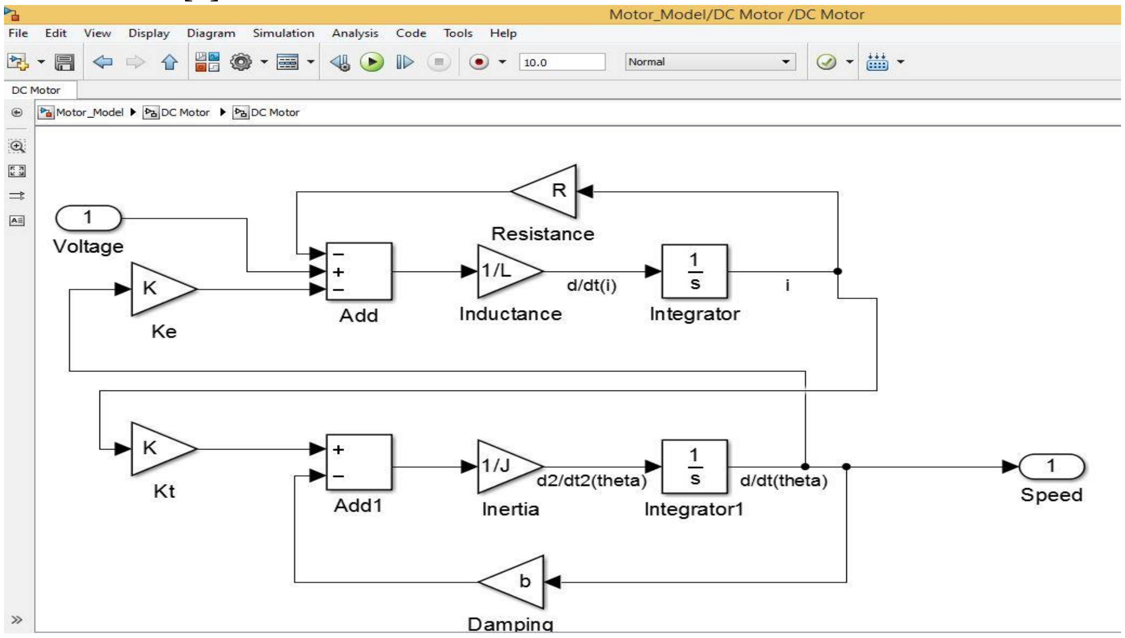


Figure 1: DC motor modeled in Simulink

2.4.2 Dymola

Dymola is commercial simulation software developed by Dassaut Systemes. Dymola modeling is based on Modelica Language. Dymola offers text based modeling as well as a graphical interface where models can be built. It is easy to display models thanks to the possibility to create own icons over blocks/models. Dymola also contains a simulation environment. When buying Dymola a large number of libraries in varies areas are included, e.g. electric power, hydraulics, engine dynamics, vehicle dynamics etc. In figure 2 below a simple DC motor is shown. As can also be seen the libraries are placed to the left in the Package Browser. To run simulations the window has to be changed in the bottom right corner (Simulation button) [3].

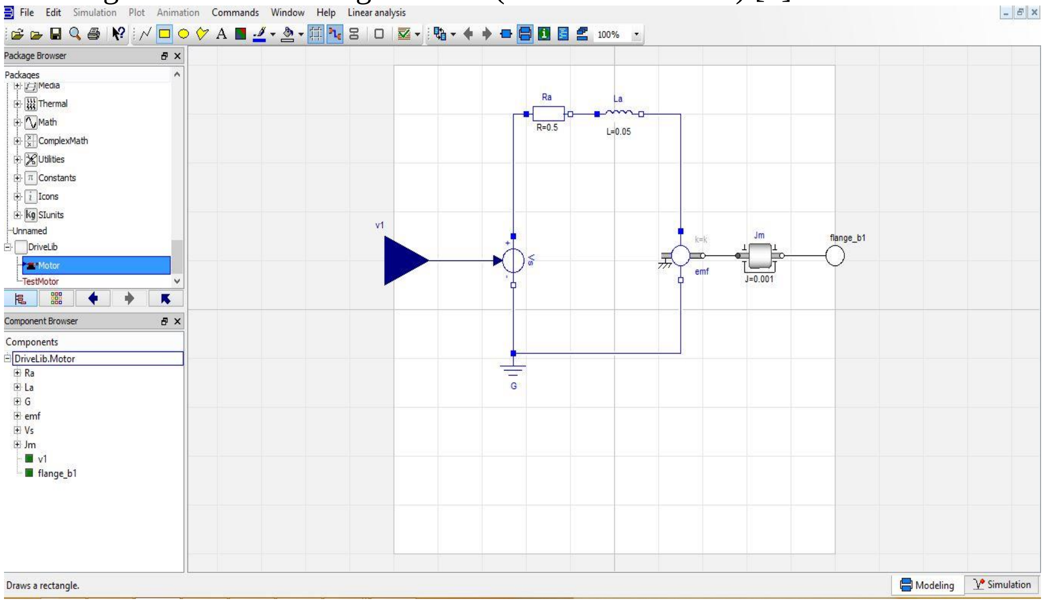
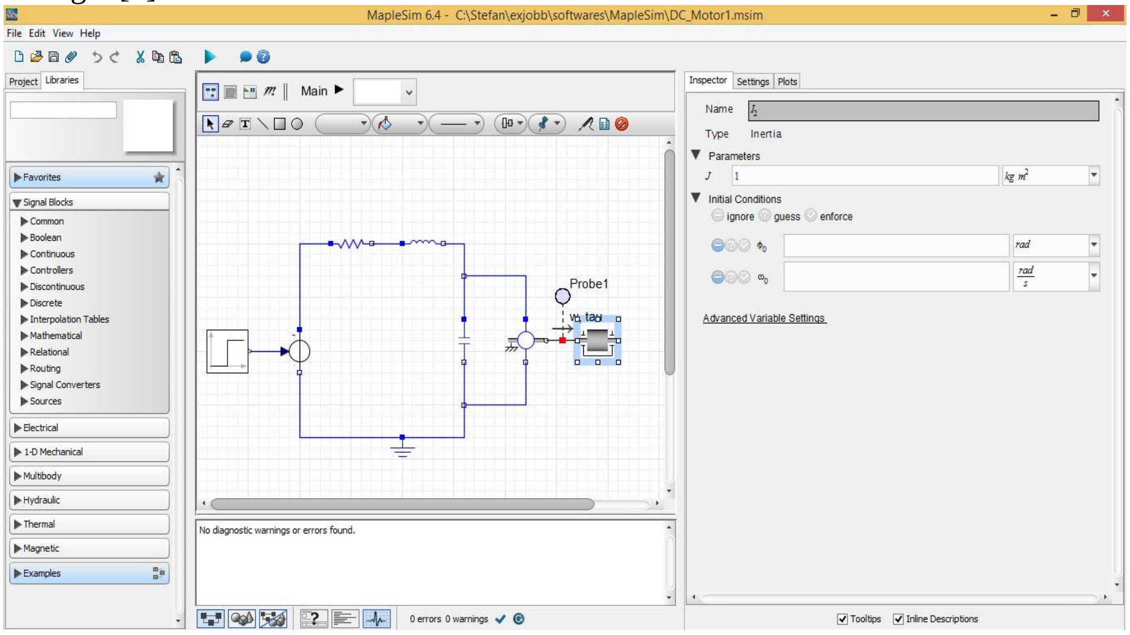


Figure 2: DC motor modeled in Dymola

2.4.3 MapleSim

MapleSim is a simulation tool developed by Maplesoft. It can be used to simulate advanced physical models in different areas like vehicle dynamics, electrics, airplanes, advanced hydraulics etc. It is also possible to use in optimization of designs.
Other features included are 3-D simulation animations, tire modeling, battery modeling and more.
Figure 3 shows how a model of a DC engine, together with the MapleSim graphical interface. The libraries are placed on the left side and parameters can be changed to the right [4].



Name J_2
Type Inertia
Parameters
J 1 $kg\ m^2$
Initial Conditions
ignore guess enforce
ϕ_0 rad
ω_0 $\frac{rad}{s}$
Advanced Variable Settings

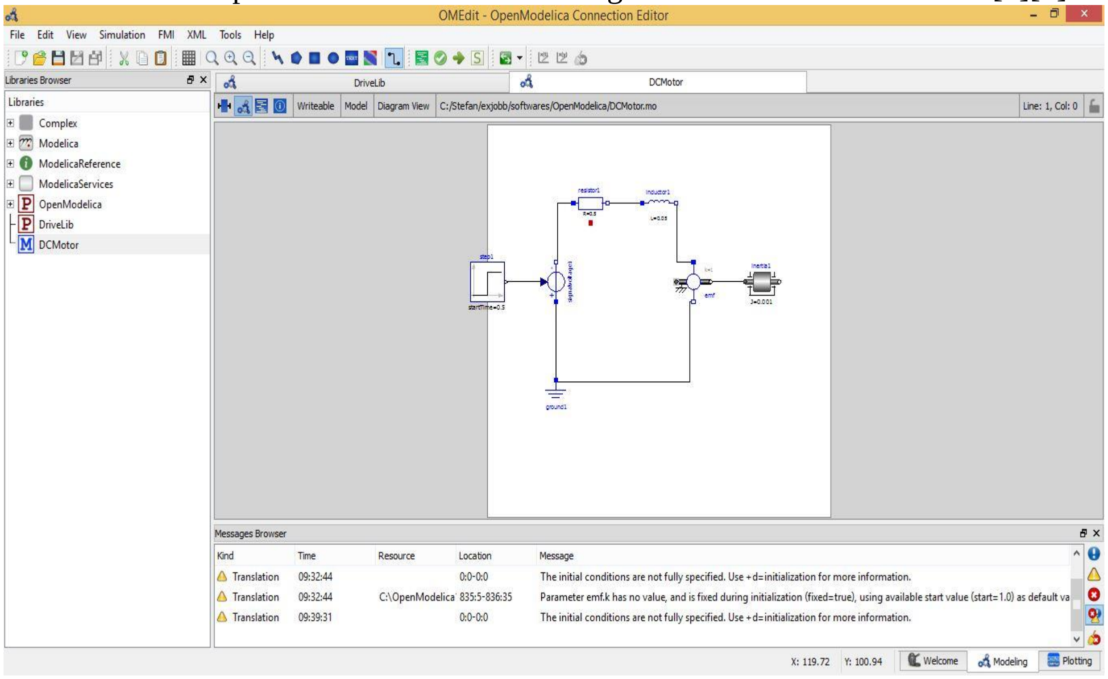
No diagnostic warnings or errors found.

0 errors 0 warnings

Figure 3: DC motor modeled in MapleSim

2.4.4 OpenModelica

OpenModelica is a tool that supports modeling with the language Modelica. OpenModelica is supported by Open Source Modelica Consortium (OSMC), a non-profit organization where a number of companies and universities are members. The aim of OSMC is developing and promoting OpenModelica, as well as provide support and maintenance.
It is free software intended for teaching, industrial and research usage. It includes modeling, compilation and simulation environment. Both text based and graphical interface modeling can be used.
Below is an example of an DC motor in OM together with the GUI interface [5][6].



Kind	Time	Resource	Location	Message
Translation	09:32:44		0:0-0:0	The initial conditions are not fully specified. Use +d=initialization for more information.
Translation	09:32:44	C:\OpenModelica'	835:5-836:35	Parameter emf.k has no value, and is fixed during initialization (fixed=true), using available start value (start=1.0) as default va
Translation	09:39:31		0:0-0:0	The initial conditions are not fully specified. Use +d=initialization for more information.

Figure 4: DC motor modeled in OpenModelica

2.4.5 Scilab

Scilab is open source software for numerical computations and is developed by Scilab enterprises. Scilab includes a large number of mathematical functions for simulations in varies areas.
Scilab also includes a graphical editor named Xcos. In this editor models/systems can be designed and simulated. Xcos includes a library with blocks in areas such as electrical, hybrid, math operations etc. Figure 5 below shows the graphical interface, a continuous function modeled and an example on how the result can be displayed. The libraries with blocks are, as can be seen, placed to the left [7].

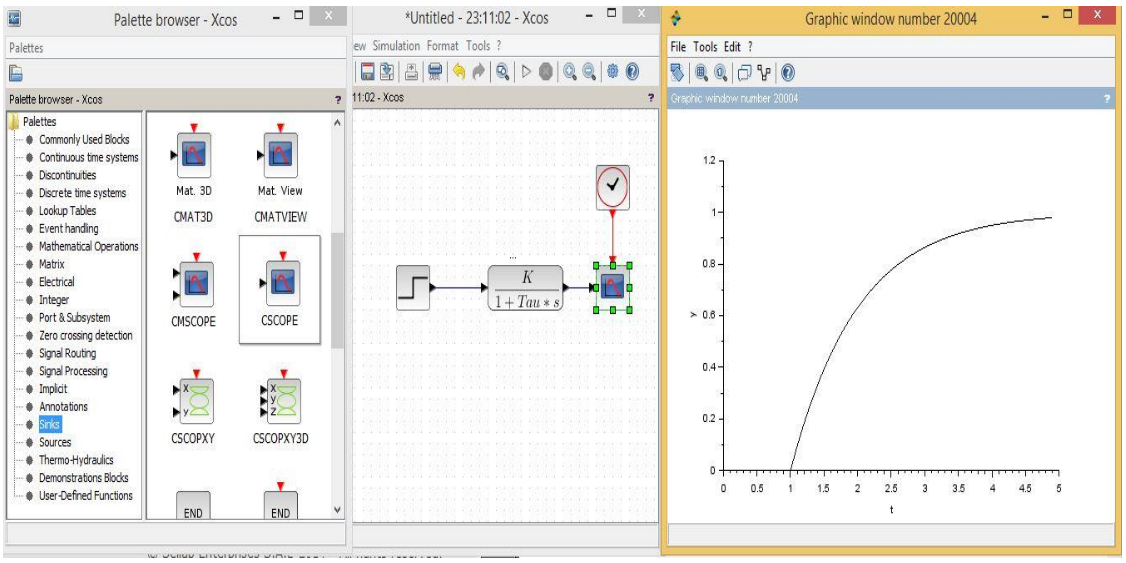


Figure 5: Continous function Scilab

2.4.6 Scicos

Scicos was originally developed at INRIA by Ramine Nikoukhah and is open source software. Scicos is distributed with the software package ScicosLab. ScicosLab is a platform toolkit for creating graphical user interfaces. ScicosLab includes a number of libraries and a simulation environment.
Below is a picture of a simple function in Scicos together with result window. Libraries/Palettes to the left [8].

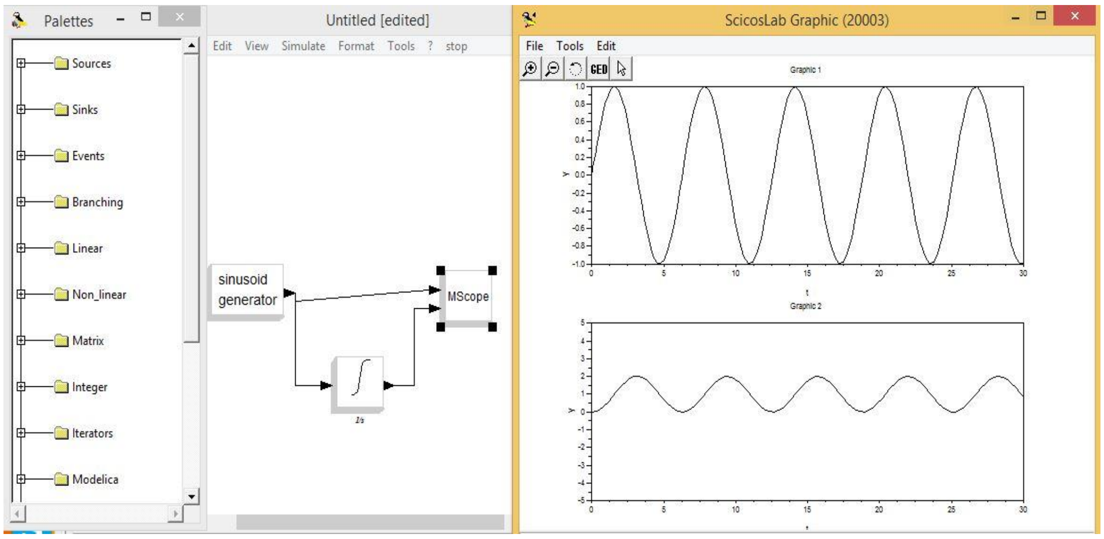


Figure 6: Simple function in Scicos.

2.4.7 Python

Python is open source software administrated by the Python Software Foundation. It is used in many applications, such as Web and Internet development, Education and Scientific and numeric computing. However does not Python offers any graphical interface, only text based modeling can be made [9].

2.5 Functional Mock-up Interface (FMI)

FMI is a standard that was initiated by Daimler AG. The purpose was to make it easier to exchange models between different tools, e.g. between suppliers and OEMs. With a FMI toolbox models can be exported into FMU (Functional Mock-up Unit), which is a zip archive containing an XML file with definition of variables used and other optional data, and then imported into tools/softwares that support FMI exchange [10].
FMI is the only standardized way to exchange models, and therefore a very suitable way to fulfill requitrement “Simulink compatibility” below in Table 1. The model can be imported in the form of an S-model, meaning that it is possible to change parameters in the model, but the code/equations behind the model will be hidden.
Some softwares has a FMI toolbox built in when installing the software, so import/export can be done directly. Other softwares, like Simulink, needs a FMI toolbox to be installed and a license to be bought.
It is also important to know that some softwares might have the possibility to import FMU models but not export, or the other way around. E.g. a FMU model is built and exported from software 2 and imported in software 1, but software 1 can not export a model build in software 1.
FMI standard is growing and more and more programs are joining.

FMI offers two formats for model exchange.

- FMI - ME (Model Exchange): is based on a continuous-time hybrid ODE reprensentation. Inputs and outputs are provided in the FMU - ME as well as fuctions of setting parameters. Simulation tools importing FMU - ME has to provide an integrator or ODE solver that integrates the dynamics of the model. [11].

- FMI – CS (Co-Simulation): here both the model and an integrator are encapsulated inside the FMU - CS. FMU - CS also provides inputs and outputs and means to set model parameters. An integrator function is also provided to integrate the dynamics of the model for a specified time interval [11].

2.6 Evaluation

To help making the decision about what tools to choose a Pugh Matrix was created. A Pugh matrix is a method used to choose the best of several alternatives in a systematic and objective way. To have in mind is that the below Pugh matrix is not an exact science, it is more aimed to give a help to decide what tools that looks promising for this thesis based on research and tests of the different simulation tools.

Table 1: Pugh matrix with criteria, simulation softwares and weighting

		Commercial				Open						
Criteria	**Weight**	**Matlab/Simulink**	**Dymola**	**Maple/Maple Sim**		**OpenModlica**	**Scilab**	**Scicos**	**Python**			
Low cost	0	0	0	0		1	1	1	1			
Built on international standards	2	0	1	0,25		0,5	0	0	0		🟥	uncertain info
De facto std	4	1	🟥 0	🟥 0		🟥 0	🟥 0	🟥 0	🟥 0,5		🟩	certain info
Modeling capability	4	1	1	1		1	🟥 1	🟥 1	0,5			
Simulink compatibility	2	1	0,75	0		0,75	0	0	0			
Stability and/or good diagnostics	2	1	🟥 0,75	🟥 1		🟥 0,5	🟥 0,5	🟥 0,5	🟥 0,75			
Graphic modular	4	0,75	1	1		0,75	0,5	0,5	0			
Other usability	4	1	1	1		0,5	0,75	0,75	0			
		19	**17**	**14,5**		**12,5**	**10**	**10**	**5,5**			

First, the criteria will be explained:

- *Low cost:* Refers to if the software is free or has a cost, open or commercial
- *Built on international standards:* Possibility to use FMU, import/export between different tools etc.
- *De facto std:* How common are these tools to use in companies, countries etc.
- *Modeling capability*: Is it possible to model e.g. both discrete and continuous dynamics, electrics and dynamics etc.
- *Simulink compatibility*: Is it possible to import/export to Simulink. This criterion is added because of the decision that Simulink will be one of the tools to use.
- *Stability:* How reliable is it, is the result from simulations trustworthy, etc.
- *Graphic modular:* Is it easy to change blocks, understand the built model etc.
- *Other usability:* Easy to use if never tried it before, good user manuals, understand menus, etc.

The criteria have been given a weight number that is different between some of them. This number basically tells if the criterion is more or less important.

There are also two colors incorporated in the Pugh matrix, green and red. Green is for certain information and red for uncertain info. This has been done because some information might not be as trustworthy as other. For example, cost is certain information, the tools cost or are free to download.

The two criteria that have been marked red in some of the tools are “De facto standard”, this because it is hard to find information about how commonly used these programs are around the world. A tool that isn’t very much used in Sweden (e.g. MapleSim) might be big in Asia or North America.

The second criterion that is red marked is “Stability and/or good diagnoses”. This is because even though all the tools (except Python) were tested, the models build and simulated were not very complex. How would these simulation tools work when modeling and simulation more complex models.

2.7 Chosen simulation tools

With the help of the Pugh matrix, tests and discussion with people involved in this thesis two main tools and one secondary tool have been chosen.

1. Matlab/Simulink: as explained before was this tool decided to be used already from the beginning.
2. OpenModelica (OM): OM will be the open source tool used in this thesis. It is a tool that gets bigger and bigger, and another positive aspect of this tool is its similarities with Dymola. Because both OM and Dymola are based on Modelica is it possible to exchange models between the two.
3. Dymola: Will be the secondary software. As well as being able to exchange models with OM it also has the possibility to communicate with Simulink by using FMUs. Only Dymola models built with Modelica format will be used, so these can be exported to OM. Models in Dymola built with other Dymola libraries can not be imported in OM.

There is nothing that hinders other secondary tools also, as long they support either of:

- Modeling in Simulink format, import and export on FMU format.
- Modeling in Modelica format, import ant export on FMU format.

Dymola and OM already has a buit-in FMI toolbox, that installs automatically when installing the softwares. They can both import and export FMU models.
Simulink does not have a FMI toolbox installed. However there is a FMI toolbox that can be installed to allow model exchange between Dymola and Simulink. As mentioned above Dymola can import/export with the buit-in toolbox. The Simulink FMI toolbox is just for Simulink FMU exchange, to be able to export FMU models from Simulink, or import to Simulink. For this a license is also needed to be bought from the company who created the tool box. The FMI toolbox for Matlab/Simulink is provided from Modelon, who have also provided an evaluation license for this thesis work.
Using this toolbox will be timesaving and instead of building same model two times one model can be built more complex.

3 Operating cycle

In order to perform simulations a operating cycle is needed.
A traditional operating cycle usually contains information about the allowed speed over a certain time or distance. The operating cycles used in this thesis work also contains information about altitude change during the cycle. With the altitude change a gradient of the road can be calculated. The road gradient is used when calculating the resistances of the vehicle.

The operating cycle used in this thesis is representing the road Gothenburg- Alingsas-Gothenburg.
Below are a figure of the driving route using Google Maps. The figure after shows altitude change and vehicle speed vs distance.

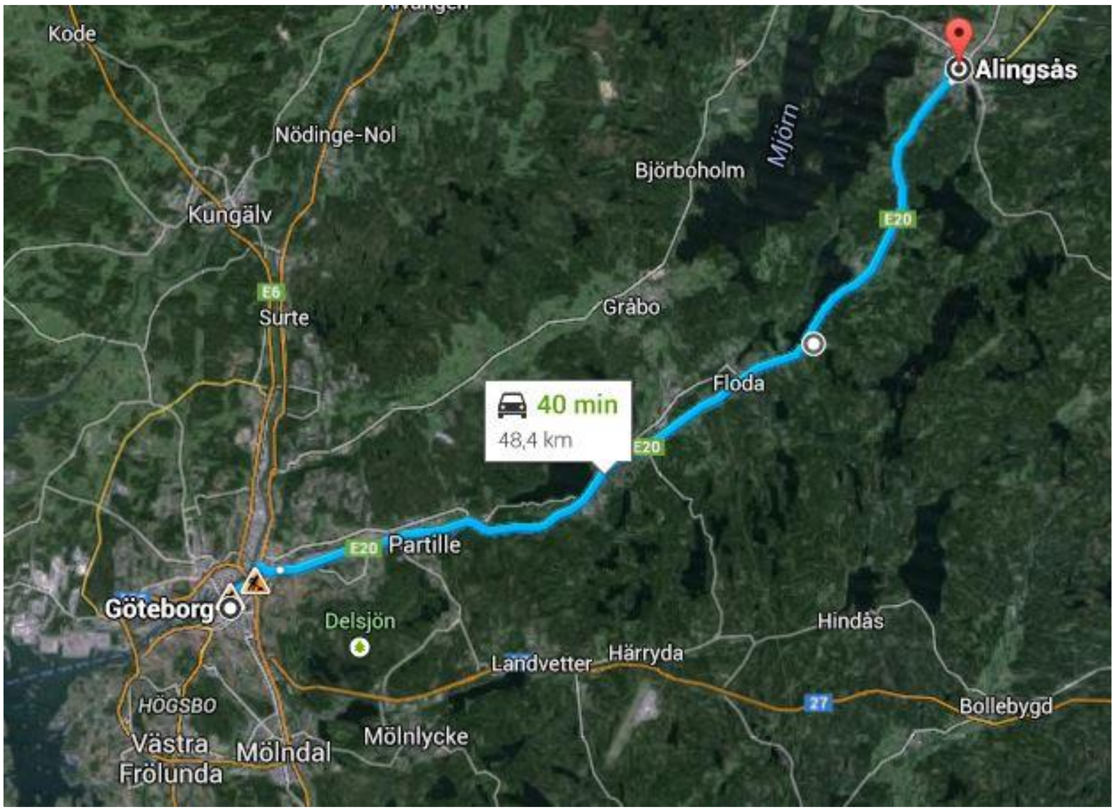


Figure 7: Driving cycle GAG using Google Maps

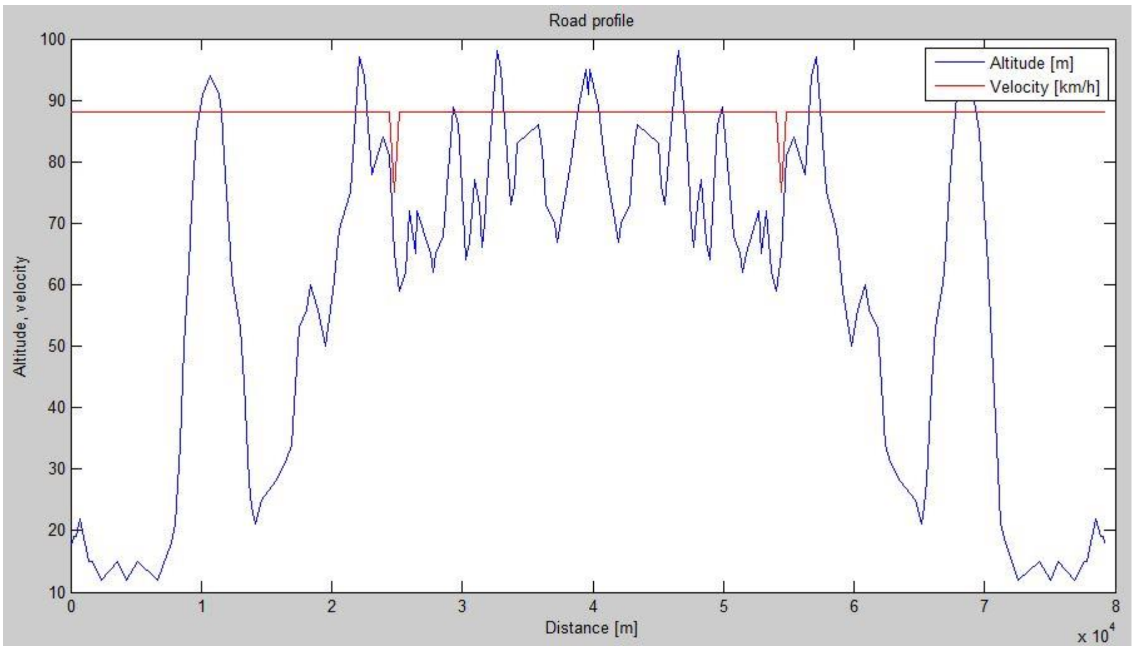


Figure 8: Altitude and velocity profile for GAG driving cycle

4 Vehicles

For the simulations one truck will be used. The chosen truck is a FH16 6x2 ridgid (svenska: jämnlastare) with air suspensions.

Figure 9: FH 16 truck

The data for the truck is shown in the table below.

Table 2: Vehicle data

Engine	DK13K540, 540 hk, 2600 Nm
Transmission	SPO2812
Gear ratios	11.73/9.21/7.09/5.57/4.35/4.41/2.70/2.12/1.63/1.28/1.00/0.78
Final Drive	3.61
Frontal area A	8.72 m^2
Drag coefficient C_d	0.7
Rolling resistance ff	0.0056
Tires	315_70_R225
Wheel radius r_w	0.49 m
Vehicle mass loaded	50 000 kg

Since the truck uses an automatic transmission it is necessary to know at what vehicle speeds the transmission will shift gear. This is possible to calculate with the use of the max engine torque for this engine and the gear ratios and final drive for the used transmission [12].
The max engine torque curve is shown in appendix 9.1. With the help of 16 points taken from the curve together with the corresponding engine speeds a v-F_t diagram for each gear can be computed and the speed at where the shifting should be done chosen from the diagram. The equations used for this was:

$$\mathrm{F} = \mathrm{T}_{\mathrm{e,max}}(\omega_{\mathrm{e}}) * \mathrm{i}_i * FD / r_w \quad (1)$$

$$\mathrm{v} = \omega_{\mathrm{e}} * r_w / (\mathrm{i}_i * FD) \quad (2)$$

Where:
F = Traction force [N]
v = vehicle speed [m/s]
$T_{e,max}$ = max torque [Nm]
$\omega_{e,max}$ = max engine speed for corresponding torque [rpm]
i_i = gear ratios for a gear I =1, 2, 3, … [-]
FD = Final drive ratio [-]
r_w = wheel radius [m]

With equations 1 and 2 above the traction force and vehicle speed will be calculated for one point chosen from the max torque curve, with a selected gear ratio. The easiest way to calculate the v-F plot for all gears for all points of the torque curve is to use vectors; one vector with points from the max torcue curve, one vector with the corresponding engine speed and one vector with the gear ratios. Put these vectors in equations 1 and 2 and the result will be as shown in Figure 10 below.

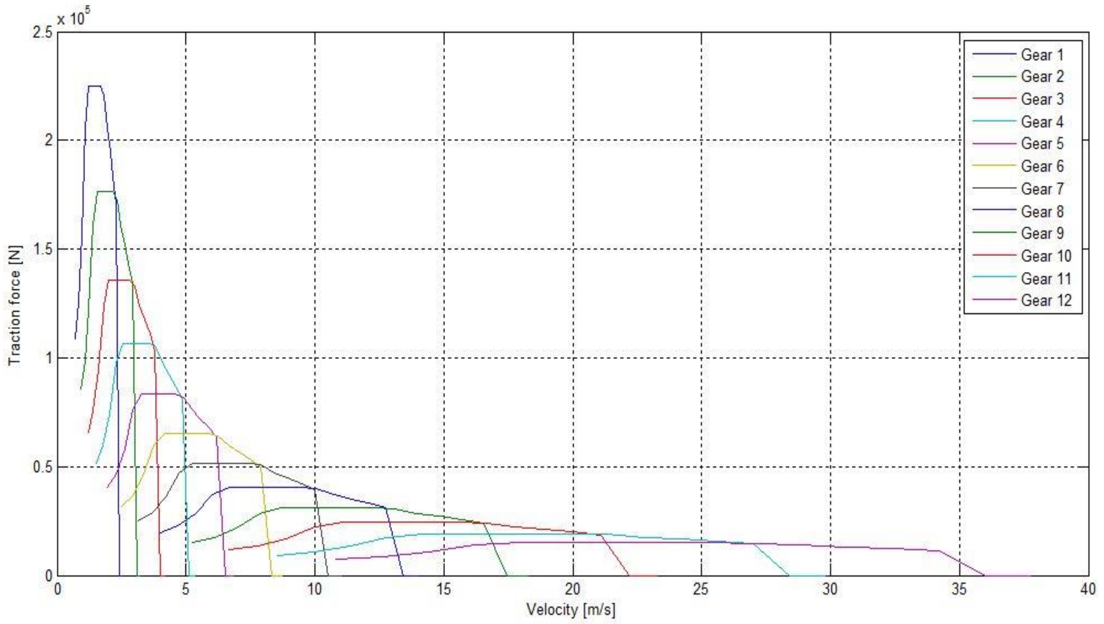


Figure 10: v-F plot for all gears

The plot above shows which range each gear works in. From here the shifting speed for the gears can be chosen. A suitable point to shift gear for maximize propulsion is around the point when the curve from one gear crosses the curve of another gear. The shifting speed was chosen in this way and are as shown in Table 3. This data will be used when building the model of the transmission in the simulation softwares.
An example on how to interpret the table is: the speed to shift from gear 1 to gear 2 is 2.32 m/s. The speed to shift from gear 2 to gear 1 is 2.16 m/s.

Table 3: Velocity for gear shifting up and down.

Gear	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Upshift [m/s]	2.32	2.97	3.84	5	6.27	7.96	10.2	13	16.7	21.25	27.3	N/A
Downshift [m/s]	N/A	2.16	2.85	3.7	4.6	6	7.5	9.5	12.2	15.7	20	24.9

5 Model design

In order to run a simulation of a vehicle, three systems are needed: Operating cycle that includes road data, driver behavior that tells how aggressive or carefully the vehicle is driven, and a vehicle model, including engine, transmission and chassis. The driver model is here a close-loop controller, which tries to make the vehicle's actual speed follow a set speed.
This kind of simulation is called a forward simulation. The model also include feedback systems representing real life operations. For example, a driver in the real world will see the speed limit of a certain road and try to follow this speed by adjusting with the acceleration/brake pedals. In the simulation model this is done by comparing the allowed speed from the operating cycle with the actual speed, coming from the chassi block. This would be a feedback system. The figure below summarizes how a system can be structured.

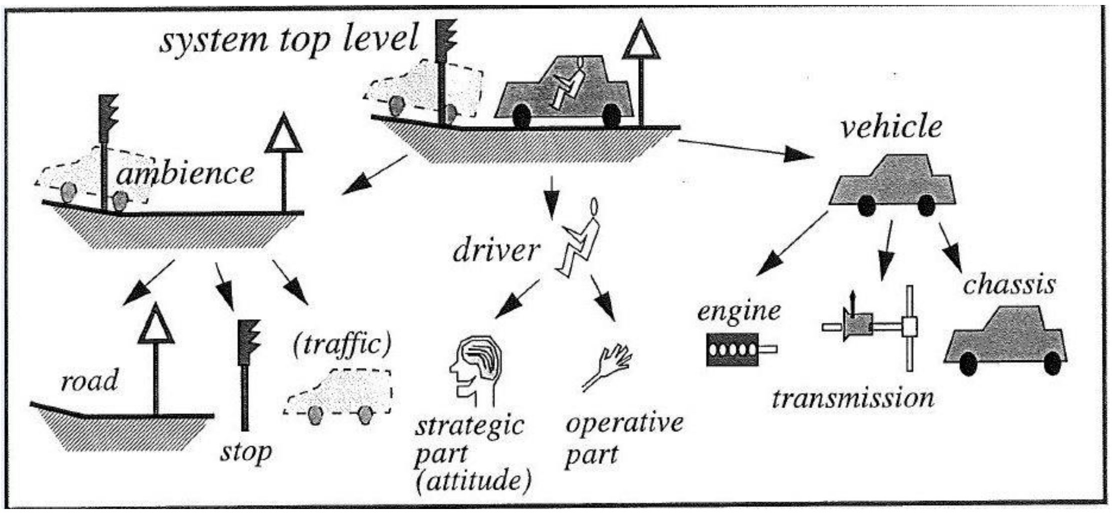


Figure 11: System composition [13]

Figure 13 on page 21 shows how the full model looks like.

There are four simulation models that have been modeled, one in Simulink, one in OM, one in Simulink where some sub models have been changed to FMU parts and one in OM where FMU parts have been used.
The table below explains the combinations and also what parts that are FMU in the models. S = Simulink, OM = OpenModelica and D = Dymola.

Table 4:Simulation combinations.S = Simulink,OM = OpenModelica, M = Modelica,D = Dymola.

				Modelling Tool/Format			Simulation tool
nr	**testname**	**Op Cycle**	**Driver**	**Engine**	**Transmission**	**Chassi**	
1	S	S/S	S/S	S/S	S/S	S/S	S
2	OM	OM/M	OM/M	OM/M	OM/M	OM/M	OM
3	D -> S	S/S	D/M	S/S	D/M	S/S	S
4	S -> D	D/M	D/M	D/M	S/S	S/S	D
2b	D	D/M	D/M	D/M	D/M	D/M	D

Some of the submodels in Simulink have been built by directly using mathematical equations. These sub models are the operating cycle, Transmission and Chassis. These equations will be explained in chapter 5.1.1 – 5.1.3

5.1.1 Operating cycle

The operating cycle concists of distance, wanted speed and altitude. The speed and distance is obtained by loading the data from the operating cycle file. The road gradient is obtained by using triogonometry. By taking the altitude at the actual distance x meter driven by the vehicle and then take the altitude at a distance x - wheel base meter before the actual distance the gradient is deduced. Visually this is shown in figure below, the wheel base is set to 30 m in this figure

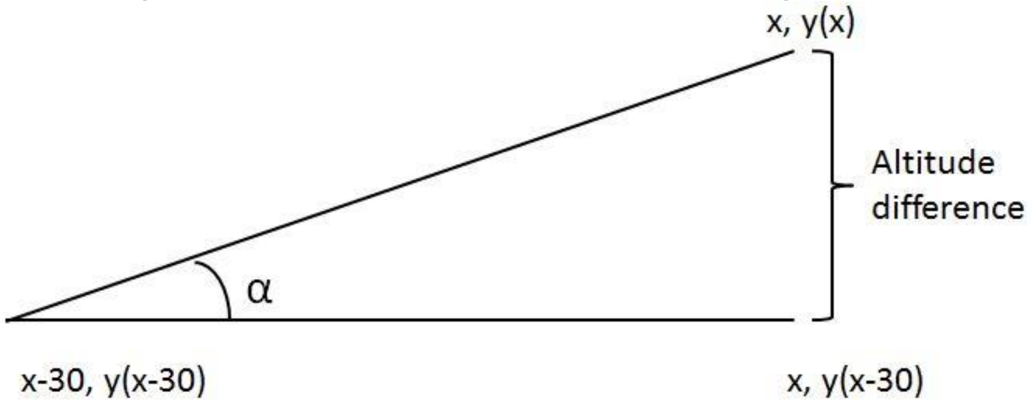


Figure 12: Calculating gradient α by triogonometry

The equation used to get α is:

$$\alpha = \arcsin\left(\frac{y(x) - y(x-30)}{30}\right) = a\tan\left(\frac{\Delta y}{30}\right)$$

5.1.2 Driver

The driver is the one controlling the acceleration/brake pedals and thus controlling the speed. By comparing the speed of the road with the speed the vehicle currently is driving in adjustments are done with the pedals to try to keep the wanted/allowed speed. In a model this would be done with two input signals, wanted speed v_set and actual speed v_is. The output signals is pedal positions of either the accelerator pedal or brake pedal.

5.1.3 Vehicle

5.1.3.1 Engine

When modelling engines in simulation softwares usually engine maps are used. There are a number of different ways of modelling engine maps. Volvo Trucks uses a fueling map where the input is given in g/ml and the same has been used in these models. However, the engine map that is used in these models are not the correct one for the engine DK13K540. The max torque curve however is the one for this engine. No moment of inertia has been included in the engine model, this because this would have made it to complex [14].

5.1.3.2 Transmission

The transmission uses the engine torque to calculate what the wheel torque is with the corresponding gear. The way this is done is shown in below equations:

$$T\_w = T\_e * i_i * FD * \eta_t$$

Where

T_w = wheel torque

T_e = engine torque

i_i = gear ratio for the gear i used

FD = final drive

η_t = transmission efficiency

The second output calculated from the transmission is the engine speed. This is done by using the actual speed v_is as follows:

$$\omega_e = \frac{v_{is}}{r_w} * i_i * FD$$

Where r_w = wheel radius

5.1.3.3 Chassi

In the chassi the acceleration or deceleration of the vehicle is calculated. The calculation is as follows:

$$a = \frac{\frac{T_w}{r_w} - F_{res}}{m}$$

The driving resistance consists of drag resistance F_a, rolling resistance F_r and road inclination resistance F_g and is calculated as below [12]:

$$F_{res} = F_a + F_r + F_g = \frac{C_d * \rho_{air} * A * V_{rel}}{2} + m * g * \sin(\alpha) + m * g * f_f * \cos(\alpha)$$

Where
m = mass of vehicle
C_d = drag coefficient
A = frontal area of wheicle
f_f is rolling resistance coefficient
V_{rel} = relative speed, meaning v_is – v_air

5.2 Simulink model

A short explanation of how Simulink starts the simulation is neccesary to better understand what is happening. When the “run” button is pressed in Simulink, the program looks for a value to “start” with. In this model it finds two integrator blocks. One integrator block is used in the chassi model (see Figure 19, page 25). This block integrates the acceleration to actual speed, with initial condition set to 0 m/s.
The second integrator block is used in the vehicle block, and integrates the vehicle speed to distance driven. The initial condition is set to 0. This value will then be fed into the input port of the operating cycle, that will look for the corresponding speed in the road data matrix. Simulink uses both of these “starting” values to initiate the simulation and calculate the other parameters.
It is also important to remember that if parameters are named by letters instead of numbers in Simulink they must be initiated before starting the simulated. This can be done by using an m-file.

The three submodels that together forms the full simulation model are the operating cycle, the driver and the vehicle.

Operating cycle: The input signal to the operating cycle is distance travelled. The distance tells where in the cycle the vehicle is. The distance is derived from v_is, that is the actual speed. The output signals are the gradient alfa and the target speed (v_set).

Driver: The driver has two input signals, v_set and v_is (actual speed) and two output signals, Accelerator Pedal and Brake Pedal. These values are in percentages and denote how much the pedals are used.

Vehicle: The input ot the vehicle model is the road gradient alfa, the accelerator pedal and brake pedal actuation. Output is v_is, the actual speed the vehicle has and the distance the vehicle has travelled.

In Appendix 9.2 a list of all the abbreviations used in the Simulink model is shown and explained.

Below is a figure of the Simulink model

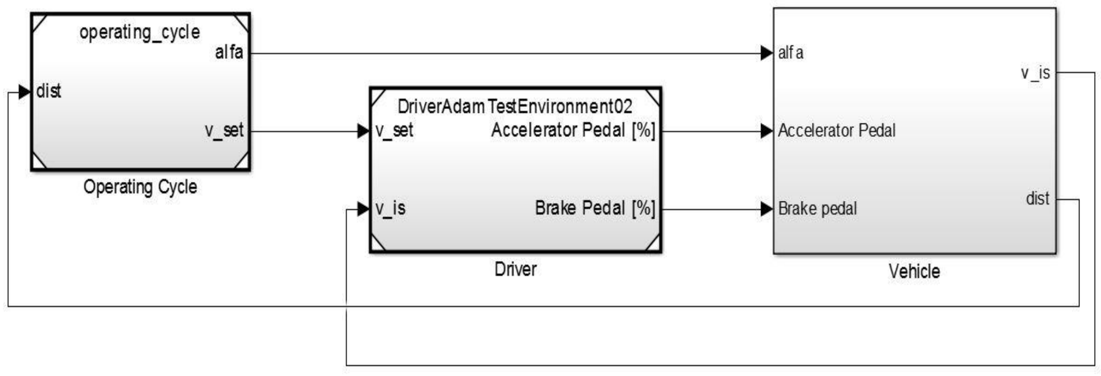


Figure 13: Full model in Simulink

5.2.1 Operating cycle

The operating cycle is built up as can be seen in Figure 14.

With the help of the input signal distance a Lookup Table is used to find the corresponding allowed speed v_set. The Lookup Table uses linear interpolation to calculate the matching speed at that specific distance.

The slope angle block is where the gradient is being calculated using the method explained in chapter 5.1.1.

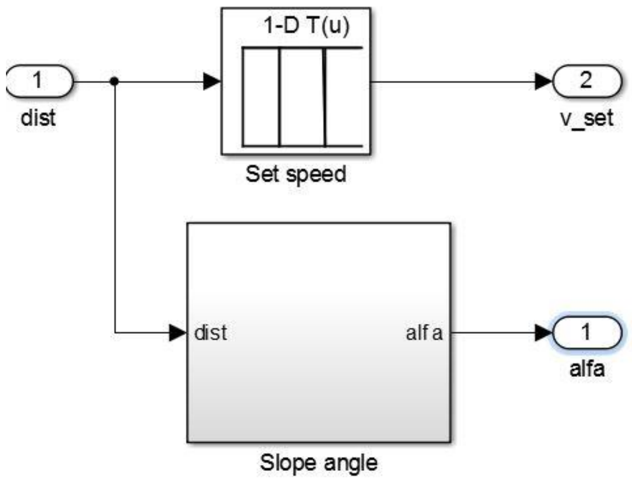


Figure 14: Operating cycle build-up

5.2.2 Driver

Below the driver model is shown.The input signals v_set and v_is are compared in a sumblock and the result is a error value that is fed into a PID Controller. The PID controller consists of three gainblocks, one integrator and one derivate block and finally one block to sum the three signals. The PID Controller evaluates the error value and gives an output between -100 to 100. If the value is between 0 to 100 the accelerator pedal will be engaged to that amount and in the value is between -100 to 0 the same goes for the brake pedal.

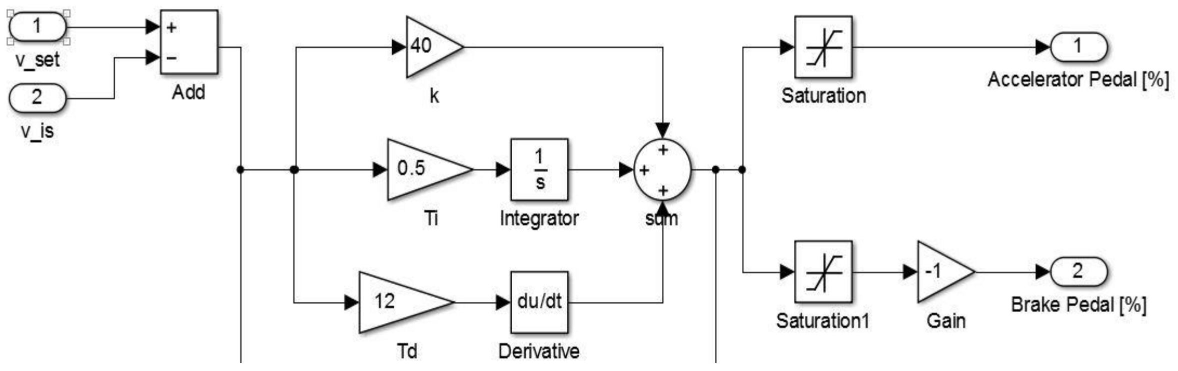


Figure 15:Model of test driver Adam

5.2.3 Vehicle

The vehicle model is as mentioned before built of engine, transmission and chassi. A tank block is also added to measure fuel consumption in l/10km.

Engine: The input to the engine is the accelerator pedal position and the engine speed. The output signal is engine torque and a fuel consuption value expressed in l/sec.

Transmission: The input signal to the transmission is engine torque, gear and the actual speed v_is. Output is drive shaft torque (wheel torque) and engine speed.

Chassi: The cassi gets three input signals, road gradient alfa, wheel torque and brake pedal position. Output is the v_is (actual speed) and the distance travelled.

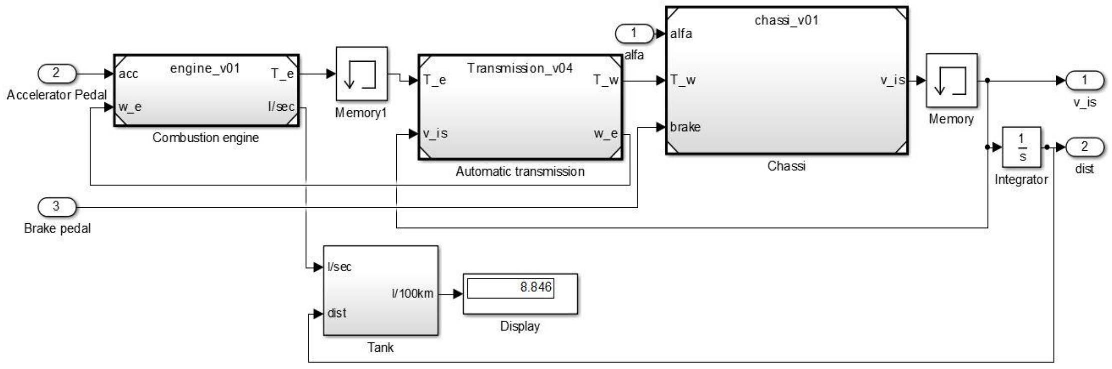


Figure 16: Vehicle model

5.2.3.1 Engine

As can be seen in Figure 17 the engine model consists of several blocks. The input signal engine speed is involved in all of these. The blocks pedal map, max torque curve, fueling and steady state torque map are all built up by a Lookup Table. The input signal engine speed is the x-axis value in all of these.

The pedal map lookup table consists of a 11x11 matrix from where the required torque can be derived with the help of the current engine speed and accelerator pedal position. This value is then compared to the max torque for the same engine speed, so that the required torque is not higher than the max torque. The required torque is then the input in the fueling block.

The fueling block also consists of a lookup table that has a 11*11 matrix. The 11*11 matrix consists of the q-value, which is the amount of fuel used per stroke, expressed in mg/stroke.
The lookup table uses the current engine speed and the required torque to find the matching q-value.

The steady state torque maps works the same way as the two mentioned above. It has a lookup table with a 11*11 matrix. The input engine speed and q-value is used to derive what the Steady state torque and gives this as an engine torque output.

The q-value is also used to obtain a fuel consumption rate in l/sec.

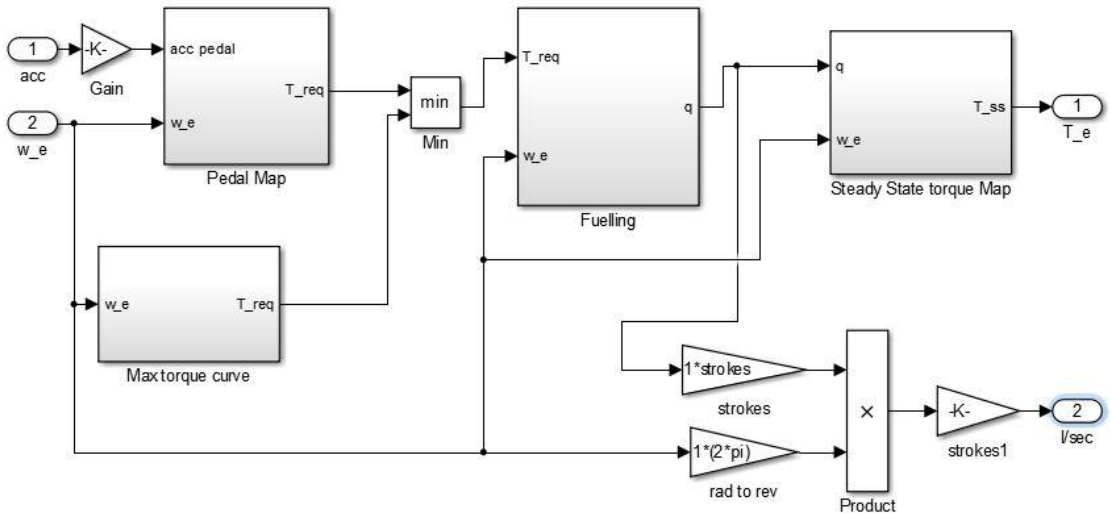


Figure 17: Engine model

5.2.3.2 Transmission

The transmission used is an automatic transmission and is modelled as shown below.

Because it is an automatic transmission the gear is decided depending on the actual speed v_is. The lookup table is where the gear is decided and gives the gear ratio as output signal. With the help of the gear ratio and final gear the torque to the wheels are calculated. The actual speed is also used to calculate the engine speed. It was explained how in chapter 5.1.2.1.

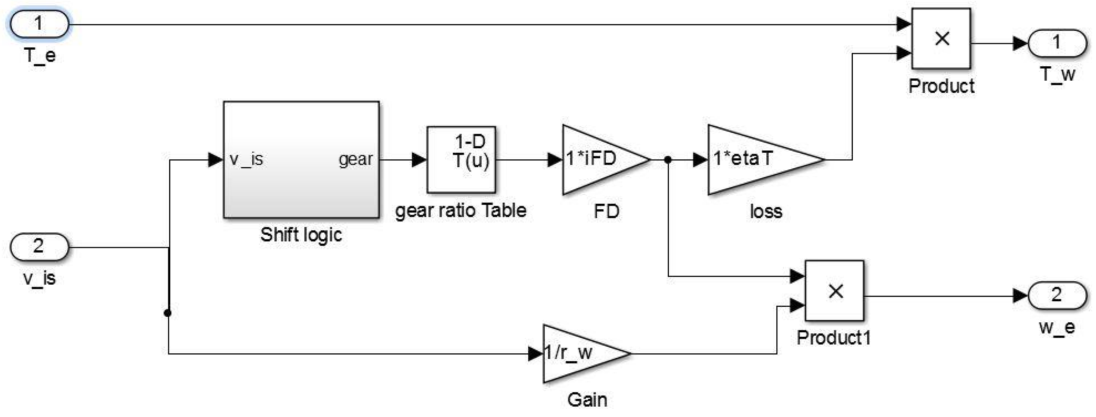


Figure 18: Automatic transmission model

5.2.3.3 Chassi

The last submodel is the chassi shown below in Figure 19.

The model might look like a mess but is basically built accordingly to the equations explained in chapter 5.1.2.2.
The resistance forces obtained from slope inclination, road friction and aerodynamics are added together and then substracted from the wheel force. The acceleration can then be calculated and from that the actual speed v_is.
The second input signal to the chassi is the brake pedal position. With this the brake torque can be obtained. If the brake pedal is used a negative torque will help the

vehicle to slow down faster. An example of when a brake torque will be used is if the actual speed v_is is bigger than the allowed speed v_set.

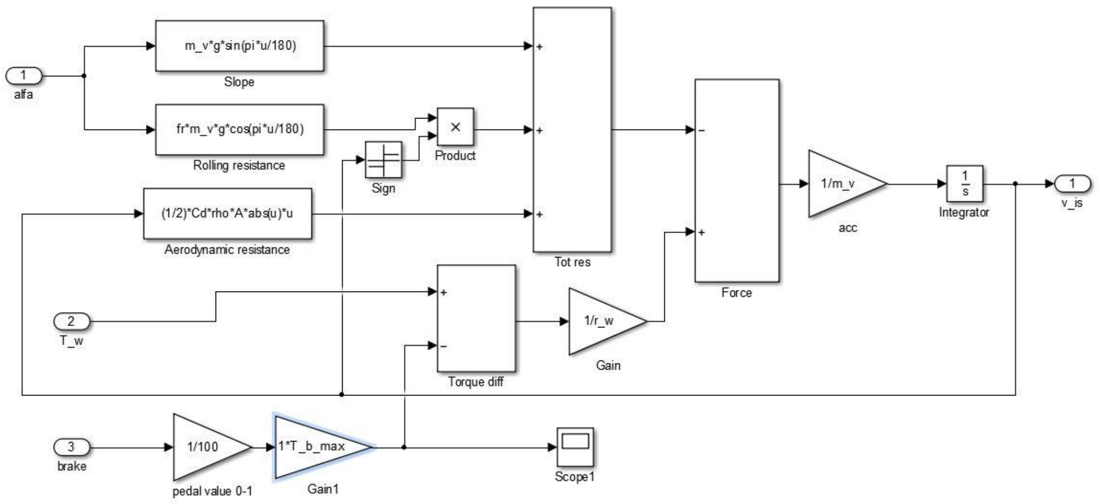


Figure 19: Chassi model

5.3 Open Modelica model

The simulation model built in Open Modelica (OM) is basically modelled in the same way as the Simulink model. The full model consists of operating cycle, driver, engine, transmission and the chassi. The inputs and outputs from the submodel are the same. The full model is shown below.

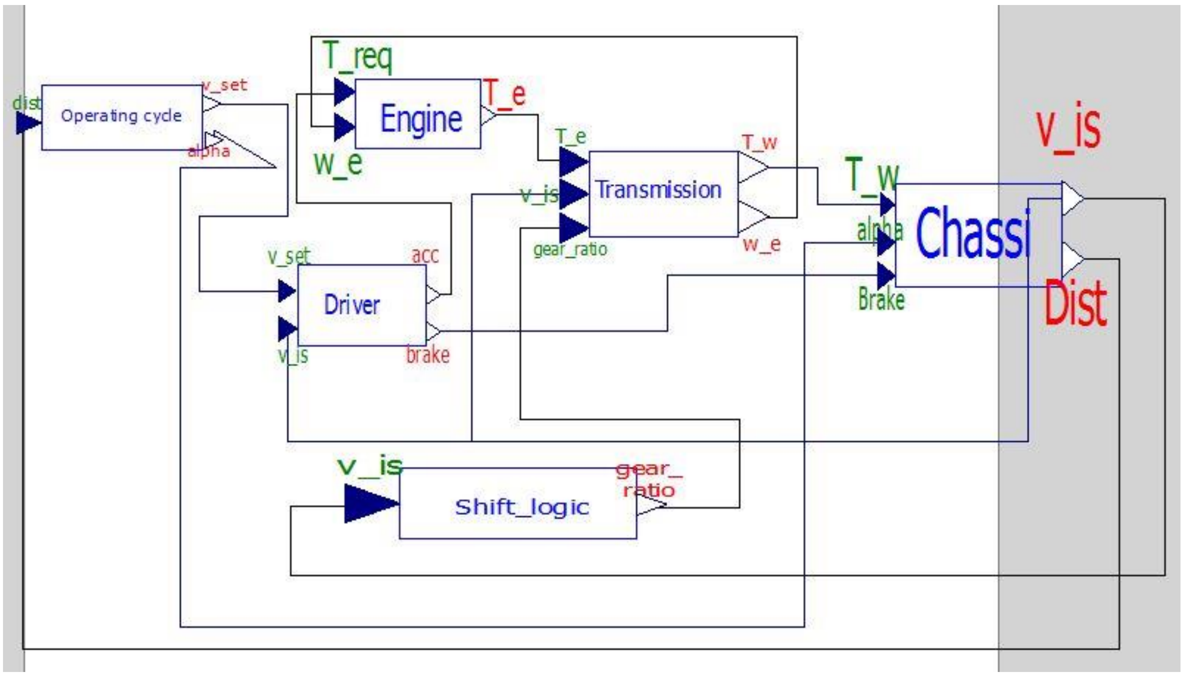


Figure 20: Full simulation model in Dymola

5.3.1 Operating cycle

The operating cycle consists of an input signal “distance” and two output signals, v_set and alpha. The operating cycle has two tables, in OM called combitable1D, that works in the same way as look-up tables does in Simulink. It uses interpolation to derive correct output signal based on the input signal. One table that loads the desired speed v_set and one with the inclination It works in the same way as the model in Simulink.

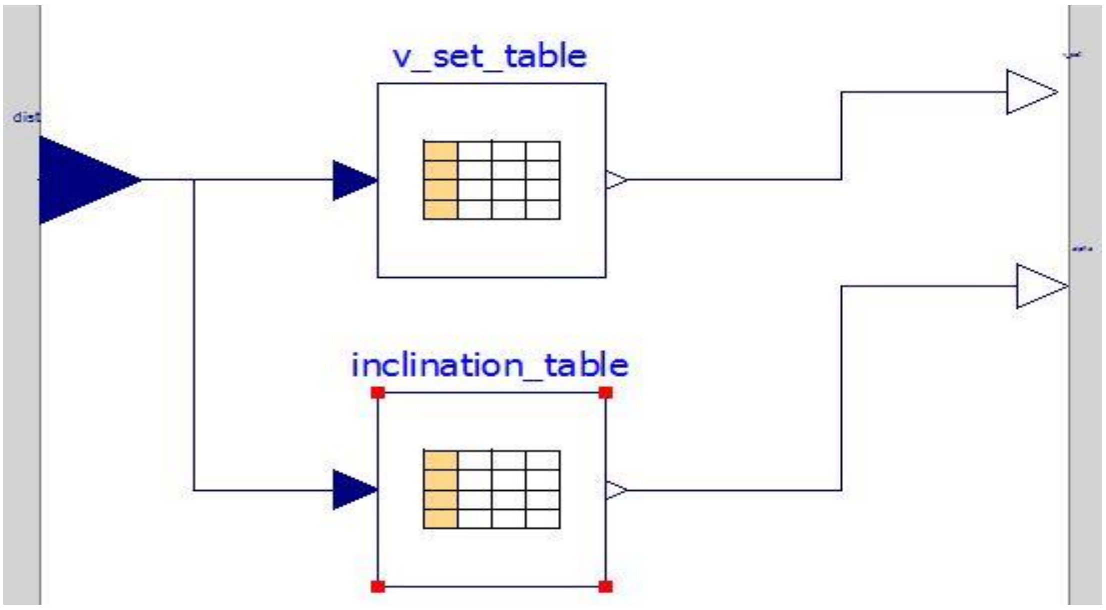


Figure 21: Operating cycle Dymola

5.3.2 Driver

The driver model in OM uses a PID controller that is built of gain-, integrator-, and derivator blocks. The PID compares the two input signals, v_set and v_is, to give an error value which is fed to either the accelerator pedal or the brake pedal depending on if it is of positive or negative value. Model is shown in Figure 22 on the next page.

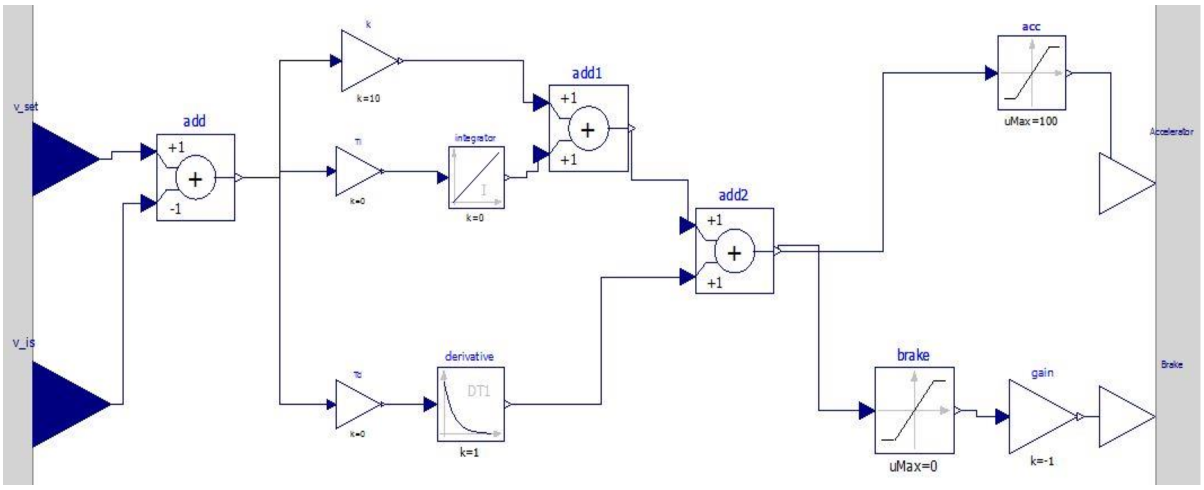


Figure 22: Driver model in Dymola

5.3.3 Engine

The engine is built-up of several 2D combitables. As with 1D tables the 2D tables uses interpolation to calculate the output signal. The difference with the 1D table is that the 2D table has two input signals. Three 2D tables and one 1D table is used in the engine model, and each 2D table includes a 11x11 matrix. The matrices, input values to the tables and outputs from the tables are exactly the same as in the engine model built in Simulink, explained in chapter 5.2.3.1.

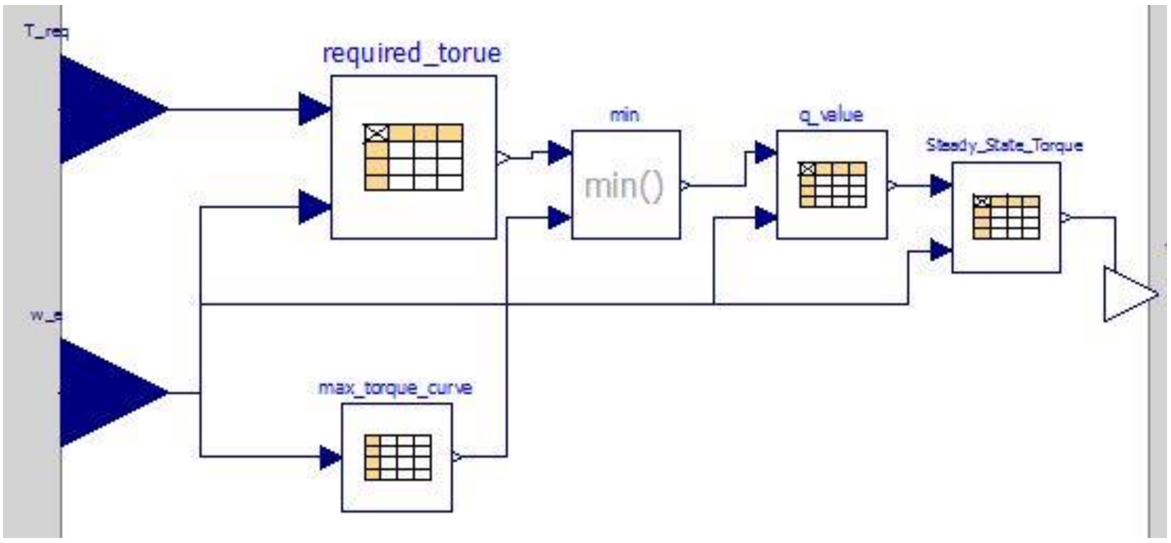


Figure 23: Engine model in Dymola

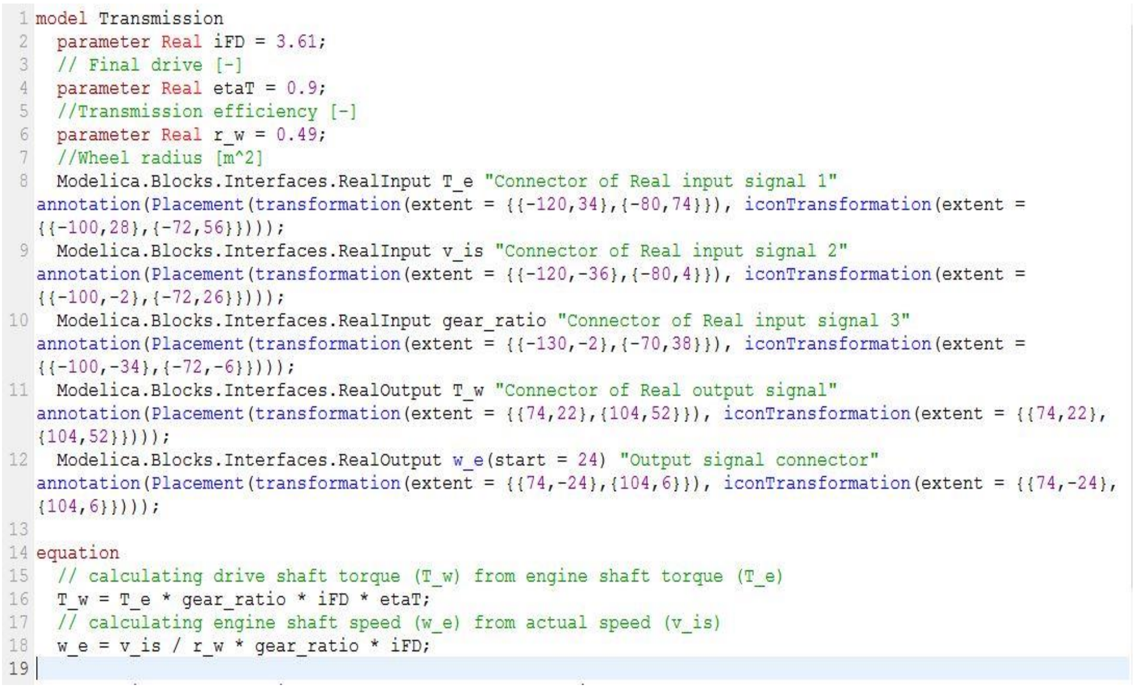
5.3.4 Transmission

The transmission model in OM is built in a different way compared to Simulink. Instead of using blocks Modelica text is used. The Modelica text is built-up in the following steps:

1. Parameters used in equations later in the text are defined and given values
2. Input and output signals are defined
3. Equations are written

For step one and two it does not matter which one that is coded first. However, equations are always written after parameters are defined.

There are two equations used in the transmission model, one that calculates wheel torque and one that calculates engine speed. Input signals are engine torque and actual speed. Output signal are wheel torque and engine speed.



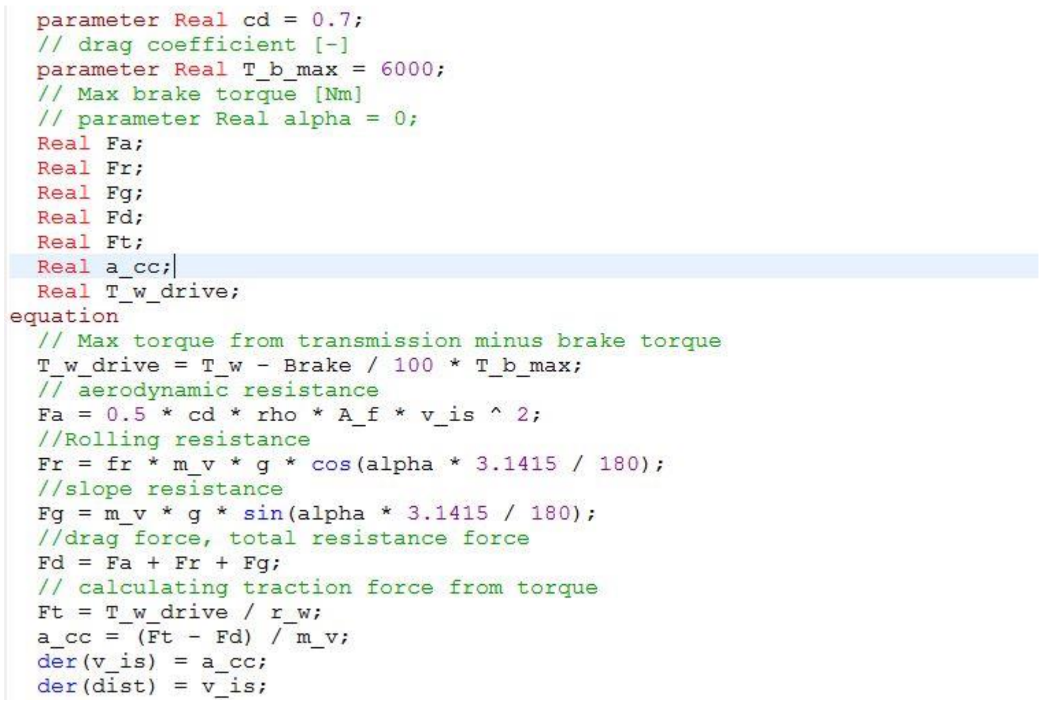
```
model Transmission
  parameter Real iFD = 3.61;
  // Final drive [-]
  parameter Real etaT = 0.9;
  //Transmission efficiency [-]
  parameter Real r_w = 0.49;
  //Wheel radius [m^2]
  Modelica.Blocks.Interfaces.RealInput T_e "Connector of Real input signal 1"
annotation(Placement(transformation(extent = {{-120,34},{-80,74}}), iconTransformation(extent =
{{-100,28},{-72,56}})));
  Modelica.Blocks.Interfaces.RealInput v_is "Connector of Real input signal 2"
annotation(Placement(transformation(extent = {{-120,-36},{-80,4}}), iconTransformation(extent =
{{-100,-2},{-72,26}})));
  Modelica.Blocks.Interfaces.RealInput gear_ratio "Connector of Real input signal 3"
annotation(Placement(transformation(extent = {{-130,-2},{-70,38}}), iconTransformation(extent =
{{-100,-34},{-72,-6}})));
  Modelica.Blocks.Interfaces.RealOutput T_w "Connector of Real output signal"
annotation(Placement(transformation(extent = {{74,22},{104,52}}), iconTransformation(extent = {{74,22},
{104,52}})));
  Modelica.Blocks.Interfaces.RealOutput w_e(start = 24) "Output signal connector"
annotation(Placement(transformation(extent = {{74,-24},{104,6}}), iconTransformation(extent = {{74,-24},
{104,6}})));

equation
  // calculating drive shaft torque (T_w) from engine shaft torque (T_e)
  T_w = T_e * gear_ratio * iFD * etaT;
  // calculating engine shaft speed (w_e) from actual speed (v_is)
  w_e = v_is / r_w * gear_ratio * iFD;
```

Figure 24: Transmission in Dymola built with Modelica text

5.3.5 Chassi

The chassi is also built using Modelica text in OM. Here the input and output signals are defined first, following with the parameters. After that comes the equations.
The total resistance is calculated, substracted from the traction force and devided with the vehicle mass in order to calculate the acceleration. The acceleration is integrated to get the actual vehicle speed, and integrated again to get the travelled distance. The actual vehicle speed and distance travelled are output signals.



```
  parameter Real cd = 0.7;
  // drag coefficient [-]
  parameter Real T_b_max = 6000;
  // Max brake torque [Nm]
  // parameter Real alpha = 0;
  Real Fa;
  Real Fr;
  Real Fg;
  Real Fd;
  Real Ft;
  Real a_cc;
  Real T_w_drive;
equation
  // Max torque from transmission minus brake torque
  T_w_drive = T_w - Brake / 100 * T_b_max;
  // aerodynamic resistance
  Fa = 0.5 * cd * rho * A_f * v_is ^ 2;
  //Rolling resistance
  Fr = fr * m_v * g * cos(alpha * 3.1415 / 180);
  //slope resistance
  Fg = m_v * g * sin(alpha * 3.1415 / 180);
  //drag force, total resistance force
  Fd = Fa + Fr + Fg;
  // calculating traction force from torque
  Ft = T_w_drive / r_w;
  a_cc = (Ft - Fd) / m_v;
  der(v_is) = a_cc;
  der(dist) = v_is;
```

Figure 25: Chassi model in Dymola

5.4 Simulink with FMU sub models

In test 3 the driver and transmission sub model has been changed to FMU parts. The FMU format to import with in Simulink is FMU – ME as it was exported as this from Dymola. As can be seen in Figure 26 the FMU submodel are seen as a white box.

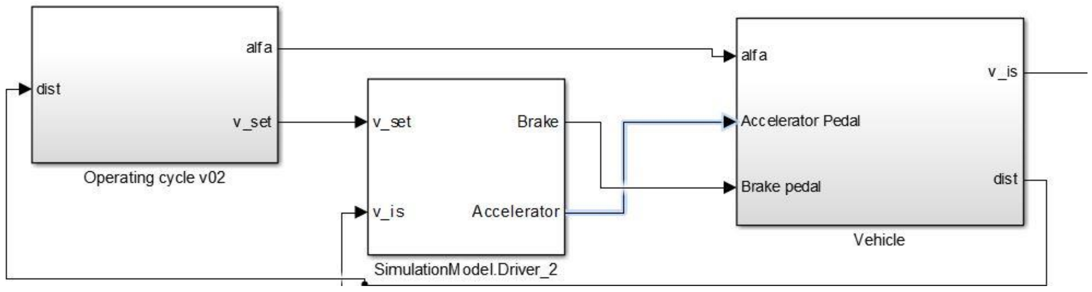
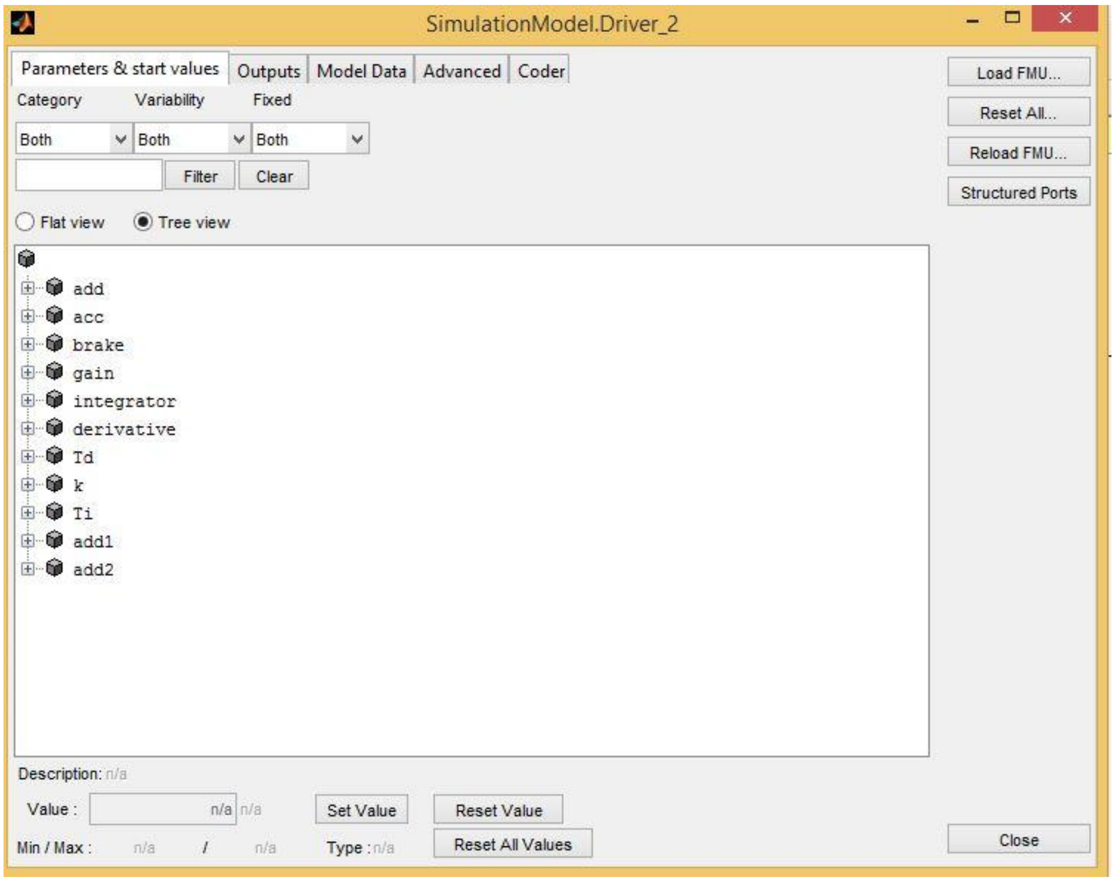


Figure 26: Full model with FMU driver in Simulink

The vehicle that contains the transmission has the same look, the transmission is a white box instead of the light grey that the other models have.

When opening a FMU part in Simulink the parameters defined in the model before it was exported can be changed. Under the tab "Output" signals can be added or deleted. Se figure Figure 27 on page 30 for a view of the FMU sub model.



```
add
acc
brake
gain
integrator
derivative
Td
k
Ti
add1
add2
```

Figure 27: View of a FMU imported and opened in Simulink

5.5 Dymola with FMU sub models

In test 4 FMU submodels were exported from Simulink, the sub models exported were the transmission and chassi. The model where this time built in Dymola because OM does not support simulations with FMUs from Simulink.
The imported FMUs are in the format FMU – CS. This because when exporting FMUs from Simulink the FMU – CS format offers the possibility of defining the workspace parameters from Simulink that should be included in the FMU block. FMU – ME exported from Simulink does not offer that, instead block parameters can be included. Figure 28 shows the full model.

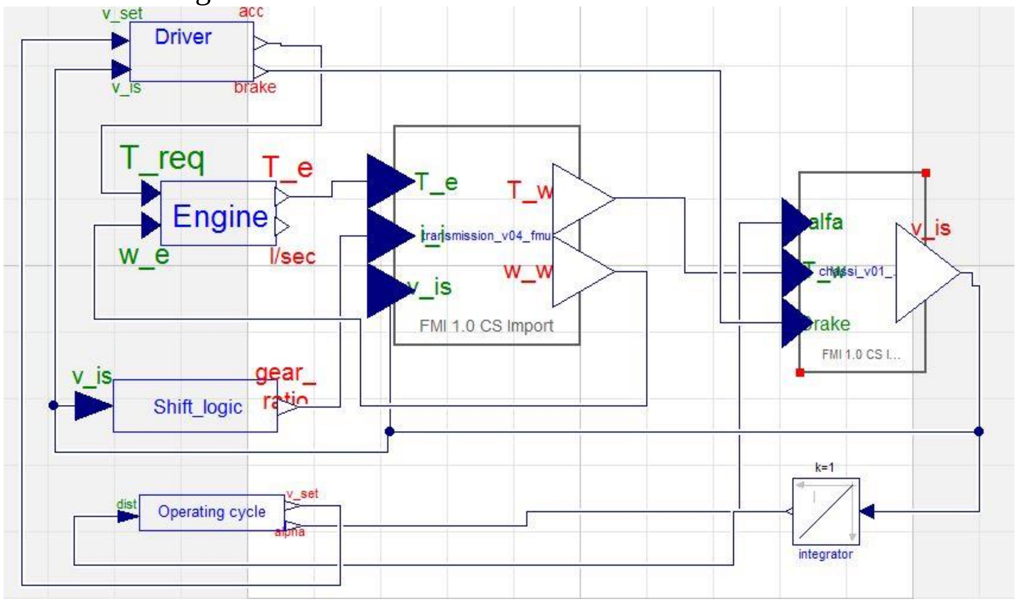
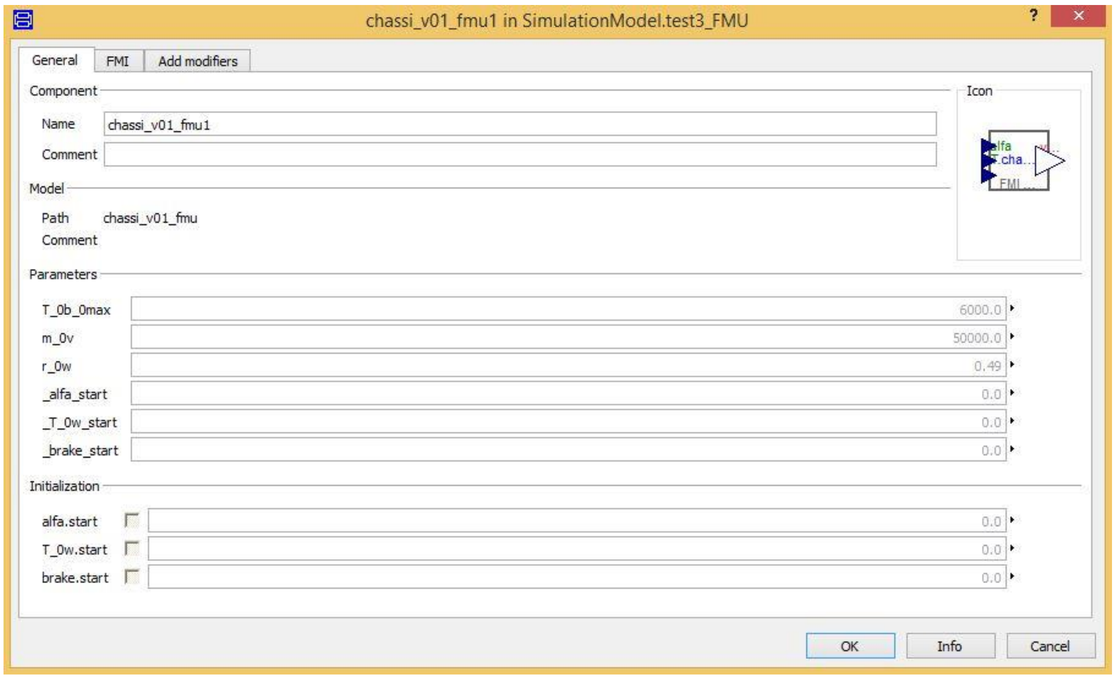


Figure 28: Dymola model with FMU parts

When opening a FMU part in Dymola parameters defined in Simulink can be changed. Figure 29 shows how it looks like.



chassi_v01_fmu1 in SimulationModel.test3_FMU

Component

Name	chassi_v01_fmu1
Comment	

Model

Path	chassi_v01_fmu
Comment	

Parameters

T_0b_0max	6000.0
m_0v	50000.0
r_0w	0.49
_alfa_start	0.0
_T_0w_start	0.0
_brake_start	0.0

Initialization

alfa.start	0.0
T_0w.start	0.0
brake.start	0.0

Figure 29: Detailed FMU view in Dymola

6 Result

As mentined in chapter 5 four models have been designed and simulated:

1. Simulink model built and simulated in Simulink
2. Model modelled in in a Modelica tool simulated in Dymola
3. Simulink with FMUs generated in a Modelica tool
4. Model in Modelica tool with FMUs generated in Simulink and simulated in Dymola

In all simulations that have been done the same operating cycle and vehicle have been used. This has been used because rather of conducting a lot of simulations with different vehicles and cycles the point of interest have been to compare data between the simulations in the two softwares with the different simulation set-ups (explained in chapter 5), how well data matches when using same start parameters, and also simulation time. The result will be presented in this chapter and discussed in chapter 6.5. The data of interest from the simulations made in in the two simulation tools are:

- Desired speed v_set
- Actual speed v_is
- Accelerator pedal position
- Brake pedal position
- Fuel consumption
- Simulation time

6.1 Actual speed v_is

The first plot shows the desired speed v_set and the actual speed v_is from the different simulation models.
As can be seen in Figure 30 it is not stable driving. The truck is heavy and the GAG road is not flat but consists of rather big altitude changes. This gives a heavy truck problem keeping a constant speed uphill and downhill.

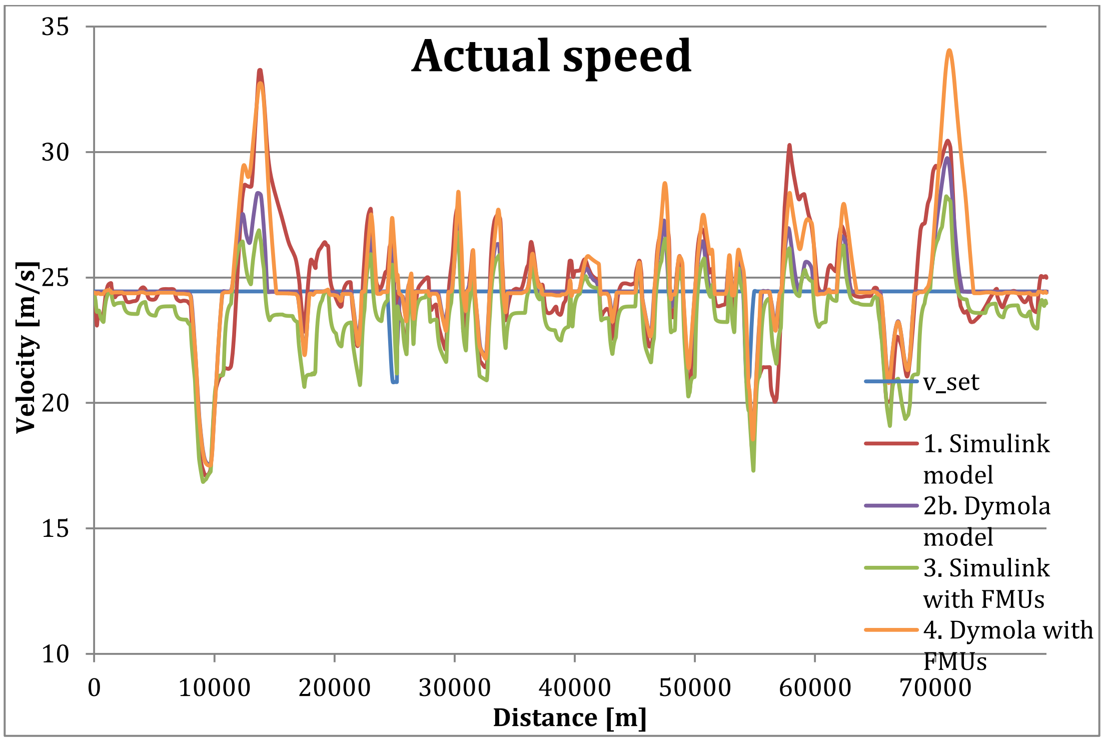


Figure 30: Actual speed v_is for the different simulation models

6.2 Accelerator pedal position

The plot below is the accelerator pedal position during the cycle. The y-axis is in percentage %. Basically the accelerator pedal should be engaged on flat roads and uphill when more throttle is needed. If comparing with Figure 30 this can be seen.

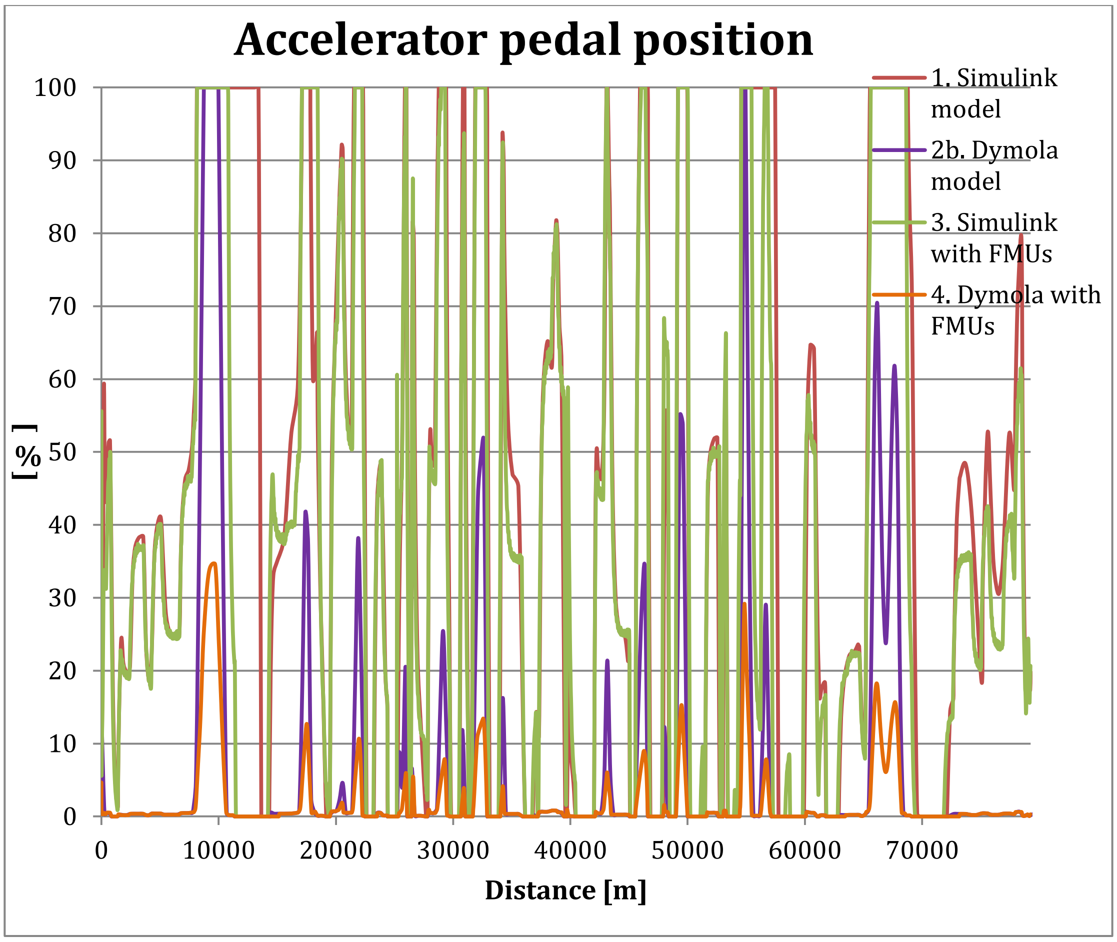


Figure 31: Position of accelerator pedal in the four different simulations

6.3 Brake pedal position

For brake pedal position basically the opposite from the accelerator pedal position is valid. When the vehicle travels downhill the speed increases and the brake pedal is engaged. The level of engagement depends on how much bigger the actual speed v_is is compared to the desired speed v_set.

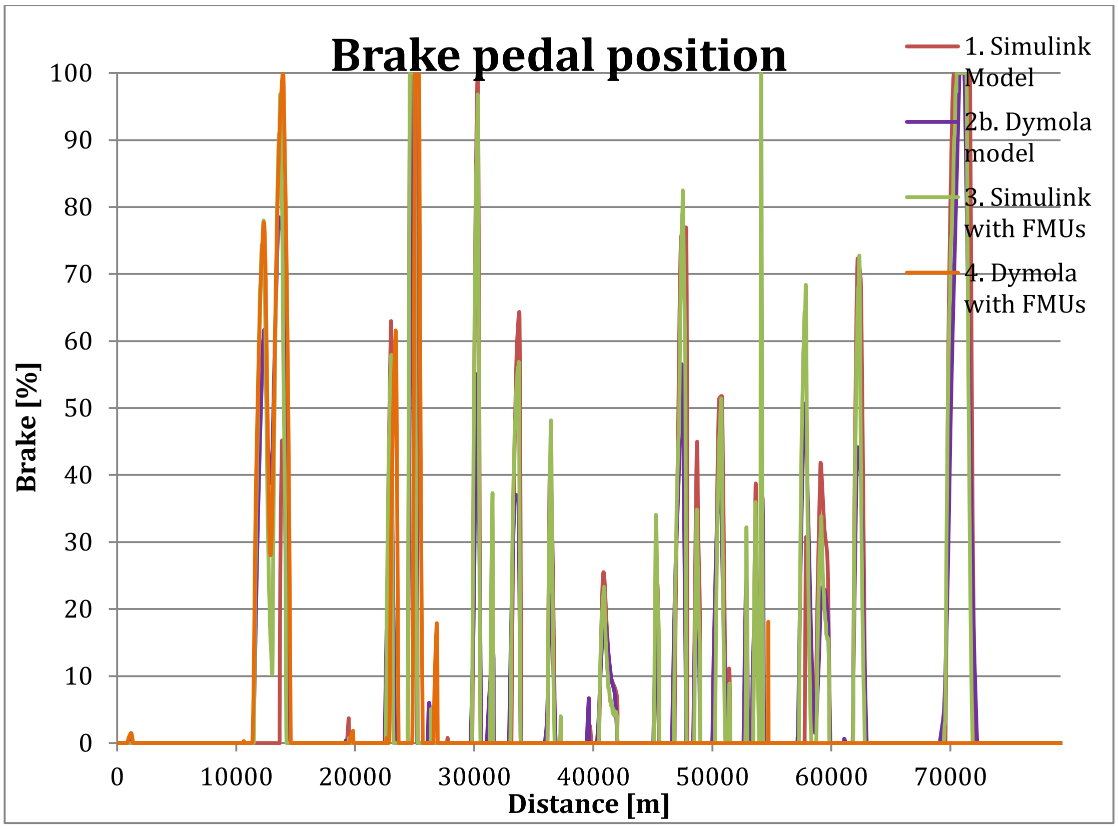


Figure 32: Position of brake pedal in the four different simulations.

6.4 Fuel consumption and simulation time

Table 5 below shows how long the simulation time was for the different simulation models. The fuel consumption is also reported.

Table 5: Result of simulation time and fuel consumption in the different simulation softwares

	Simulink	Dymola	Simulink FMU	Dymola FMU
Simulation time [s]	0.51	0.12	163.1	60
Fuel consumption [l/10km]	3.96	3.90	3.88	4.05

6.5 Discussion of result

One would think that the results from the different simulations would be a close match, because basically the mathematics behind the models are the same. The models are also built in the same way. However the result shows that big variation can be seen during some time steps. Some small diffrences was expected, differences because different solvers in the simulation softwares and settings but not more than under a percentage or smaller.

The most plausible explanation is that this is because of human error. Even though I was been very carefull when declaring vehicle parameters and controlled several times that they are the same in the different softwares and models there could be some that I have missed and been giving wrong value.

One sub model that can be the reason for at least a part of the inconclusive result is the driver model. The driver is built in the same way in both Dymola and Simulink but when the Dymola driver was imported as FMU in Simulink and simulated with the results differed from when the pure Simulink model was used.

The positive side is that all simulations generated results, also the one made with FMU parts. It took a lot of work to get these models to simulate.
Also worth mentioning is that when rewieving the result in Figure 30 the plots from the simulations have a similar behavior, peaks and dips are placed in the same places on the x-axis, which also is very positive.

Regarding the fuel consumption this value lies around 3.95 l/10km. However the engine maps used in the simulations are not the correct one for the chosen engine so the fuel consumption value has to be treated with caution. The fuel consumption was added in the results so it could be compared between the different simulations and it only differs about 0.2 l/10km between the smallest and biggest value.

The difference in simulation time between Simulink and Dymola without FMUs is a factor of 6, Simulink has 0.51 sec and Dymola 0.08 sec and shows that Dymola conducts the simulation faster, this also goes for simulations with FMU. Both simulations largely increases in time when simulating with FMUs. This however can depend of different reasons.
The biggest reason is the number of simulation steps during the simulations. Simulink used more steps in both the simulation without FMUs as well as in the simulation with FMUs, and the more steps the longer it takes. This problem was tried to be solved by changing the error tolerances used in the simulations in Simulink but when this was tried the simulations took even longer time or crashed because it was to big error tolerance. Also different solvers were tested but without and significant change in result.

7 Conclusions and Future Work

7.1 Conclusion

From the simulation software investigation a set of formats and tools are proposed:

- For modelling the proposed formats are: Simulink and Modelica
- For simulation the proposed tools are: Simulink and OpenModelica
- For model exchange the proposed format is: FMI

OpenModelica:
OpenModelica has for now some functionaliy problems that are serious. The identified problems are further described in Future work below.

FMI: FMU sub models slows down the simulations quite much. The built models are not very complicated and should not take 180 sec. This is something that needs to be investigated more.

7.2 Future work

OpenModelica:
OM looked promising when the simulation software investigation was conducted but it turns up that there are some problems and thus the recommendation for the pHd project OCEAN has to be that OM is not a tool that should be used right now.
Two major problems are:

- Combitable: when building Operating Cycles and Engines usually some kind of tables are used with the needed data. In OM the tables that can be used are called CombiTable1D and 2D. However when trying to load the data into the CombiTables it shows up that it is not possible. This makes it imposible to build up engine maps and a operating cycle and no simulations can be made.
- FMI: When OM was chosen as one of the simulation softwares to proceed with one major reason was because its FMI compability. However when trying to load FMUs exported from Simulink it is shown that it is not possible to conduct a simulation with the FMU model. This is also a major drawback with OM.

About the FMI problem between OM and Simulink I found out, unfortunally to late to test, that this could be because of compatibility. I have used and exported FMUs from Simulink in 64 bit, and apparently OM only supports 32 bit for now. This is something that will have to be investigated futher.

Som smaller problems is the graphical interface. As can be seen in eg. Figure 20 on page 26 it is very messy. With more experience maybe this can be made to look more structured.

Even though the recomendation of OM is that for now it should not be used I think that this software looks promising for the future. It’s an open source and quite “young” software, and the people/companies joining the OM consortium grows, making development of the software possible.

When CombiTables and FMU implementation are improved this could be something to investigate again for simulations on vehicles.

Engine:
Implement moment of inertia in engine model to make it more realistic. Also the fuel maps are not from the actual engine. Using the real engine maps is something that can be done.

8 References

1. http://www.energimyndigheten.se/Forskning/Projektdatabas/
2. Simulink. http://se.mathworks.com/products, 2015
3. Dymola. http://www.3ds.com/products-services/catia/capabilities/modelica-systems-simulation-info/dymola/, 2015
4. Maplesoft. http://www.maplesoft.com/products/maplesim, 2015
5. OpenModelica. https://openmodelica.org/, 2015
6. Fritzon, Peter. (2004): *Principles of object-oriented modeling and simulation with Modelica 2.1.* John Wiley & Sons Inc, United State of America, 2004
7. Scilab. http://www.scilab.org/, 2015
8. Scicos. http://www.scicos.org/, 2014
9. Python. https://www.python.org/, 2015
10. Functional mock-up interface. https://www.fmi-standard.org/, 2015
11. *FMI Toolbox User´s Guide 1.8.5*. Modelon, Lund, Sweden, 2014
12. Jacobson, Bengt (2013): *Vehicle Dynamics.* Department of Applied Mechanics, Chalmers University of Technology, Göteborg, Sweden, 2013
13. Jacobson, Bengt (1997): *Modular Simulation Tool for vehicle propulsion concerning energy consumption and emissions.* VehProp project. Department of Machine & Vehicle Design, Chalmers University of Technology, Göteborg, Sweden, 1997
14. Heywood, John B (2011): *Internal combustion engine fundamentals.* McGraw-Hill, New York, United State of America, 1988

9 Appendices

9.1 Max torque curve for engine D13K540

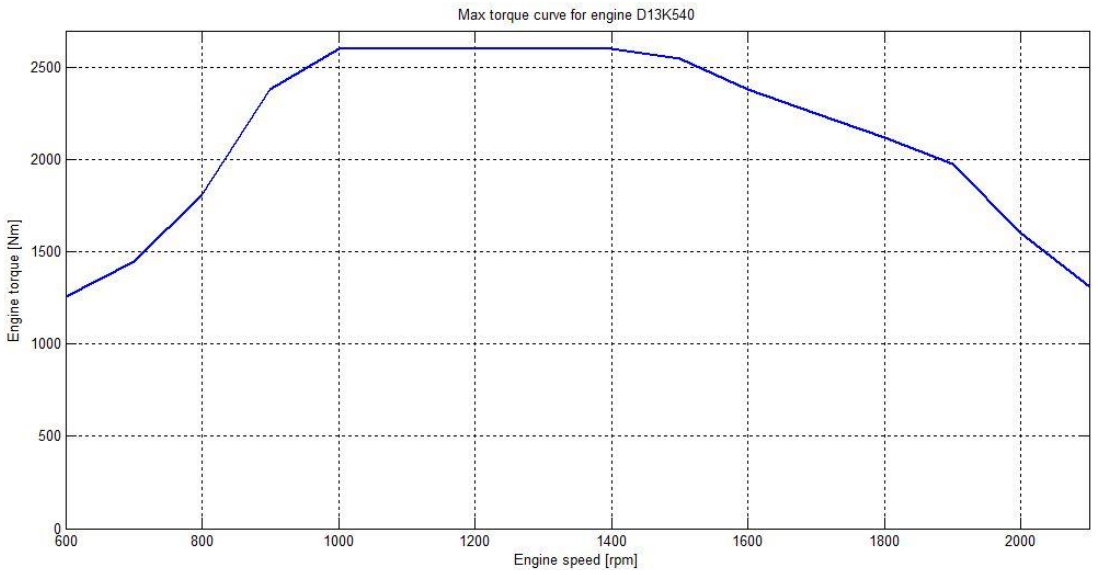


Figure 33: Max torque curve for used engine

9.2 Simulink abbrivations

Operating cycle:
dist – the distance traveled [m]
alfa - Inclination of the road [°]
v_set - speed limit at road [m/s]
Road_Matrix – includes information about road.
Dist, vset, alt – Distance, set speed and altitude in drive cycle

Driver
v_set - same as above
v_is - the actual speed, derived from chassi submodel [m/s]
acc - Acceleratior Pedal position [%]
brake - Brake Pedal position [%]
PID controller- depending of driver behavior wanted PID parameters can be changed
Engine
acc – same as above
w_e – engine speed [rad/sec]
w_e_m – engine speed input from vehicle data {rad/sec]
T_req – required torque to obtain v_set [Nm]
T_req_m – Matrix to obtain required torque from at a certain acc pedal position and w_e
Max_torque_curve – torque curve, depending of vehicle [Nm]
q_fuel – fuel consumption map [mg/stroke]
q – fuel consumption at a certain T_req and w_e [mg/stroke]
T_ss – steady state Torque [Nm]
T_e – engine torque [Nm]
l/sec – as it says, fuel used in l/sec

Tank
l/sec – same as above
v_is – actual speed [m/s]
l/100km – as it says, fuel used in l/100km

Transmission
gear_ratios – vector with gear ratios for vehicle simulated
i_FD – Final drive
etaT – transmission losses
r_w – wheel radius [m]

Chassi
alfa – road inclination [°]
T_w – wheel torque [Nm]
m_v – mass of vehicle [kg]
g – gravity coefficient [m/s^2]
mu - rolling resistance coefficient [-]
Cd- Drag resistance coefficient [-]
A – frontal area [m^2]