Driveline Modelling using MathModelica

Per Nobrant

Reg nr: LiTH-ISY-EX-3114

2001-03-02

Driveline Modelling using MathModelica

Master thesis performed in Vehicular Systems
at Linköpings Institute of Technology by

Per Nobrant

Reg nr: LiTH-ISY-EX-3114

Supervisor:	Lars Nielsen, Vehicular Systems LiTH
	Lars Eriksson, Vehicular Systems LiTH
Examiner:	Lars Nielsen, Vehicular Systems LiTH

Linköping 2 March 2001

Avdelning, Institution
Division, Department

Vehicular Systems
Dept. of Electrical Engineering

Datum:
Date:

2001-03-02

Språk Language	**Rapporttyp** Report category
☐ Svenska/Swedish	☐ Licentiatavhandling
☒ Engelska/English	☒ Examensarbete
	☐ C-uppsats
	☐ D-uppsats
☐ ______	☐ Övrig rapport
	☐ ______

ISBN

ISRN

Serietitel och serienummer Title of series, numbering **ISSN**

LiTH-ISY-EX-3114

URL för elektronisk version
`http://www.fs.isy.liu.se/`

Titel:
Title: Driveline Modelling using MathModelica

Författare: Per Nobrant
Author:

Sammanfattning
Abstract

This thesis is a study of driveline modelling. The modelling is done in MathModelica. Components from the free standard Modelica library have been used when appropriate. The remaining components have either been defined by mathematical equations only or by using mathematical equations together with standard components.

The main work has been put into modelling and verifying the clutch, manual gearbox, final drive, shafts, brake and wheel. To this, models of engine, driver, chassis and air drag have been connected and simulated during different driving cycles. The vehicle model is moving in longitudinal motion.

The results shows the expected behaviours, however since no measurements has been available it is hard to say how near the truth the model is, just that the qualitative behaviour is correct.

Nyckelord
Keywords driveline, modelling, manual gearbox, slip, rolling resistance, Modelica

Abstract

This thesis is a study of driveline modelling. The modelling is done in MathModelica. Components from the free standard Modelica library have been used when appropriate. The remaining components have either been defined by mathematical equations only or by using mathematical equations together with standard components.

The main work has been put into modeling and verifying the clutch, manual gearbox, final drive, shafts, brake and wheel. To this, models of engine, driver, chassis and air drag have been connected and simulated during different driving cycles. The vehicle model is moving in longitudinal motion.

The results shows the expected behaviours, however since no measurements has been available it is hard to say how near the truth the model is, just that the qualitative behaviour is correct.

Notation

For respective indexes applies $\omega_{\square} = \dot{\theta}_{\square}$.

Engine

θ_e	engine angle	[rad]
m_{ac}	mass of air in the cylinders	[kg]
M_e	torque out of engine	[Nm]
p_{amb}	ambient pressure	[Pa]
p_i	intake manifold pressure	[Pa]
u_e	accelerator pedal signal	

Clutch

$\theta_{c,in}$	angle of clutch input	[rad]
$\theta_{c,out}$	angle of clutch output	[rad]
ω_{rel}	relative angular velocity, i.e. $\omega_{c,in} - \omega_{c,out}$	[rad/s]
μ_c	friction coefficient between clutch discs	
$c_{c,s,k}$	ratio between maximum static and kinetic torque	
F_c	normal force pressing the clutch discs together	[N]
M_c	friction torque transferred in clutch	[Nm]
$M_{c,k}$	kinetic friction torque transferred in clutch	[Nm]
$M_{c,s}$	static friction torque transferred in clutch	[Nm]
$M_{c,in}$	torque driving clutch	[Nm]
$M_{c,out}$	torque out of clutch	[Nm]
u_c	clutch signal, 0 for free and 1 for fully engaged clutch	

Gearbox

$\theta_{g,in}$	angle of gearbox input	[rad]
$\theta_{g,out}$	angle of gearbox output	[rad]
$M_{g,in}$	torque driving gearbox	[Nm]
$M_{g,out}$	torque out of gearbox	[Nm]
u_g	gear lever signal	

Propeller shaft

θ_p	angle of propeller shaft	[rad]
$M_{p,in}$	torque driving propeller shaft	[Nm]
$M_{p,out}$	torque out of propeller shaft	[Nm]

Final drive

$\theta_{f,in}$	angle of final drive input	[rad]
$\theta_{f,l}$	angle of left final drive output	[rad]
$\theta_{f,r}$	angle of right final drive output	[rad]
i_f	gear ratio for final drive	
$M_{f,in}$	torque driving final drive	[Nm]
$M_{f,l}$	torque of left final drive output	[Nm]
$M_{f,r}$	torque of right final drive output	[Nm]

Drive shaft

$\theta_{d,in}$	angle of inner end of drive shaft	[rad]
$\theta_{d,out}$	angle of outer end of drive shaft	[rad]
$M_{d,in}$	torque driving drive shaft	[Nm]
$M_{d,out}$	torque out of drive shaft	[Nm]

Brake

θ_b	angle of brake	[rad]
μ_b	friction coefficient between brake blocks and disc	
$c_{b,s,k}$	ratio between maximum static and kinetic torque	
F_b	normal force pressing the brake blocks against the disc	[N]
M_b	torque from braking friction	[Nm]
$M_{b,in}$	torque driving brake	[Nm]
$M_{b,out}$	torque out of brake	[Nm]
u_b	brake signal, 0 for no brake and 1 for full brake	

Wheel

θ_w	angle of wheel	[rad]
μ	friction coefficient between tyre and ground	
F_w	force from chassis braking the wheel	[N]
F_r	rolling resistance	[N]
F_d	driving force from tyre-ground contact	[N]
J_w	wheel inertia	[kg m^2]
m_w	wheel mass	[kg]
M_w	torque driving wheel	[Nm]
N	normal force from ground acting on tyre	[N]
r_w	wheel radius	[m]
s	slip coefficient	

Chassis

F_c	force driving the vehicle	[N]
F_a	air drag force braking the vehicle	[N]
m_v	vehicle mass (excluding the mass of the wheels)	[kg]
x	covered distance for the vehicle	[m]
v	vehicle speed, i.e. $\dot{x}$	[m/s]

Components from the free standard Modelica library

The following components and units from the free standard Modelica library has been used in different models in this thesis. For more information on the implementation of each component see [1].

Modelica.Mechanics.Rotational:

BearingFriction	Inertia
SpringDamper	Brake
GearEfficiency	Torque
IdealGear	Clutch
Flange_a	Flange_b

Modelica.Mechanics.Translational:

Force	SlidingMass
Flange_a	Flange_b

Modelica.SIunits:

AngularVelocity	Velocity
AngularAcceleration	Acceleration
Torque	Force

Modelica.Blocks:

Interfaces.InPort	Continuous.TransferFunction
Interfaces.OutPort	Nonlinear.Limiter

ModelicaAdditions.Tables:

CombiTableTime	CombiTable1D

Contents

1 Introduction

The driveline is a fundamental part of a vehicle. It is the driveline that transforms the energy from the combustion in the engine to kinetic energy of the vehicle. It is of great importance that this is done as efficient as possibly, which will lead to better performance and lower fuel consumption. Besides, the driveability and comfort must still be high. To reach this it is important to be able to simulate the behaviour of the driveline, and thus it is necessary to have a model of it.

1.1 Background

In product development today it is becoming more and more important to design and test products by computer simulations before building prototypes, so called virtual prototyping. The benefits are lower expenses, shorter time-to-market, more iterations in the testing cycle and thus higher quality products. This certainly applies to the product development in the automotive industry.

In this project an automotive driveline has been modelled using the newly developed modelling and simulation environment MathModelica. The work has been carried out on Vehicular Systems at Linköping University in cooperation with MathCore.

1.2 MathCore

MathCore was founded in 1998 in Linköping, Sweden. The purpose of the company was to create a generally available and easy-to-use simulation tool, MathModelica, for complex products and system development based on Modelica. For more detailed discussions about Modelica and MathModelica, see chapter 2.

MathCores competence domains is control, modelling, simulation and animation. The company has today 15 employees. For more information about the company, see [2].

1.3 Simulation Problems Addressed in the Thesis

This thesis is a study of object oriented driveline modelling.

The main goal is to model an automotive driveline connected to a vehicle model in longitudinal motion. The modelling is done in MathModelica. First and foremost existing components from the free Modelica standard library shall be used, and when these are not sufficient new components are developed.

There are some modelling difficulties in a driveline, and how these can be handled in MathModelica is of special interest.

The clutch connects or disconnects the engine with the rest of the driveline. When the engine is disconnected from the driveline there must be at least two states to describe the motions, and when the engine and driveline is connected it is enough with one state. Thus the clutch model must handle a different number of states.

In the manual gearbox the gear ratio is changed in steps. This means a problem because the inertias can not change their rotational speeds instantly. Furthermore the possibility and need for the model to handle neutral gear shall be investigated.

The wheels have a rolling resistance including a resistance when standing still. The wheel model must be implemented so that the correct resistance is applied. The model shall also handle slip.

Since the driveline is elastic, mechanical resonance may occur. These resonance affects functionality and driveability and causes mechanical stress and noise, so elasticity is an important part of the model.

1.4 Limitations

Since no measurements have been available, the work has been concentrated on modelling the qualitative behaviours. Most parameter values in the implementations are approximate and aims to get reasonable results, and the possibility to discover that something is incorrect.

The engine is such a large and complex system that an extensive model of it would demand as much work as the rest of the driveline. Therefore, in this thesis it will be enough to model the general behaviour of the engine, such as the maximum torque, engine braking and that the torque is decreasing with increasing engine speed. This will be sufficient to simulate the rest of the driveline under realistic conditions for a drive cycle.

The vehicle is assumed to only be driving forward, i.e. $v \geq 0$.

2

Simulation Environment

This section describes the multi-domain modelling language Modelica and the MathModelica environment. Modelica is a standardized language for complex multi-domain dynamic modelling while MathModelica is a modelling and simulation environment building on the Modelica language.

2.1 Modelica

Modelica is a language for modelling of physical systems, designed to support effective library development and model exchange. It is a modern language built on acausal modelling with mathematical equations and object-oriented constructs to facilitate reuse of modelling knowledge [3].

The work with Modelica started in September 1996 with a group of about fifteen persons with knowledge about modelling languages and models with differential algebraic equations (DAE). The first language definition, version 1.1, came in December 1998. In this work version 1.3 has been used, but since the middle of December 2000 version 1.4 is available. Furthermore the language contains a standard library with finished components that can be downloaded from the internet [1]. The intention is that the user shall use these and self developed components and build a complete model by dragging components and drawing lines between them like in MATLABs Simulink [4].

The big difference, compared to Simulink, is that the components are connected just like in the real system. An example illustrating this among other things will be seen in section 2.2.1.

Acausal Modelling

One advantage of Modelica in comparison to other modelling languages is that it supports acausal modelling. Modelica is based on equations, and there is of no significance in which way the equation is read. In many other modelling languages it is common that the variable is assigned a value, thus making the dataflow restricted to have just one direction.

An equation based modelling language increases the reusability of the components significantly as the direction of the dataflow has two options. This makes the components more general and can thus be used in different applications [5].

Multi Domain Modelling

It can be difficult to make a good and easy-to-grasp model at transitions between different physical domains, such as electronics, mechanics, hydraulics or thermodynamics. Previous languages have been good in one domain, whereas Modelica handles transition between domains. In Modelica the components from different domains are connected just like in the real system. This makes the model easier-to-grasp and easier to develop and maintain [5].

For more information about the Modelica language see [1].

2.2 MathModelica

MathModelica is a modelling tool developed by MathCore (see section 1.2), building on the Modelica standard.

MathModelica is based on Mathematica [6], a powerful technical computing system for doing both numeric and symbolic computations using standard mathematical notation. This makes it possible to include the equations in a simulation, notated in a natural fashion. The intention is that most work will be done in the graphical model editor, using already finished components. The neat connection to notebooks offers an excellent documentation system, and the possibility to manipulate the models in MathModelica-code.

The simulation engine of Dymola [7] from Dynasim is used for compiling and simulation. The source code is written either in plain Modelica or MathModelica syntax, where MathModelica-syntax is similar to both Modelica and Mathematica-syntax. At simulation MathModelica translates the source code to Modelica-code. This code is sent to Dymola that generates C-code, compiles and simulates. The result is sent to MathModelica and can be presented as plots or animations and used for analysis.

2.2.1 Example

This electrical example intends to show how modelling and simulation in MathModelica works, and some characteristics of the language. The example is chosen so that it will be easy to understand the modelling technique, but still complex enough to demonstrate the characteristics and power of the language. For a more detailed description of the Modelica syntax, see [1].

Figure 2.1 shows an electrical circuit modelled in Simulink. The circuit contains a sine voltage source *V*, a resistance *R* and an inductor *L* in series. The blocks is arranged after the computations in due order. Thus the block reveals how and in which turn the computations shall be made. This part of the modelling must be done manually before implementing in Simulink. In this example the equations describing the circuit is

$$v - u_R - u_L = 0$$
$$u_R = Ri$$
$$u_L = L\frac{di}{dt}$$

Well-known relations such as Kirchoffs and Ohms laws give the equations. For bigger systems one often uses bond graphs to get all equations.

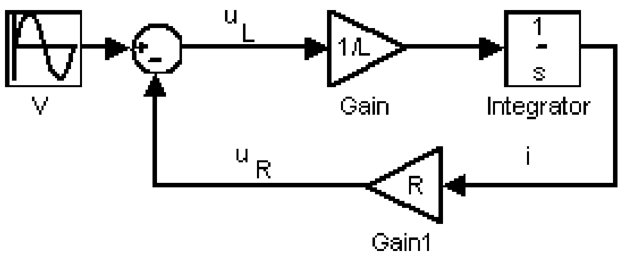


Figure 2.1. A simple electrical circuit modelled in Simulink.

In the figure the current i and the voltages u_L over the inductor and u_R over the resistor has been marked.

Figure 2.2 shows the same electrical circuit modelled in MathModelicas graphical model editor. The components are from the standard Modelica electrical library. There are no predetermined directions for the signals to be calculated but the model has the same topology as in reality. This component-based structure tells *what* to be simulated. The program decides how this is done.

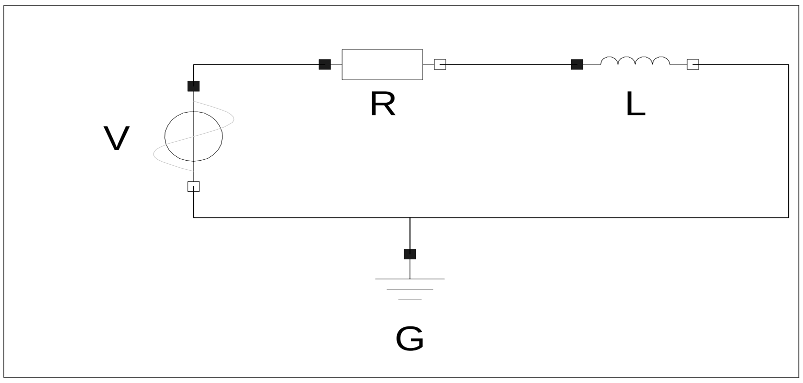
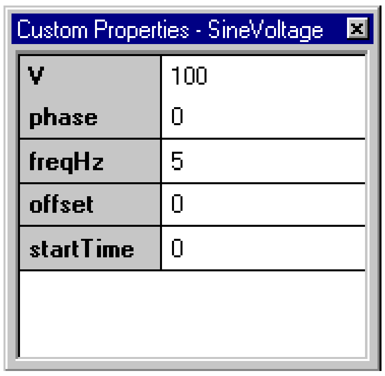


Figure 2.2. A simple electrical circuit modelled in MathModelica.

The parameters can easily be changed through the parameter list. Figure 2.3 shows the parameter list for the sine voltage source.


Custom Properties - SineVoltage

V	100
phase	0
freqHz	5
offset	0
startTime	0

Figure 2.3. Parameters for Sine Voltage source.

If one wants to modify the model, changes can be made directly in the source code. The code generated for the electrical circuit is shown below.

```
Model[Circuit,
 Resistor R[{R == 100}];
 Inductor L[{L == 0.01}];
 SineVoltage
  V[{offset == 0, startTime == 0, V == 100, phase == 0,
    freqHz == 5}];
 Ground G;
 Equation[
  Connect[R.n, L.p];
  Connect[R.p, V.p];
  Connect[L.n, V.n];
  Connect[V.n, G.p];
 ]
]
```

The model is named Circuit, and consists of the following parts:

- Resistor R
- Inductor L
- Sine Voltage source V
- Ground G

The parameter values are set within the square brackets. In the Equation-part the connections between the components are stated. Each end in a component that can be connected to another component must have a name. In this electrical example they have been denoted positive, *p*, and negative, *n*. Note that *p* and *n* just are name conventions and that it goes excellent to flip a component and connect *R.p* to *L.p*.

Inductor

To go further down in the structure we will take a close look at how the inductor is modelled. The other components are implemented in a similar way so they will be leaved out. The code describing the inductor can be seen below:

```
Model[Inductor, "Ideal electrical inductor",
 Extends[TwoPin];
 Parameter Real L[{Unit == "H"}]; "Inductance";
 Equation[
  L i' == v;
 ]
]
```

Extends denotes inheritance. The declaration of *L* as a parameter places it in the parameter list for the inductor. A parameter is constant during a simulation but can be changed between the simulations. In the Equation-part the equations for an ideal inductor is stated, $u_L = L\,\frac{di}{dt}$. Note that `i'` denotes $\frac{di}{dt}$, and that space can be used at multiplication instead of the multiplication sign (*). It does not matter on which side of the equal sign the variables are.

To know where the variables *i* and *v* comes from we have to look at the model TwoPin, below.

```
Model[TwoPin,
 "Superclass of elements with two electrical pins",
 Pin{p, n};
 Voltage v;
 Current i;
 Equation[
  v == p.v - n.v;
  0 == p.i + n.i;
  i = p.i
 ]
]
```

TwoPin can be seen as a class defining an object with two electrical connections. The objects that inherit this class will thus have these characteristics. Furthermore the relations between these connections are defined in the equation part. Kirchoffs laws give that the voltage over the component is the electrical potential at the positive end minus the potential at the negative end, and that the current in to the component is equal to the current out of it.

Also, in TwoPin the declarations of *i* and *v* are made. The types Voltage and Current is defined as:

```
Type[Voltage, Real[{Unit == "V"}]]
```

```
Type[Current, Real[{Unit == "A"}]]
```

Finally, the object Pin is seen as an electrical connection on the component and is defined as

```
Connector[Pin,
 Voltage v;
 Flow Current i;
]
```

The "Connector"-statement creates a physical connection in the graphical interface and defines how the signals shall be treated. The declaration of *v* means that between two connections of the type pin, the value of *v* shall be the same, as follows from Kirchoffs voltage law. By typing *Flow* in front of *i*, the current will be treated as a flow and thus sum up to zero in a node, as follows from Kirchoffs current law.

Simulation

Below the model is simulated during one second with the parameter values stated in Circuit.

```
res = Simulate[Circuit, {t, 0, 1}];
```

When the simulation is finished the result can be plotted. As an example the current and voltage for the inductor are plotted for 0.5 seconds in figure 2.4.

```
PlotSimulation[{L.i[t], L.v[t][t]}, {t, 0, 0.5}];
```

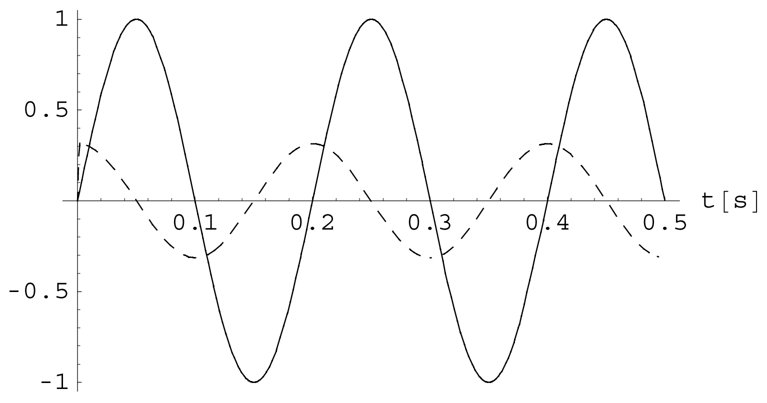


Figure 2.4. Current (solid) and voltage (dashed) for the inductor L, in the electrical circuit.

2.2.2 Mechanical Modelling

There are two types of mechanical components, rotational and translational. For the rotational components we have the torque as intensity and angular velocity as flow, and for the translational force respective velocity. The equations describing the components are thus describing the relations between intensity and flow. To describe the connections between components connectors is used.

Connectors

The connectors for mechanical components are called flanges. The ingoing connection is usually called flange_a and the outgoing flange_b. Between two connections of the type flange, the value of the angle (distance) shall be the same. The torque (force) is defined as a flow and thus sums up to zero in a node, i.e. the outgoing torque from component one is equal to the ingoing torque of component two. For more information about flanges see [1].

3 Physical Modelling

In this section a short review of how to make mathematical models for dynamic systems will be given. To simulate the system we want a model written on the form

$$\begin{aligned} \frac{d}{dt} x(t) &= f(x(t), u(t)) \\ y(t) &= h(x(t), u(t)) \end{aligned} \tag{3.1}$$

The modelling work can be divided in three phases

1. Structuring the Problem

The first phase is to divide the system into subsystems and decide the main relations, which variables that are important and how they influence each other. It is in this phase where one decides the level of complexity and the degree of approximation for the model, thus it is important to be aware of the purpose of the model. In this phase it is of great importance that the model builder has good knowledge and intuition of the system to be modelled. This part of the modelling ends in some sort of block diagram describing the subsystems and their interaction.

2. Setting Up the Base-equations

This phase aims to describe the subsystems, blocks, from phase 1. The relations between the variables and constants in the subsystems are stated. For physical systems this is done by using the laws of nature and physical base equations that are supposed to be valid, such as Kirchoffs and Ohms laws for electricity and Newtons laws for mechanics. This often means that approximations and idealizing are introduced (like "point mass" or "ideal gas").

3. Compiling the State Space Model

After the two first phases the model is actually complete. However, the equations need to be organized in the form of equation 3.1, so they can be simulated. This work can be done manually, or better, in a computer program for modelling. By using MathModelica the equations from phase 2 and the connections between the subsystems is entered and the compilations is done by MathModelica before starting the simulation. Thereby the third phase is automatized so that the model builder can avoid the error prone work and concentrate on the first two phases.

For more information about physical modelling there is several books, for example [8] or [9].

4

The Driveline Parts

The driveline is an integral part of the vehicle. The driveline is the system that transfers the signals from the driver (gear lever, accelerator and brake pedal) via the combustion in the engine to vehicle speed. This section will give a quick survey of the different parts and subsystems of the driveline. Figure 4.1 shows how a driveline for a rear-driven vehicle is assembled.

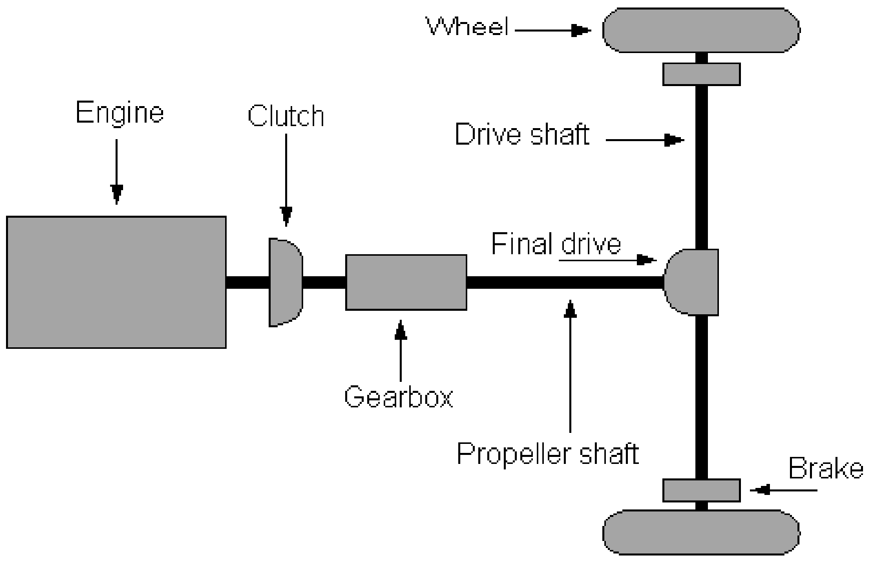


Figure 4.1. Driveline for a rear-driven vehicle.

Engine

The engine is the power source in the drivetrain. The driver controls the engine with the gas pedal. The output from the engine is the torque resulting from the combustion, and the resulting engine speed. This thesis will not enter deeply into the modelling of the engine and its subsystems but a model of the engine torque will be treated. This model will be good enough to use when simulating the rest of the driveline in a drive cycle. The modelling of the engine will be treated more detailed in section 8.3.

Clutch

The function of the clutch is to connect and disconnect the engine to the rest of the driveline in vehicles equipped with a manual gearbox. A friction clutch consists of a clutch disc connecting the flywheel of the engine and the input shaft of the gearbox. The clutch will be discussed thoroughly in section 5.1.

Manual Gearbox

The function of the gearbox is to change the gear ratio from the engine to the wheels and thereby extending the work range. The gearbox and its modelling will be further discussed in section 5.2.

Propeller Shaft

The propeller shaft connects the gearbox with the final drive. This part does not exist on front wheel driven vehicles where the final drive is integrated with the gearbox. The propeller shaft will be treated further in section 6.1.

Final Drive

The final drive is a differential gear that allows the driven wheels to have different speeds. This is necessary to be able to drive the car in curves. How the final drive is modelled will be discussed in section 5.3.

Drive Shafts

The drive shafts connect the final drive with the wheels. In section 6.2 the modelling of the drive shaft will be discussed.

Brakes

To be able to stop the car, it is equipped with brakes. The brakes can be of two types, disc brakes or drum brakes. Both types are friction brakes where brake blocks (shoes) are pressed against the disc (drum). The brake model will be discussed in section 6.3.

Wheel

The wheels are the part of the driveline that has contact with the ground. The tyre-ground contact transforms the rotational motion to translational motion. The tyre is very complex to model. For instance, driven wheels do not roll but instead they rotate faster than the corresponding translational velocity and a rolling wheel is deformed which causes rolling resistance. The modelling of the wheel is treated in chapter 7.

Chassis

There are many definitions of what a car chassis is, but these all have in common that the vehicle body forms a part of the chassis. Sometimes the suspensions and wheels are included. In this thesis the chassis means all the parts of the car not listed above. The model of the chassis is treated in section 8.1.

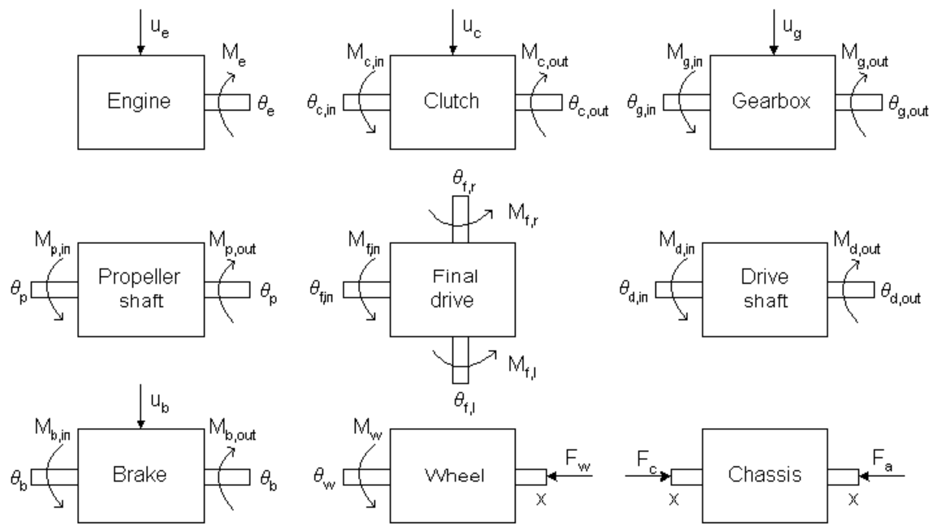


Figure 4.2. Subsystems of a driveline with their respective angle (distance) and torque (force).

Figure 4.2 shows a block diagram over the different subsystems. The torques (forces) are defined as positive in the directions indicated by the arrows. The angles (distances) are defined as positive in the direction indicated by the arrow at the ingoing connection.

In other words, at the outgoing shafts the angle (distance) is defined positive in the opposite direction as a positive braking torque (force).

5

Modelling Clutch and Gears

In this chapter the modelling of the clutch, manual gearbox and final drive is discussed. The function of each part is described in order to explain the behaviour and justify the thoughts behind the modelling. The models are verified by different test setups, and weaknesses and proposals for improvements are noted.

5.1 Clutch

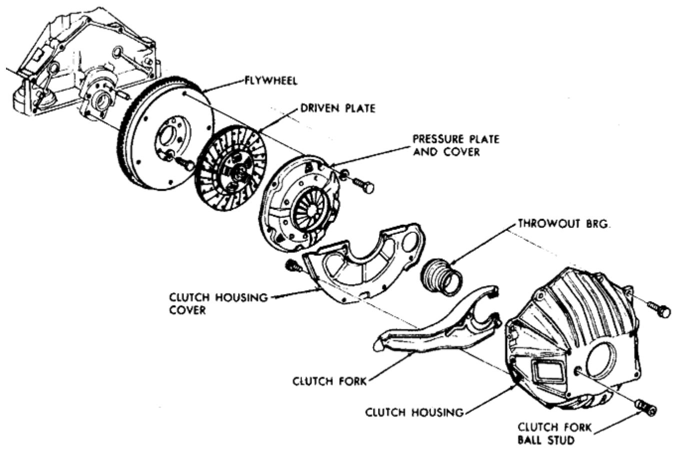


Figure 5.1. Exploded view of typical clutch.

Figure 5.1 shows an exploded view of a typical car clutch. The important parts for modelling the clutch are the flywheel which is connected to the engine crankshaft, the clutch plate and the pressure plate. The pressure plate is connected to the clutch pedal via the clutch fork and presses the clutch plate against the flywheel and thus the clutch locks by the friction between the plates.

The friction clutch connects two masses to each other, which introduces difficulties when simulating clutch connect and clutch release. These difficulties can be easily described by considering a simple system with two masses connected to each other with a clutch. When the clutch is unlocked it has two degrees of freedom, which is represented by two states in the simulation. When the clutch is locked the two masses are acting like one mass with only one degree of freedom. Thus at the time when the clutch locks one state must disappear in the simulation, and likewise one state must appear when the clutch unlocks.

In the free Modelica standard library a clutch is already implemented. The following sections will discuss the behaviours that distinguish a friction clutch and verify that the standard Modelica library clutch fulfil these.

5.1.1 Modelica Standard Clutch

The clutch is modelled to have two flanges where friction is present between the two flanges. These two flanges are pressed together via a normal force, F_c, which influences the maximum torque that the clutch can transfer.

The normal force has to be provided as input signal u_c in a normalized form, $F_c = F_{c,max}\, u_c$, where $F_{c,max}$ (the maximum possible normal force F_c) has to be provided as parameter. Friction in the clutch is modelled in the following way:

When the relative angular velocity, ω_{rel}, is not zero, the friction torque is a function of the velocity dependent friction coefficient $\mu_c(\omega_{\mathrm{rel}})$, and of the normal force F_c

$$M_{c,k}\,(u_c,\ \omega_{rel}) = M_{c,k,max}\,(\omega_{rel})\ u_c \tag{5.1}$$

where the maximum kinetic torque, $M_{c,k,max}(\omega_{rel})$ transferred in the clutch is

$$M_{c,k,max}\,(\omega_{rel}) = n\ \frac{r_o + r_i}{2}\ \mu_c\,(\omega_{rel})\ F_{c,max} \tag{5.2}$$

where r_i is the inner radius and r_o is the outer radius of the clutch discs, and n is the number of friction interfaces. For an automobile clutch n is two, where the first friction interface is between the flywheel and the clutch plate, and the second is between the clutch plate and the pressure plate.

When the relative angular velocity becomes zero, the elements connected by the friction element become stuck, i.e., the relative angle remains constant. In this phase the friction torque is calculated from a torque balance due to the requirement, that the relative acceleration shall be zero. The elements begin to slide when the friction torque exceeds a threshold value, called the maximum static friction torque, computed via:

$$M_{c,s} = c_{s,k}\, M_{c,k,max}\, (\omega_{rel} = 0)\, u_c \tag{5.3}$$

This procedure is implemented in a "clean" way by state events and leads to continuous/discrete systems of equations if friction elements are dynamically coupled, see [1].

The full implementation can be seen in [1].

Two Mass System

The two clutch disks are connected to two inertias, J_e corresponds to the engine side and J_v to the vehicle side. A torque M_e is driving the system and a torque M_v is braking it.

The two masses rotate with speeds independent of each other but their respective motions are connected to each other through the torque transferred by the clutch. Under these conditions the system surrounding the clutch has at least two degrees of freedom.

The equations that describes this are

$$J_e\, \dot{\omega}_{c,in} = M_e - M_c$$
$$J_v\, \dot{\omega}_{c,out} = M_c - M_v$$

The torque transferred through the clutch is

$$M_c\, (\omega_{rel}) = M_{c,k}\, (u_c,\, \omega_{rel})\, \mathrm{sgn}\, (\omega_{rel})$$

where sgn is the signum function. The equations above are valid when $\omega_{rel} \neq 0$.

One Mass System

When the clutch has locked the two rotating masses are locked to each other and thus rotating with identical speed. Under these conditions the system surrounding the clutch only have one degree of freedom.

$$\begin{gathered}(J_e + J_v)\, \dot{\omega}_{c,in} = M_e - M_v \\ \omega_{c,in} = \omega_{c,out}\end{gathered} \tag{5.4}$$

When the clutch is locked the torque transferred through the clutch depends on the applied torques and the inertias. It can be calculated as the driving torque minus the inertia effect of the first mass

$$M_{c,locked} = M_e - J_e\, \dot{\omega}_{c,in} \tag{5.5}$$

Combining equation 5.4 and 5.5 gives the clutch torque

$$M_{c,locked} = \frac{M_e\, J_v + M_v\, J_e}{J_e + J_v} \tag{5.6}$$

5.1.2 Verification of the Modelica Standard Clutch

The simulation is chosen such that the model can easily be verified for correctness [10]. The setup of the test model can be seen in figure 5.2.

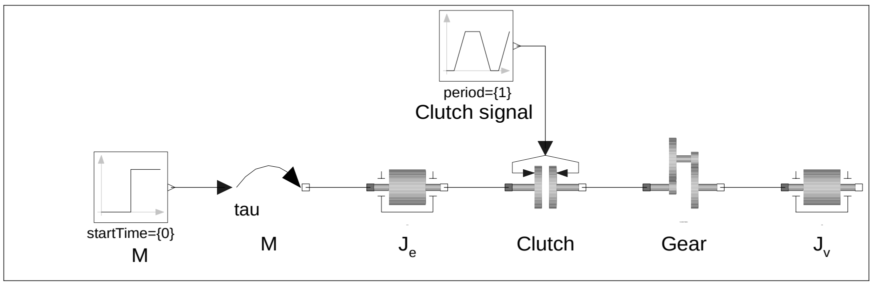


Figure 5.2. Setup for test model.

Clutch Lock

Prior to the lock event the two systems rotate freely from each other and two conditions must be fulfilled for the clutch to lock:
1. The angular velocities must match, $\omega_{c,in} = \omega_{c,out}$, i.e. $\omega_{\text{rel}} = 0$.
2. The torque that the clutch transfers under locked conditions (equation 5.6), must be less or equal to the maximum possible static friction (equation 5.3), i.e. $M_{c,\text{locked}} \le M_{c,s}$.

Two rotating masses with inertia $J_e = 1$ and $J_v = 2$ are connected to the clutch. The gear ratio is $i_g = 2$. Initially J_e rotates with 1 rad/s while J_e stands still, and the clutch signal is zero. Between t = 1 s and t = 2 s the clutch signal, u_c, is ramped from 0 to 1 which results in $\omega_{c,in}$ slowing down and $\omega_{c,out}$ increasing. At t = 1.25 s $\omega_{c,in}$ equals $\omega_{c,out}\, i_g$ and the clutch locks. At t = 2 s a torque is applied on the engine side and the system starts to accelerate. Between t = 3 s and t = 4 s the clutch is ramped down from 1 to 0 and just before t = 4 s the clutch breaks up and J_v stops accelerating. The simulation results can be seen in figures 5.3-5.5.

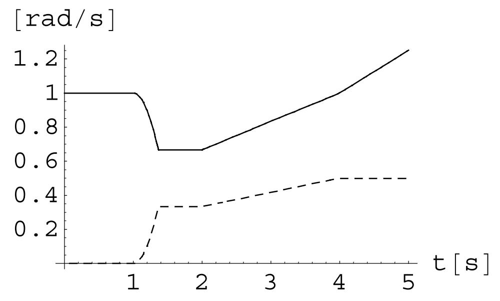


Figure 5.3. $\omega_{c,in}$ *(solid) and* $\omega_{c,out}$ *(dashed).*

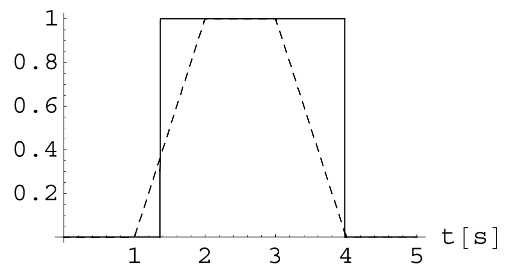


Figure 5.4. Clutch lock (solid) and clutch signal (dashed).

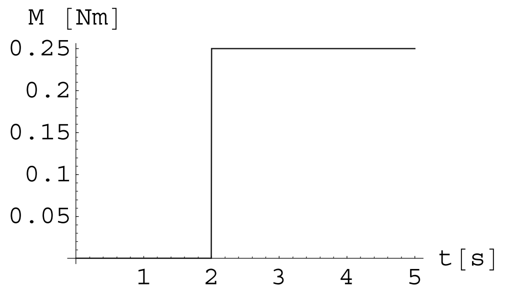


Figure 5.5. Driving torque.

Energy Conservation

One important verification of the clutch is to study the energy conservation, figure 5.6. In the figure the cumulative energy in the system simulated in the clutch lock section above is plotted. The dashed line shows the rotational energy in J_e, $E_e = \frac{J_e\,\omega_{c,in}^2}{2}$, at t = 1 s when the clutch is engaged the energy is transferred to the second inertia, and the dotted line shows the sum $E_e + E_v = \frac{J_e\,\omega_{c,in}^2}{2} + \frac{J_v\,\omega_{c,out}^2}{2}$. The rotational energy in the system decreases when the clutch is engaged which is due to the energy dissipated in the friction clutch while the clutch locks. The work produced on the clutch discs can be calculated as

$$W_c = \int M_{c,k}(t)\,\omega_{rel}(t)\,dt$$

The sum $E_e + E_v + W_c$ is constant 0.5 J when no additional energy is put into the system. At t = 2 s the total energy starts to increase due to the added input torque. The discussion above shows that the Modelica standard library clutch captures the energy conservation.

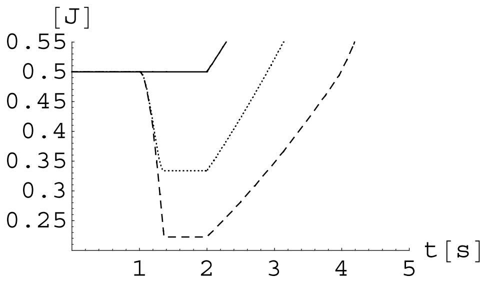


Figure 5.6. Cumulative energy plots of E_e (dashed), $E_e + E_v$ (dotted) and $E_e + E_v + W_c$ (solid).

Clutch Break Up

The condition for determining clutch break up is when the torque that the clutch transfers under locked conditions exceeds maximum for the static friction torque in the clutch

$$M_{c,locked} > M_{c,s}$$

When the clutch breaks up the new state must be initialised so that $\omega_{c,in} = \omega_{c,out}$.

The following simulation intends to show the maximum torque that the clutch can transfer, and that the maximum static friction is bigger than the maximum kinetic friction [11]. The setup for testing clutch break up can be seen in figure 5.7.

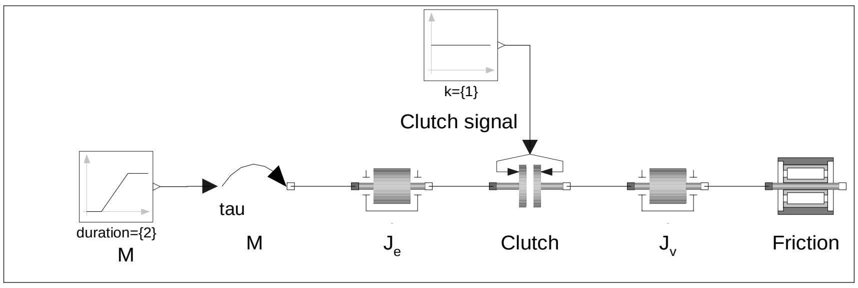


Figure 5.7. Test set up for clutch break up.

The clutch has been designed to transfer a torque of up 405 Nm (which is 1.5 times the maximum torque of the engine introduced in section 8.3). The maximum static torque is set to be 1.1 times the maximum kinetic torque, i.e. the parameter $c_{s,k}$ is 1.1.

The torque is ramped up to 450 Nm and the clutch signal is constant one. Up till 405 Nm the torque transferred in the clutch is equal to the driving torque. When the driving torque exceeds 405 Nm the clutch starts to skid (unlocks, $t = 36\,s$) and the maximum transferred torque is now 368 Nm, see figures 5.8 and 5.9. This verifies that the maximum kinetic torque $M_{c,k,max}$ is 1.1 times bigger than $M_{c,s,max}$.

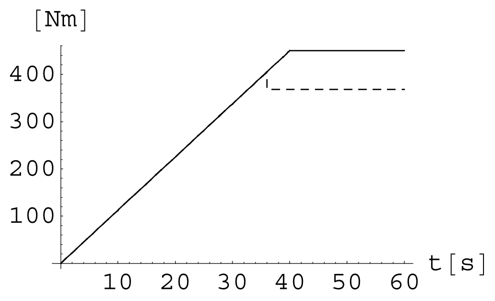


Figure 5.8. Applied (solid) and transferred (dashed) torque.

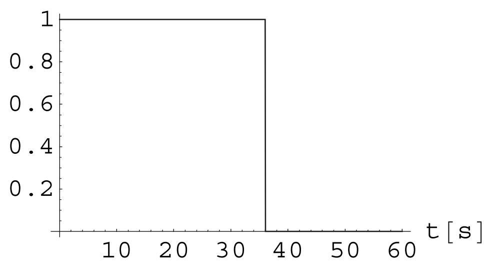


Figure 5.9. Clutch locked.

This simulation with clutch lock, clutch release and energy plots shows that the standard Modelica clutch is implemented correctly. When the clutch is locked the relative angular velocity is less than the smallest number ϵ, such that ϵ and $-\epsilon$ are representable on the machine, which demonstrates that the clutch handles a different number of states.

5.2 Manual Gearbox

In this section we will take a closer look on the design and function of the manual gearbox and the modelling of it. The functionality concentrates on how the changing of gear and synchronization is done. The gearbox modelled will have five gears forward since that is the most common in todays cars.

5.2.1 Design of a Manual Gearbox

A simple illustration of the design of a manual gearbox can be seen in figure 5.10. The main parts of the gearbox are

■ Ingoing shaft from the engine via the clutch. When the clutch is engaged the cogwheel, directly connected to the shaft, turns at the same speed as the engine.When the clutch is disengaged the cogwheel and its shaft is disconnected from the engine.

■ Layshaft with cogwheels. These are connected as a single piece, so all of the gears on the layshaft and the layshaft itself rotate as one unit. The layshaft is connected directly to the ingoing shaft through their meshed gears. Thus if the ingoing shaft is rotating so is the layshaft.

■ Outgoing splined shaft to the differential via the propeller shaft. If the wheels are rotating this shaft will rotate.

■ Cogwheels riding on bearings, so they spin on the outgoing shaft. If the engine is standing still and the wheels are rotating, the outgoing shaft will rotate whereas the ingoing shaft, the layshaft and the cogwheels is motionless.

■ Collars connected to the gear lever fork and through the splines directly to the outgoing shaft. However, the collars can one at a time slide left or right along the outgoing shaft to engage either of the cogwheels. Teeth on the collars, called dog teeth, fit into holes on the sides of the cogwheels to engage the cogwheels to the outgoing shaft.

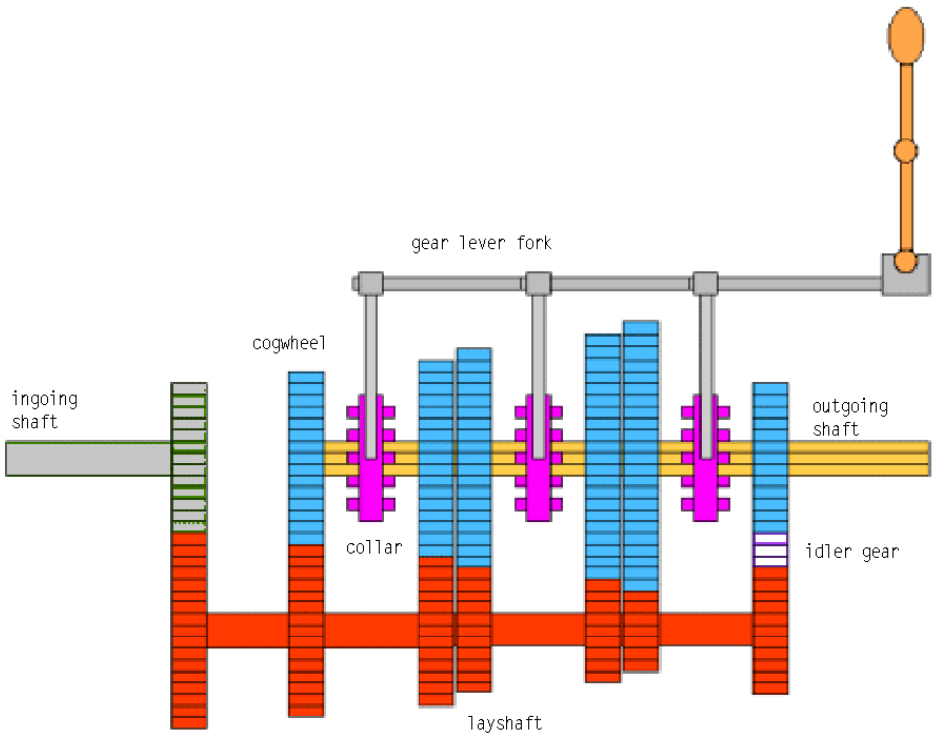


Figure 5.10. Illustration of the principle design of a five speed manual gearbox with reverse.

5.2.2 Synchronization

To make the gear shifting smooth and with minimum noise the speed of the cogwheel that shall be engaged and the speed of the outgoing shaft must be synchronized. The purpose of the synchromesh is to make frictional contact between the collar and the gear and thus synchronize their speeds before the dog teeth make contact with the gear. Figure 5.11 shows the course of events when engaging a gear.

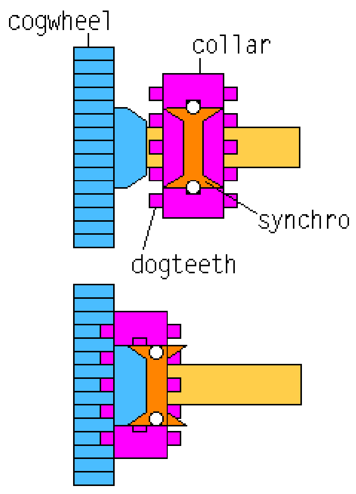


Figure 5.11. Synchronization in the manual gearbox.

The cone on the cogwheel fits into the cone-shaped area, the synchro, in the collar, and friction between the cone and the synchro synchronizes the collar and the gear. The outer part of the collar is attached with springs to the synchro. The outer part of the collar then slides so that the dog teeth can fit into the holes on the cogwheel and engage the gear.

5.2.3 The Gearbox Model

The manual gearbox model is built up as the structure of figure 5.12 shows. The purpose of the clutches is to model the interaction between the cogwheel and collar with synchro. These clutches are controlled so that only one clutch at a time can transfer the torque. The gearbox also has gear efficiency due to friction between the teeth and bearing friction. Friction in the bearings leads to a velocity dependent, additive loss-torque, whereas friction between the teeth of the gear leads to a torque-dependent reduction of the driving torque. The gearbox manufacturers measure both effects together and determine the gear efficiency from it, although for simulation purposes the two effects need to be separated [1].

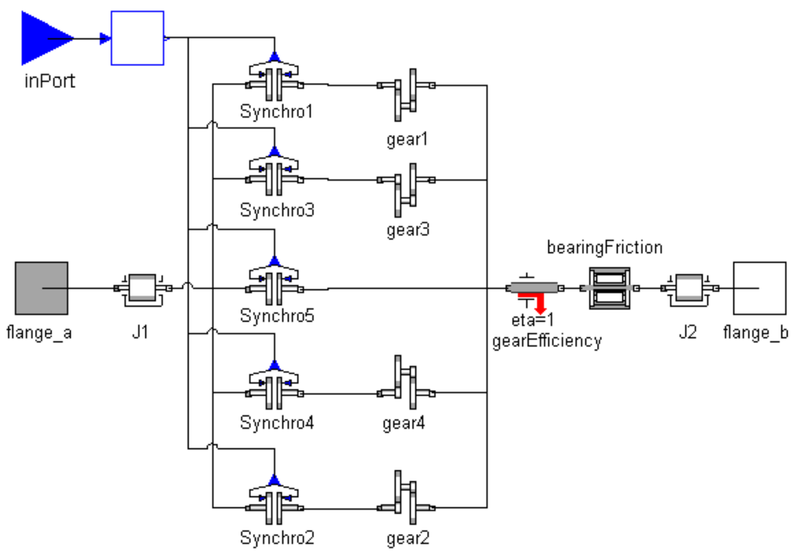


Figure 5.12. Structure of the gearbox model.

This model of the gearbox is quite extensive and models not only the changing gear ratio, for example like in an ideal CVT (continuous variable transmission), but also tries to model the effect of the synchronization. To keep the complexity down for faster simulation, some simplifications have been done. There is one gear efficiency and bearing friction for the whole gearbox and not different efficiencies and frictions for every gear. No consideration has been taken for elasticity, damping or backlash.

The implementation of the gearbox model in MathModelica code can be seen in Appendix A.2.

5.2.4 Evaluation of the Gearbox Model

For testing of the model a simple test setup is used, where the ingoing shaft to the gearbox rotates with constant speed, 15 rad/s and the gear lever signal is increased from zero to five so that all gears will be engaged one after another. In figure 5.13 the rotational speed of the ingoing and outgoing shafts can be seen for the different gears. The gear lever signal is ramped when going from one gear to another.

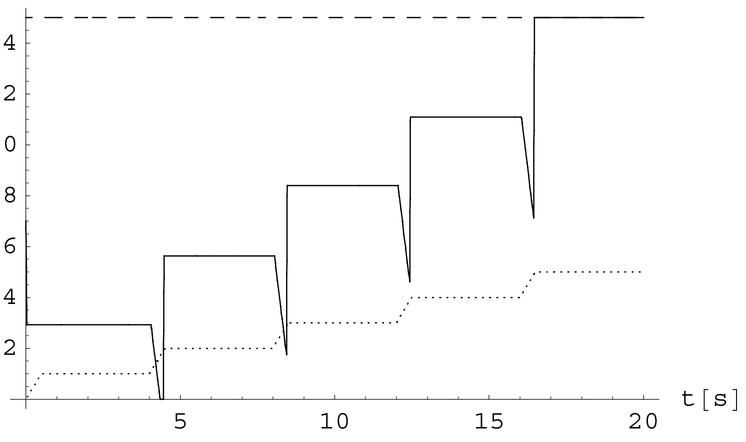


Figure 5.13. Rotational speed into (dashed) and out of (solid) gearbox plus gear lever signal (dotted).

As seen in the figure all the gears work fine. The gear changes will be discussed in the neutral gear section below.

5.2.5 Neutral Gear

Modeling of neutral gear is not a necessity because when the gearbox is simulated in a driveline it is connected with a clutch. Neutral gear can in this case be modelled by disengaging the clutch. However, with the implementation of the gearbox as in 5.2.3 we can achieve neutral gear by letting all the clutches in the gearbox model be disengaged at the same time. In the implementation of the gearbox model all the clutches have been designed to be disengaged between the different gears (see Appendix A.2). When the clutch signal is ramped from 2 to 3 the gearbox will go from gear number 2 via neutral to gear 3. Figure 5.13 shows what happens during gear changing. When going from a gear to neutral the outgoing shaft slows in and when going from neutral to the next gear the speeds starts to synchronize and finally the new gear will be engaged.

5.3 Final Drive

This section will treat the final drive and its parts. The ideal differential and the equations describing it will be discussed detailed. The model of the final drive will be discussed and evaluated.

5.3.1 Ideal Differential

In cornering the inner and the outer wheels have different speeds. However, driven wheels connected to the engine via the driveshafts must both be turned by gearing and this gear train must thus allow differential motion of the inner wheel with respect to the outer wheel. Figure 5.14 shows a simple illustration of an automotive differential.

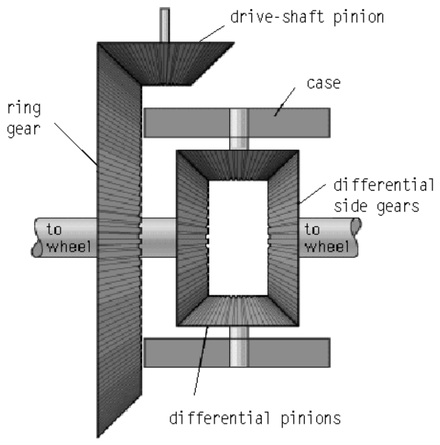


Figure 5.14. Illustration of a simple automotive differential.

The differential applies equal torque to the left wheel and to the right wheel

$$M_{f,l} = M_{f,r} \tag{5.7}$$

The gear ratios give the equation for the turning angles

$$\frac{\theta_{f,in}}{i_f} = \frac{\theta_{f,l} + \theta_{f,r}}{2} \tag{5.8}$$

For an ideal differential the power in to the differential is equal to the power out of the differential. The power P is equal to the product of the torque and the angular velocity, and thus we get

$$M_{f,in}\,\omega_{f,in} = M_{f,l}\,\omega_{f,l} + M_{f,r}\,\omega_{f,r} \tag{5.9}$$

By deriving equation 5.8 and combining with equation 5.7 and 5.9 we get the relation between the torques

$$M_{f,in}\,i_f = M_{f,l} + M_{f,r} \tag{5.10}$$

Thus the equations describing the ideal differential can be written

$$\begin{aligned} \theta_{f,in}\frac{1}{i_f} &= \frac{\theta_{f,l}+\theta_{f,r}}{2} \\ M_{f,in}\, i_f &= M_{f,l}+M_{f,r} \\ M_{f,l} &= M_{f,r} \end{aligned} \tag{5.11}$$

where i_f is the gear ratio [12].

5.3.2 Modelling the Final Drive

The model of the final drive consists of a *GearEfficiency* connected to an ideal differential gear following the equations in 5.3.1. The structure can be seen i figure 5.15.

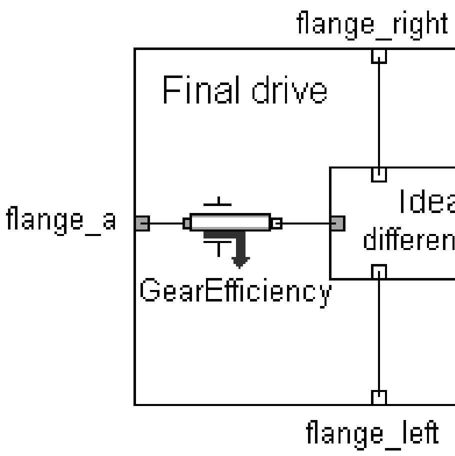


Figure 5.15. Structure of the final drive model.

The implementation of the final drive model in MathModelica can be seen in Appendix A.1.

5.3.3 Evaluation of the Final Drive Model

This test intends to show how the final drive model works. This is done by applying different torques on the different shafts and study the rotational speed for respective shaft. The applied torques is constant or changed in steps, to make the plots easy to verify. A constant torque, 1 Nm, is driving the final drive. An inertia is connected to left respective right outgoing shaft. The gear efficiency is 90% and $i_f = 2.52$.

Between 0 and 4 seconds there is no torque braking or driving the left or the right inertia, and the two outgoing shafts rotates with the same speed.

At $t = 4\,s$ the braking torque on the right inertia steps from 0 to 2.268 Nm. When the braking torque is applied the right side stops accelerating while the left side continues its constant acceleration. Figures 5.16 and 5.17 shows the rotational speed respective torque for the simulation.

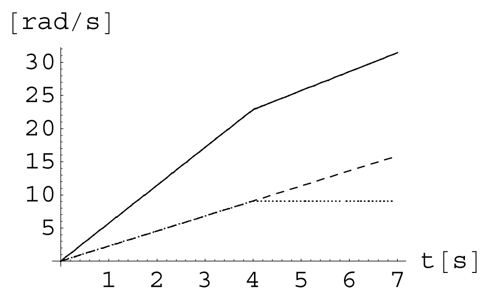


Figure 5.16. Rotational speed for the final drive, $\omega_{f,in}$ (solid), $\omega_{f,l}$ (dashed) and $\omega_{f,r}$ (dotted).

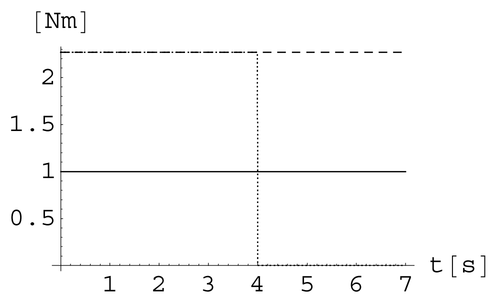


Figure 5.17. Torque for the final drive, $M_{f,in}$ solid, the resulting torque on left (dashed) respective right (dotted) inertia.

It is thus shown that the final drive allows the inner and outer wheels to be driven and to have different speeds. The final drive distributes the driving torque equal to the wheels and depending on the load on each wheel they get different speeds.

6

Modelling Shafts and Brakes

This section will concentrate on modelling the shafts that transfers the torque between the different parts of the driveline and finally to the wheel. The brake model will also be discussed here.

6.1 Propeller Shaft

The propeller shaft, which only exists on rear wheel driven cars with the engine in the front, connects the gearbox to the final drive and thus transfers the energy from the front end of the car to the rear. In reality the propeller shaft is elastic, but in comparison to the drive shafts it is less elastic (see section 6.2). Thus, to save time in simulation the simplification that the propeller shaft is stiff has been done. The equation describing the propeller shaft becomes very simple

$$\mathtt{M_{p,in} = M_{p,out}} \tag{6.1}$$

and since the propeller shaft is rigid the turning angle θ_p is the same along the shaft.

This equation describes a direct connection, compare to section 2.2.2, so there is no need to create an object for the propeller shaft, instead the gearbox and final drive can be directly connected. This assumption also gives the freedom to see the car as either front wheel driven or rear wheel driven without any changes in the models.

If one would want a model with an elastic propeller shaft the same component as used in the modelling of the drive shafts could be used, but with different parameters. See section 6.2 for the drive shaft model.

6.2 Drive Shafts

The drive shafts cannot be assumed to be stiff. In fact, because the drive shaft is relatively simple and inexpensive to repair, it is often dimensioned to be the weakest part of the driveline. Therefore the flexibility of the drive shafts will give rise to driveline oscillations [13]. The drive shafts are modelled as damped torsional flexibilities, having stiffness *k*, and internal damping *c*. Assuming there is no friction, the driving torque from the final drive $M_{d,in}$, will be equal to the braking torque $M_{d,out}$ from the wheel. Hence the model becomes

$$M_{d,out} = M_{d,in} = k\,(\theta_{d,in} - \theta_{d,out}) + c\,(\dot{\theta}_{d,in} - \dot{\theta}_{d,out}) \tag{6.2}$$

In the free Modelica standard library the component *SpringDamper* is built on equation 6.2 and can thus be used as drive shaft model. See [1] for further information about the *SpringDamper*.

A plot showing the elasticity in the drive shafts during a simulation in a european driving cycle (discussed in section 9.2) can be seen in Appendix B.

6.3 Brake

A model of a disc brake is implemented in the free Modelica standard library. The implementation of the brake and clutch models have a lot in common so the brake model will not be treated as extensive as the clutch model in section 5.1.

6.3.1 Modelica Standard Brake

The brake blocks are pressed against the disc with a normal force F_b. The normal force has to be provided as input signal u_b in a normalized form, $F_b = F_{b,max}\, u_b$, where $F_{b,max}$ (the maximum normal force F_b) has to be provided as parameter.

When the absolute angular velocity ω_b is not zero, the friction torque is a function of the velocity dependent friction coefficient $\mu_b(\omega_b)$, and of the normal force F_b

$$M_{b,k}(\omega_b) = M_{b,c,max}(\omega_b)\, u_b \tag{6.3}$$

$$M_{b,k,max}(\omega_b) = n\,\frac{r_o + r_i}{2}\,\mu_b(\omega_b)\, F_{b,max} \tag{6.4}$$

When the absolute angular velocity becomes zero, the elements connected by the friction element become stuck, i.e., the absolute angle remains constant. In this phase the friction torque is calculated from a torque balance due to the requirement, that the absolute acceleration shall be zero. The elements begin to slide when the friction torque exceeds a threshold value, called the maximum static friction torque, computed via:

$$M_{b,s} = c_{b,s,k}\, M_{b,k,max}(\omega_b = 0)\, u_b \tag{6.5}$$

This procedure is implemented in a "clean" way by state events and leads to continuous/discrete systems of equations if friction elements are dynamically coupled, see [1].

When the brake has locked the braking torque M_b has to be calculated so that

$$M_{b,in} - M_b - M_{b,out} = 0$$

The braking torque can thus not be greater than the torque acting on the brake. If this is not implemented correctly the wheel could be provided energy, and thereby begin to rotate while braking. Compare with the implementation of rolling resistance at zero velocity in section 7.2.

For a full description of the brake model see [1].

6.3.2 Test of Modelica Standard Brake

In a simple set up to see that the brake works as expected, the brake is driven with a torque of 2000 Nm and an *Inertia* is connected to the outgoing side of the brake. In a typical car with a maximum engine torque of 270 Nm and a gear ratio of 13 for the first gear, the torque after the gearbox can reach $270 \cdot 13 = 3510\,Nm$ and even more at braking. At $t = 2\,s$ the brake signal steps from zero to one. Figure 6.1 shows the rotational velocity and the friction torque in the brake. The braking torque is about 4300 Nm, i.e. $M_{b,k,max}$, as long as the inertia is rotating. When the inertia stops, the braking torque becomes 2000 Nm, making the resultant torque on the brake zero.

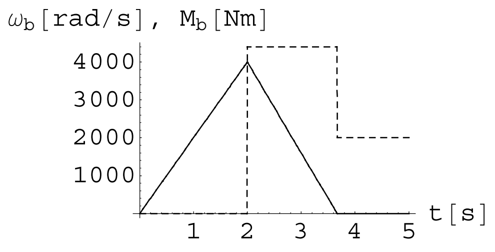


Figure 6.1. Rotational velocity ω_b (solid), and friction torque M_b (dashed) for the brake.

7

Modelling the Wheel

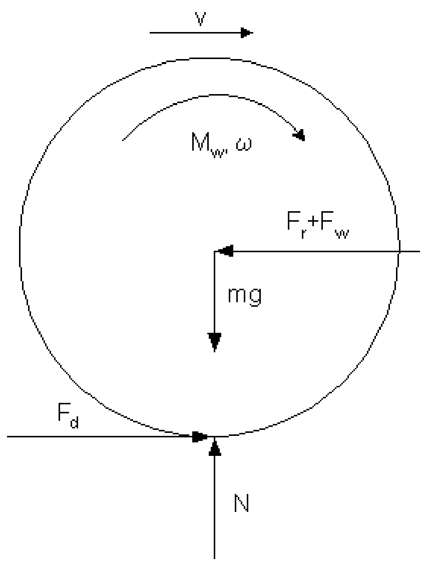


Figure 7.1. Torque and forces acting on wheel.

The forces and torque acting on the wheel can be seen in figure 7.1.

The forces are:

N	normal force from the ground
mg	gravitational force i.e. $(m_w + \frac{1}{4} m_v)\, g$
F_r	rolling resistance
F_w	braking force from the car
F_d	longitudinal friction force driving the wheel
M_w	driving torque from engine via the driveline

As the vehicle is only moving in longitudinal direction Newton's second law gives

$$\begin{aligned} N - \left(m_w + \frac{1}{4} m_v\right) g &= 0 \\ F_d - F_w - F_r &= m_w \dot{v} \\ M_w - F_d\, r_w &= J_w\, \dot{\omega}_w \end{aligned} \qquad (7.1)$$

An extension of the model to make the vehicle able to turn would require a more complex tyre model with lateral forces and a steering model, which would require several weeks of work.

7.1 Wheel with Rolling Condition

If it is assumed that the wheel is rolling, a wheel model can easily be built of Modelica standard components, where the component *IdealGearR2T* takes care of the transition from rotation to translation, with the rolling condition

$$\omega_w\, r_w = v \qquad (7.2)$$

The structure of the wheel model with rolling condition can be seen in figure 7.2.

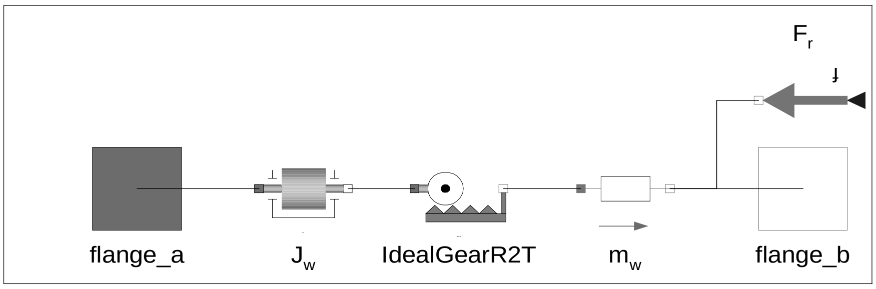


Figure 7.2. Structure of the wheel model with rolling condition.

The components on the left side of *IdealGearR2T* represent the rotation of the wheel and the components on the right side represent the translation.

To see how the rolling resistance F_r has been modelled, see section 7.2. Note that the point where F_r is acting on the wheel does not correspond with figure 7.1, F_r is causing a moment around the wheel axis that is not zero. This will be put straight in section 7.3, where slip also will be introduced to the model.

7.2 Rolling Resistance

The rolling resistance originates from tyre deformation. The centre of normal pressure is shifted in the direction of rolling, which produces a moment about the axis of rotation of the tyre, the rolling resistance moment. In a free-rolling tyre the applied wheel torque is zero, therefore a horizontal force at the tyre-ground contact patch must exist to maintain equilibrium. This horizontal force is generally known as the rolling resistance.

Here the rolling resistance will be treated as a force acting through the axis, see figure 7.1. When the vehicle is moving, v>0, a braking force that is dependent on the velocity, $F_r = c_0 + c_1\, v$, models the rolling resistance. When the vehicle stands still there is a resistance, $F_r = c_0$, that must be exceeded for the vehicle to start moving. However, if the driving force F_d is less than c_0 the vehicle is naturally not moving so the driving and the braking force have to match. Thus the rolling resistance is implemented as

$$F_r = \begin{cases} \min\,(F_d,\ c_0), & v = 0 \\ c_0 + c_1\, v, & v > 0 \end{cases} \tag{7.3}$$

7.3 Tyre Model for Slip

This section will discuss a tyre model with slip, some difficulties with the modelling and proposals for improvements.

7.3.1 Longitudinal Slip

Driven wheels do not roll. The rotational velocity is faster than the corresponding translational velocity, because of longitudinal slip described by

$$s = \frac{r_w\, \omega_w - v}{r_w\, \omega_w} \tag{7.4}$$

During braking the translational velocity is slower than the corresponding rotational velocity that corresponds to negative slip. Note that the slip is not defined for $\omega_w = 0$.

The organization SAE uses a different definition for slip, $s = \frac{r\omega - v}{v}$ [14], but this expression is not defined for v=0.

7.3.2 Longitudinal Force

Except for low velocities, the longitudinal force is a function of the longitudinal slip $F_d = f(s) = N\mu(s)$, where N is the normal force acting on the wheel, and μ(s) is the friction coefficient as a function of slip according to the slip curve in figure 7.3.

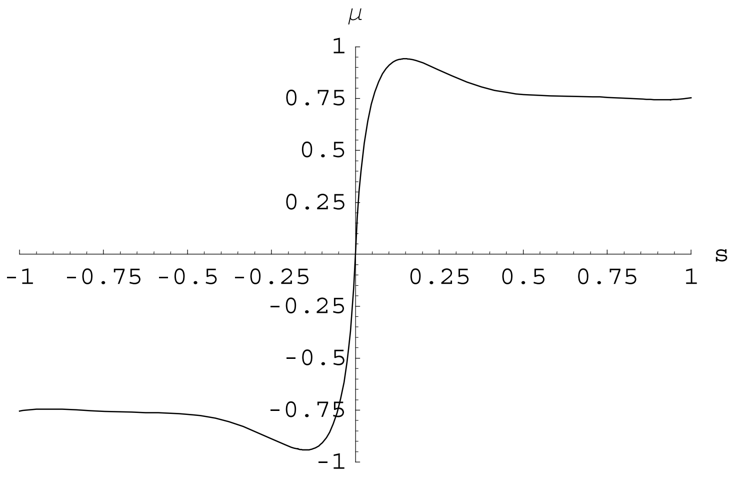


Figure 7.3. Slip curve describing the friction between tyre and ground for longitudinal motion as function of slip.

The slip curve has been implemented as a table. For low velocities (less than about 35 km/h) and stand still the longitudinal force can be described as $F_d = C(r_w\, \omega_w - v)$ [14]. The problem when simulating is the transition between the two force models. A solution to this problem that has been used here is to make an interval where the influence from one model is increasing whereas the other is decreasing.

The complete expression for the longitudinal force can thus be written

$$F_d = \begin{cases} C\,(r_w\,\omega_w - v)\,, & \omega_w \le 28 \text{ rad/s} \\ \frac{1}{2}\,C\,(r_w\,\omega_w - v)\,\left(\cos\left(\frac{\omega_w - 28}{4}\,\pi\right) + 1\right) + \\ \quad + \frac{1}{2}\,N\mu\,(s)\,\left(\cos\left(\frac{32-\omega_w}{4}\,\pi\right) + 1\right), & 28 < \omega_w < 32 \text{ rad/s} \\ N\mu\,(s)\,, & \omega_w \ge 32 \text{ rad/s} \end{cases} \tag{7.5}$$

By using the sigmoid function $\frac{1}{2}(\cos(x\pi) + 1)$, $x \in [0, 1]$, a smooth transition where both F_d and $\dot{F}_d$ is continuous is obtained.

The implementation of the wheel model in MathModelica can be seen in Appendix A.3.

7.4 Evaluation of the Wheel Model

A simple setup is used to simulate the behaviour of the wheel at start and stop. Between zero and six seconds a torque drives the wheel. The wheel is connected to a sliding mass, modelling the car, and is only braked by the air drag (section 8.2).

Start

Figures 7.4 and 7.5 shows the driving force and rolling resistance respective the translational velocity for the wheel during the first 0.15 seconds of the start phase. The rolling resistance is equal to the driving force for $F_d < c_0 = 11$ and thus is the wheel not beginning to move until the driving force exceeds the threshold value c_0.

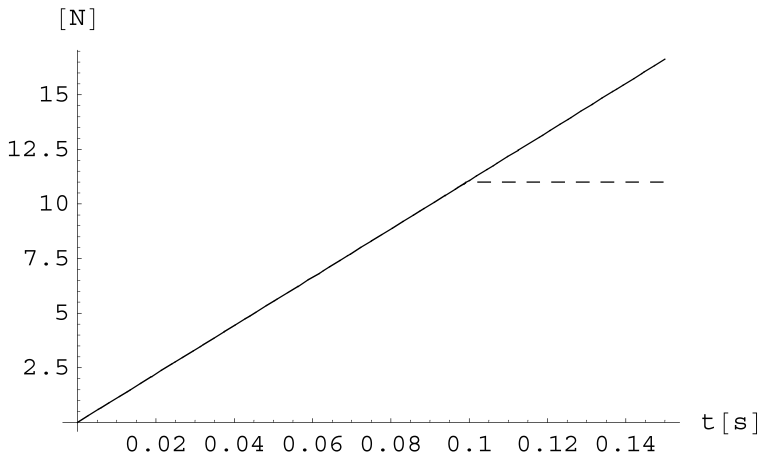


Figure 7.4. Driving force F_d (solid) and rolling resistance F_r (dashed) for the first 0.15 seconds.

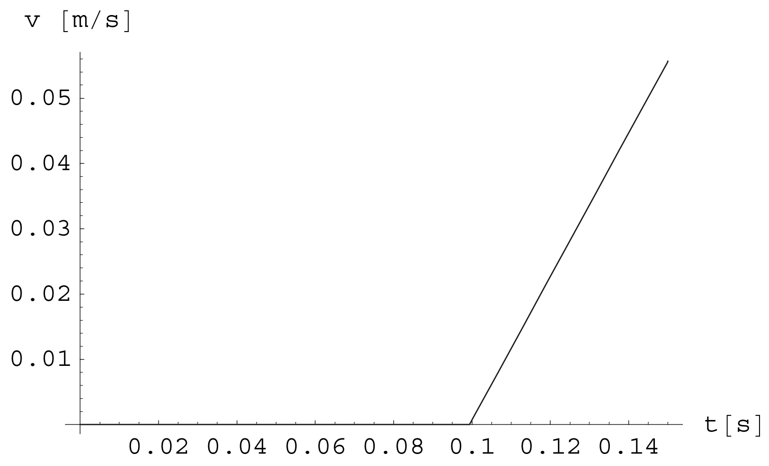


Figure 7.5. Velocity v for the first 0.15 seconds.

Stop

After 6 seconds the driving torque is zero and the wheel is braked by the air drag. The driving force F_d is very small in comparison to the rolling resistance. Figures 7.6 and 7.7 shows the rolling resistance and velocity for the wheel during the stop phase. The wheel stops smoothly after 118.65 seconds and the rolling resistance becomes zero.

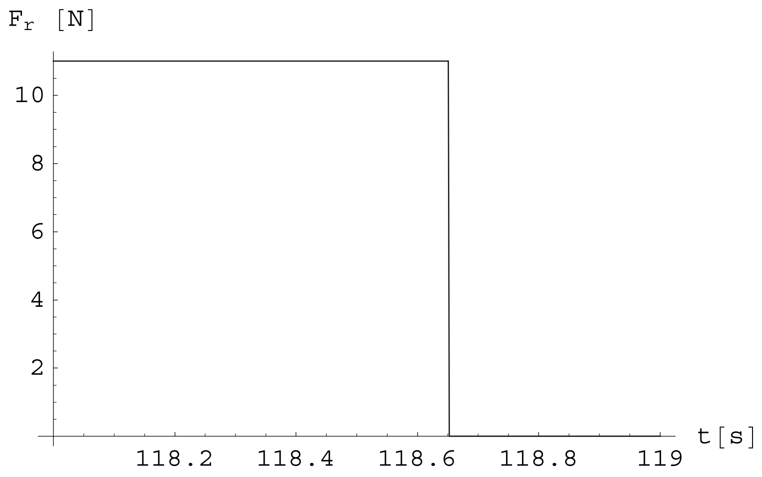


Figure 7.6. Rolling resistance F_r during the stop phase.

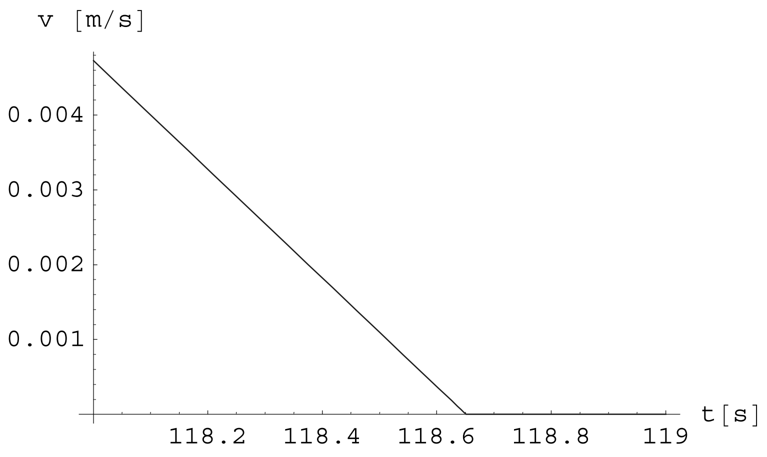


Figure 7.7. Velocity v for the wheel during the stop phase.

8

Vehicle Model

This chapter will treat the models that need to be connected to the driveline model when simulating the vehicle in a driving cycle. These models will not be discussed so thorough as the the previous models, but intends to show how the driveline is affected by them.

8.1 Chassis

For a detailed description of the automobile chassis with suspension and rigid body model see [15]. However for the longitudinal driveline model it is sufficient to treat the chassis as a sliding mass that follows Newtons second law. An illustration of the forces acting on the chassis can be seen in figure 8.1. The motion of the chassis can be described by

$$F_c - F_a = m_v \dot{v} \tag{8.1}$$

The implementation of the chassis model in MathModelica can be seen in Appendix A.4.

To save simulation time the influence of the gravitational force has been neglected, i.e. the car is assumed to be driving on a road without slope. If one would like to do simulations with the vehicle model driving on road with slopes, for example for fuel consumption, this could be easily be implemented in chassis model by introducing the longitudinal component of the gravitational force $m_v\, g \sin\alpha$, where α [rad] is the slope of the road.

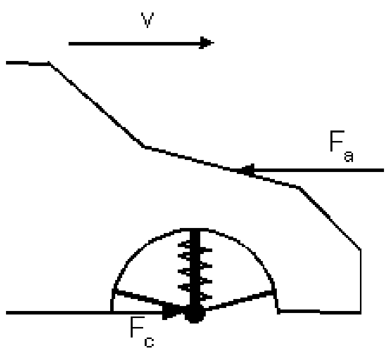


Figure 8.1. Forces acting on vehicle.

8.2 Air Drag

The braking force from the air drag is proportional to the squared velocity

$$F_{air} = c_D\, A\, \frac{\rho}{2}\, v_\infty^2 \qquad [14]$$

where A is the front area of the vehicle, ρ is the air density and v_∞ is the difference between the vehicle velocity, v, and the wind velocity, v_s. c_D is a dimensionless aerodynamic coefficient that is only dependent on other dimensionless constants, e.g. the Reynolds number and the angle β (Figure 8.2).

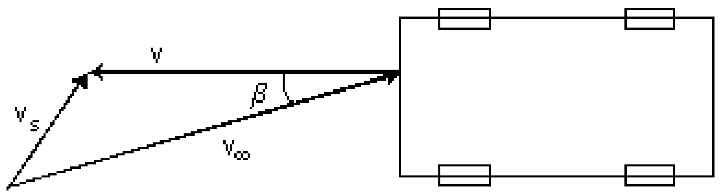


Figure 8.2. Velocity of wind relative to the vehicle.

Here the air drag model has been simplified by neglecting the wind velocity giving the expression

$$F_{air} = c_2 v^2 \tag{8.2}$$

where the constants have been lumped together, called c_2. Under normal weather conditions the influence from the wind speed is small compared to the vehicle speed.

The MathModelica implementation of the air drag model can be seen in Appendix A.5.

8.3 Engine

A physically justified engine model for spark ignited engines is used for the torque. The engine torque is parameterised to capture the following features:
1. The maximum torque $M_{e,max}$.
2. The torque decrease at high engine speed.
3. Idling and idle speed control.
4. Engine braking at speeds over idling.

The model is derived from an ideal thermodynamic cycle that neglects residual gases but includes pumping work, [10]. The basis for the parameterisation is to determine a lowest air mass flow. The focus on air mass flow can be motivated since nowadays many engine management systems have drive-by-wire systems that implement air mass flow control instead of just throttle plate position control.

The torque developing engine dynamics is modelled as a first order system with a time constant of 0.1 s from the demanded torque.

$$M_e = \frac{1}{0.1\,s + 1} M_{demand}$$

M_{demand} is the non-linear function, shown in figure 8.3, of the engine speed and the accelerator pedal signal, $u_e \in [0,1]$, from the driver.

The lines in the plot represents constant u_e. The idle speed is set to 13.3 RPS (800 RPM), which is seen in the plot, $M_{demand}(13.3, 0) = 0$ Nm. The maximum torque is 270 Nm and the engine speed where M_{demand} starts to decrease is 58.3 RPS (3500 RPM).

The demanded torque M_e, has been derived from the following equations.

$$m_{ac,demand} = u_e \left(m_{ac,max} - m_{ac,min}\right) + m_{ac,min}$$

$$p_{i,demand} = \frac{1}{2\pi} \omega_e$$

$$p_i = \min\left(p_{amb},\ \frac{c_0\, m_{ac,demand}}{p_{i,demand}}\right)$$

$$M_{demand} = c_1\, p_i + c_2\,(p_{amb} - p_i)$$

For the interested reader there is a lot of work done in engine modelling. For example [16] or [17] can be of interest.

The implementation of the engine model in MathModelica can be seen in Appendix A.6.

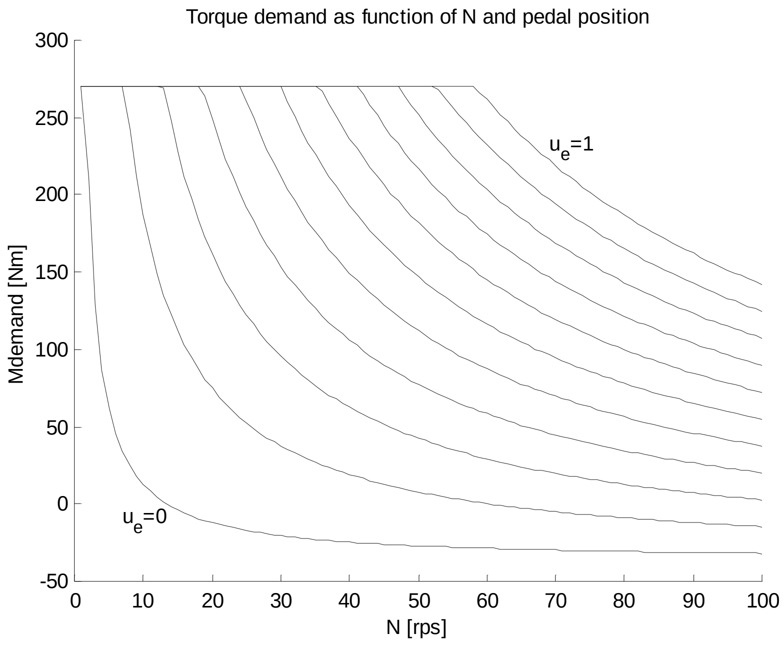


Figure 8.3. Engine torque as function of engine speed and accelerator pedal position.

8.4 Driver

The driver model consists of a table with reference speed, gear signal and clutch signal with respect to time, and a PI controller. The output from the driver model is the gear and clutch signals together with signals for accelerator and brake pedal, the driver is also provided with the actual speed (figure 8.4).

The main part of the driver is the PI controller. The purpose of the PI controller is to determine how much the driver shall accelerate or brake the car. The input to the PI controller is the difference between the demanded speed and the actual vehicle speed.

The output ranges from -1 to +1 where -1 represents maximum braking and +1 represents maximum acceleration. Three characteristics that reflects the driver is:

1. The driver cannot press the brake and the accelerator pedal at the same time. Taking the signal to the accelerator pedal as maximum of zero and the outport from the PI controller solves this. In the same way the signal to the brake pedal is the minimum of zero and the output from the PI controller (after that the sign is changed so the braking signal $\in [0, 1]$).

2. When the driver pushes the clutch down he also raises the accelerator pedal. This is implemented by multiplying the signal to the accelerator pedal with the clutch signal.

3. When a new gear is in place and the clutch is engaged, the information from the last gear is forgotten. Resetting the integral part of the PI controller does this.

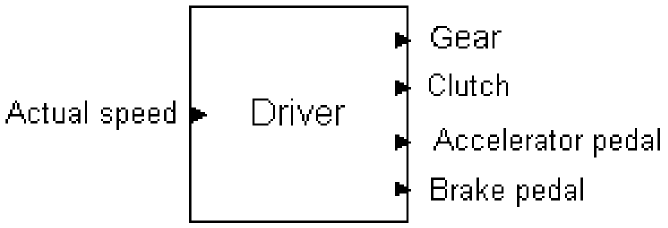


Figure 8.4. Signals in to and out from the driver.

The implementation of the driver model in MathModelica can be seen in Appendix A.7.

The biggest disadvantage with the approach with a PI-controller is that a real driver looks ahead and can adjust the speed before the changes in speed limits, while the PI-controller only reacts when the differences between vehicle and reference speed has occured. The driver model based on the PI-controller thus behaves more like a cruise control. For more information about driver models see [18].

9

The Complete Driveline Model

In this chapter the parts described in the previous chapters will be connected to each other, forming a complete driveline model together with driver and vehicle model. The model is simulated in a european driving cycle and some interesting variables are plotted and the behaviour of the driveline is discussed.

9.1 Connecting the Parts

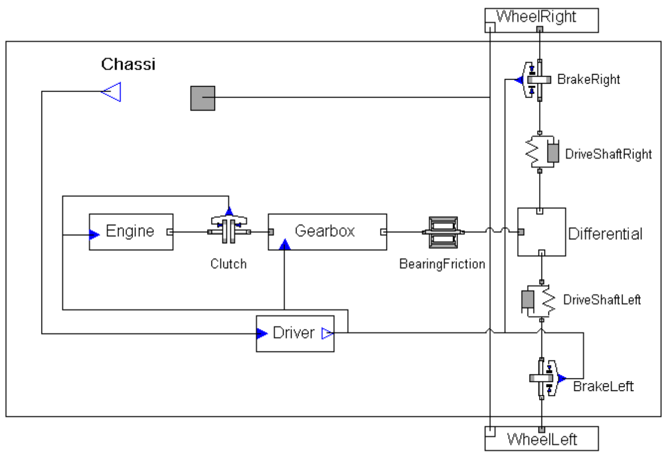


Figure 9.1. Structure of the driveline and vehicle model.

The driveline parts will be connected as showed in figure 9.1. The implementation of the driveline model in MathModelica can be seen in Appendix A.8. The Connect-statements discussed in section 2.2.2 gives the following relations between the components:

Connecting the engine to the clutch

$$\theta_{e} = \theta_{c,in}$$

$$M_{e} = M_{c,in}$$

Connecting the clutch to the gearbox

$$\theta_{c,out} = \theta_{g,in}$$

$$M_{c,out} = M_{g,in}$$

Connecting the gearbox to the propeller shaft

$$\theta_{g,out} = \theta_{p}$$

$$M_{g,out} = M_{p,in}$$

Connecting the propeller shaft to the final drive

$$\theta_{p} = \theta_{f,in}$$

$$M_{p,out} = M_{f,in}$$

Connecting the final drive to the left and right driveshaft

$$\theta_{f,l} = \theta_{d,in,left}$$

$$\theta_{f,r} = \theta_{d,in,right}$$

$$M_{f,l} = M_{d,in,left}$$

$$M_{f,r} = M_{d,in,right}$$

Connecting the driveshafts to respective brake

$$\theta_{d,out} = \theta_{b}$$

$$M_{d,out} = M_{b,in}$$

Connecting the brakes to respective wheel

$$\Theta_b = \Theta_w$$

$$M_{b,out} = M_w$$

Connecting the wheels to the chassis

$$F_{w,left} + F_{w,right} = F_c$$

Note that only the two driving wheels has been included in the vehicle model. This does not affect the behaviour when accelerating the car but during braking we only get half the braking force. It would not be too much difficulties to implement an undriven wheel with a brake and connect two of these to the chassis but this has not been done here, mainly because not enough time has been available and that it is not so interesting until the vehicle has been modelled with suspension and a system that distributes the brake force, so that the vehicle dynamics can be simulated. Under a not too tough driving cycle the lack of front wheels can be compensated by the driver pressing the brake pedal twice as much or by doubling the size of $F_{b,\max}$ in the brake model. This is OK as long as the wheel do not lock.

In Appendix A, the implementation of the driveline model in MathModelica can be seen.

9.2 Simulating the Driveline and Vehicle Model

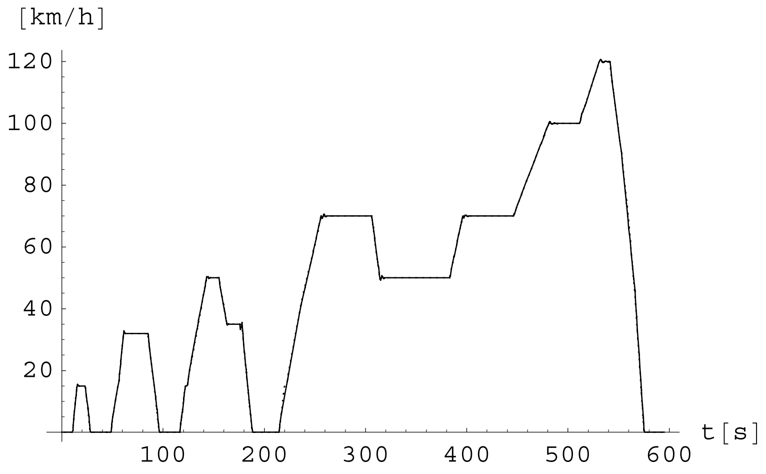


Figure 9.2. Vehicle speed (solid) and reference speed (dotted) during a european driving cycle.

The driveline model has been simulated in a new european driving cycle (NEDC) [19], lasting for 595 seconds. The european driving cycle defines the speed and which gear the car shall be driven with. Figure 9.2 shows the reference speed that the car shall follow, together with the real vehicle speed. The difference between the reference speed and the vehicle speed is mainly due to the driver model discussed in section 8.4.

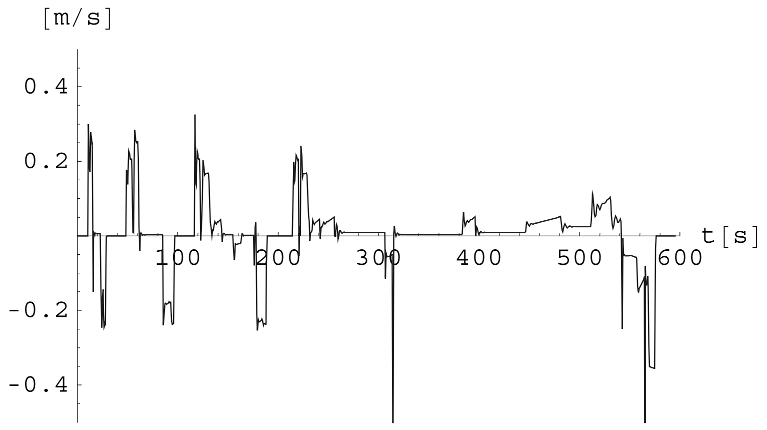


Figure 9.3. Difference between corresponding rotational wheel speed and vehicle speed, i.e. $\omega_w r_w - v$.

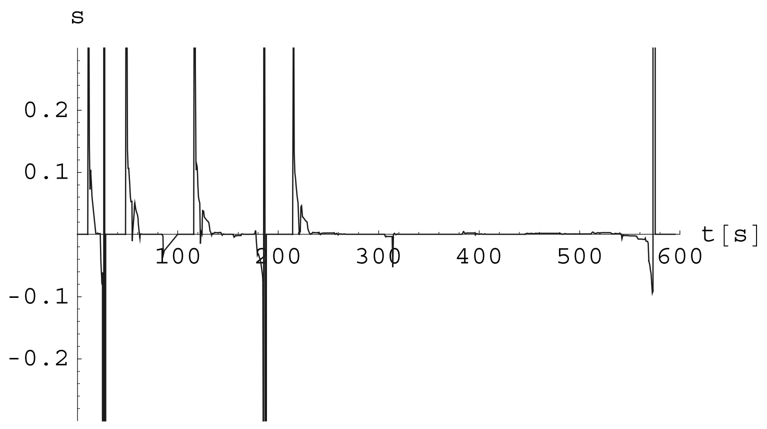


Figure 9.4. Slip during the european driving cycle.

Figure 9.3 and 9.4 shows the difference between the vehicle speed and the corresponding rotational velocity respective the slip, during the european driving cycle. As expected the slip (and the difference) is greatest when accelerating or braking, compare with figure 9.2. The peaks in the slip plot springs from the vehicle and/or the wheel standing still.

Some additional plots from the european driving cycle can be seen in Appendix B.

The simulation takes 2 minutes and 50 seconds on a 120 MHz Pentium with 160 MB RAM. This is not a long time but plotting one simulation takes about the same time (±50 %).

10

Conclusions and Recommendations

The aim of this report was to model a vehicle driveline using the MathModelica environment, and to simulate the driveline together with driver, engine and vehicle model in longitudinal motion during a driving cycle. Chapter 2 focuses on the MathModelica environment and the Modelica language and together with chapter 3, which shortly describes the methods behind physical modeling, it prepares the reader for the modeling discussions.

In chapter 5 the standard Modelica clutch is evaluated and verified to meet the demands on the automotive clutch model. The manual gearbox is modeled by using components from the standard Modelica library. A model of the final drive is created from the equations describing an ideal differential gear and completed with standard components. Chapter 6 models shafts and brakes. It is shown that this can be done by only using components from the standard Modelica library.

Chapter 7 is devoted to the wheel model. First a simple wheel model with rolling condition and rolling resistance is builded from standard components. This model is developed to a wheel model that handles longitudinal slip, and the point where the rolling resistance acts on the wheel is corrected. Some problems regarding the model for the longitudinal force was discussed. It was shown that the slip and rolling resistance is modelled correct.

In chapter 8 the peripheral models (engine, driver and vehicle) are discussed and in chapter 9 the whole model is simulated in a european driving cycle. The results shows the expected behaviours, however since no measurements has been available it is hard to say how near the truth the model is, just that the qualitative behaviour is correct.

The MathModelica environment has turned out to be a very powerful tool in this type of work. The object-orientation makes it easy to change models and reuse components and also to describe the model with equations. The components from the standard Modelica library covers the most usual problems so the modeller can concentrate on the bigger ones. The models, simulations and documentation is done in Mathematica notebooks making the work well integrated. For the future, more component libraries and better debugging will be needed to simplify the modellers work.

10.1 Recommended Future Work

Gravitational force - For more realistic simulations it is necessary to implement the effect of the gravitational force in the wheels and the chassis, so driving cycles with sloping roads can be implemented. This could be done quite easily by introducing the longitudinal component of the gravitational force in the chassis model.

Longitudinal force model - The force model needs to be verified against measured data. Especially the transition between the two force models is important to tune.

Measurements - Naturally, to make a good model it must be verified with some data. The most important things to decide in this model is where and how big the friction is, efficiency losses, the parameters for stiffness and damping in the drive shafts and the tyre model with longitudinal driving force.

Propeller shaft - A natural extension is to model the flexibility in the propeller shaft. This is easily done by using a *SpringDamper* like in the drive shafts.

Reversing car - In this model the car has been assumed to only be driving forward or standing still. For a more general driveline and vehicle model the model must be extended to be able to reverse the car. This would require work as well on the chassis and air drag models as on the wheel. The gearbox is easily extended with reverse by adding an ideal gear with negative gear ratio.

Vehicle dynamics - This modelling work can also be the foundation to a more developed vehicle model with dynamics. For example, the behaviour of the car during turning could be simulated. This work can be done in several steps, starting for example with making the car turning.

Front wheels - To get a better vehicle model, especially when the car is braking, front wheels with brakes should be connected to the chassis. Introducing front wheels would probably not demand any new models to be developed. Front wheels is naturally also needed if the vehicle model is made to be turning.

11 References

[1] Modelica Association
<http://www.modelica.org>

[2] MathCore
<http://www.mathcore.se>

[3] Modelica Association, *Modelica™ - A Unified Object Oriented Language for Physical Systems Modeling, Language Specification version 1.4*, 2000
<http://www.modelica.org/current/ModelicaSpec14.pdf>

[4] MathWorks
<http://www.mathworks.com>

[5] Idebrant A., *Multidomänmodellering och simulering av jaktrobot*, Master thesis: LiTH-ISY-EX-3085, Linköping University, 2000

[6] Wolfram Research
<http://www.wolfram.com>

[7] Dynasim AB
<http://www.dynasim.se>

[8] Ljung L., Glad T., *Modellbygge och simulering*, Studentlitteratur 1999, ISBN 91-44-31871-5

[9] Karnopp D.C., et.al., *System Dynamics, Modeling and Simulation of Mechatronic Systems*, Third Edition, John Wiley & Sons, 2000, ISBN 0-471-33301-8

[10] Eriksson L., *Simulation of a vehicle in longitudinal motion with clutch lock and clutch release*, Linköping University, Vehicular Systems, ISY, 2000

[11] Olsson H., et.al., *Friction models and friction compensation*, European Journal of Control 4(3) p.176-195, 1998

[12] Drogies S., Bauer M., *Modeling Road Vehicle Dynamics with Modelica*, 2000 <http://www.modelica.org/workshop2000/proceedings/Drogies.pdf>

[13] Pettersson M., *Driveline modeling and control*, Doctoral thesis, Linköpings University, 1997, ISBN 91-7871-937-2, ISSN 0345-7524, Dissertation No. 484

[14] Nielsen L., Eriksson L., *Course material Vehicular Systems*, Linköping University, Vehicular Systems, ISY, 1999

[15] Dixon J.C., Tires, Suspension and Handling, Second Edition, SAE, 1996, ISBN 1-56091-831-4

[16] Brugård J., Bergström J., *Modeling of a Turbo Charged Spark Ignited Engine*, Master thesis: LiTH-ISY-EX-2081, Linköping University, 1999

[17] Pettersson F., *Simulation of a Turbo Charged Spark Ignited Engine*, Master thesis: LiTH-ISY-EX-3010, Linköping University, 2000

[18] Pahv V., *En longitudinell förarmodell*, Master thesis: LiTH-ISY-EX-3115, Linköping University, 2001

[19] André M., et al., *Development of short driving cycles -Short driving cycles for the inspection of in-use cars -Representative European driving cycles for the assessment of the I/M schemes*, INRETS Report LEN 9809, May 1998 <http://europa.eu.int/comm/environment/pollutants/inusecars2.pdf>

Appendix A

The Models Implemented in MathModelica

A.1 Final Drive

```
Model[FinalDrive, "Model of final drive",
 Modelica.Mechanics.Rotational.Interfaces.Flange_a flange_a;
 Modelica.Mechanics.Rotational.Interfaces.Flange_b
  flange_left;
 Modelica.Mechanics.Rotational.Interfaces.Flange_b
  flange_right;
 Modelica.Mechanics.Rotational.GearEfficiency
  gearEfficiency[{eta == 0.9}];
 IdealDifferential idealDifferential;
 Equation[
  Connect[flange_a, gearEfficiency.flange_a];
  Connect[gearEfficiency.flange_b,
   idealDifferential.flange_a];
  Connect[idealDifferential.flange_left, flange_left];
  Connect[idealDifferential.flange_right, flange_right];
 ]
]
```

```
Model[IdealDifferential, "Model of an ideal differential",
 Modelica.Mechanics.Rotational.Interfaces.Flange_a flange_a;
 Modelica.Mechanics.Rotational.Interfaces.Flange_b
  flange_left;
 Modelica.Mechanics.Rotational.Interfaces.Flange_b
  flange_right;
 Parameter Real ratio == 2.52;
 Equation[
   (flange_a.phi)      (flange_left.phi) + (flange_right.phi)
  ---------------- == ---------------------------------------- ;
       ratio                              2
  flange_a.tau * ratio == -flange_left.tau;
  flange_left.tau == flange_right.tau;
 ]
]
```

A.2 Gearbox

```
Model[Gearbox, "Model of a gearbox",
 Modelica.Mechanics.Rotational.Interfaces.Flange_a flange_a;
 Modelica.Blocks.Interfaces.InPort inPort[{Final[n == 1]}];
 Modelica.Mechanics.Rotational.Inertia J1[{J == 0.01}];
 Modelica.Mechanics.Rotational.Clutch
  Synchro1[{fn_max == 12000}];
 Modelica.Mechanics.Rotational.IdealGear
  gear1[{ratio == 5.127}];
 Modelica.Mechanics.Rotational.Clutch
  Synchro2[{fn_max == 12000}];
 Modelica.Mechanics.Rotational.IdealGear
  gear2[{ratio == 2.667}];
 Modelica.Mechanics.Rotational.Clutch
  Synchro3[{fn_max == 12000}];
 Modelica.Mechanics.Rotational.IdealGear
  gear3[{ratio == 1.786}];
 Modelica.Mechanics.Rotational.Clutch
  Synchro4[{fn_max == 12000}];
 Modelica.Mechanics.Rotational.IdealGear
  gear4[{ratio == 1.353}];
 Modelica.Mechanics.Rotational.Clutch
  Synchro5[{fn_max == 12000}];
 Modelica.Mechanics.Rotational.GearEfficiency
  gearEfficiency[{eta == 0.8}];
 Modelica.Mechanics.Rotational.BearingFriction
  bearingFriction[{tau_pos == {{0, 0.1}}}];
 Modelica.Mechanics.Rotational.Inertia J2[{J == 0.01}];
 Modelica.Mechanics.Rotational.Interfaces.Flange_b flange_b;
 Equation[
  Connect[flange_a, J1.flange_a];
  Connect[J1.flange_b,  Synchro1.flange_a];
  Connect[J1.flange_b,  Synchro2.flange_a];
  Connect[J1.flange_b,  Synchro3.flange_a];
  Connect[J1.flange_b,  Synchro4.flange_a];
  Connect[J1.flange_b,  Synchro5.flange_a];
  Connect[Synchro1.flange_b, gear1.flange_a];
  Connect[Synchro2.flange_b, gear2.flange_a];
  Connect[Synchro3.flange_b, gear3.flange_a];
```

```
   Connect[Synchro4.flange_b, gear4.flange_a];
   Connect[gear1.flange_b, gearEfficiency.flange_a];
   Connect[gear2.flange_b, gearEfficiency.flange_a];
   Connect[gear3.flange_b, gearEfficiency.flange_a];
   Connect[gear4.flange_b, gearEfficiency.flange_a];
   Connect[Synchro5.flange_b, gearEfficiency.flange_a];
   Connect[gearEfficiency.flange_b,
    bearingFriction.flange_a];
   Connect[bearingFriction.flange_b, J2.flange_a];
   Connect[J2.flange_b, flange_b];
   Synchro1.inPort.signal[[1]] ==
    If[inPort.signal[[1]] < 1.1, 1, 0];
   Synchro2.inPort.signal[[1]] ==
    If[And[inPort.signal[[1]] ≥ 1.9, inPort.signal[[1]] < 2.1],
     1, 0];
   Synchro3.inPort.signal[[1]] ==
    If[And[inPort.signal[[1]] ≥ 2.9, inPort.signal[[1]] < 3.1],
     1, 0];
   Synchro4.inPort.signal[[1]] ==
    If[And[inPort.signal[[1]] ≥ 3.9, inPort.signal[[1]] < 4.1],
     1, 0];
   Synchro5.inPort.signal[[1]] ==
    If[inPort.signal[[1]] > 4.9, 1, 0];
  ]
 ]
```

A.3 Wheel

```
Model[Wheel, "Model of wheel with longitudinal slip",
 Modelica.Mechanics.Rotational.Interfaces.Flange_a flange_a;
 Modelica.Mechanics.Translational.Interfaces.Flange_b
  flange_b;
 ModelicaAdditions.Tables.CombiTable1D
  table[{fileName == "c:/home/perno586/forcetable",
    tableName == "forcetab", icol == {2}}];
 Parameter Real rw == 0.3;
 Parameter Real c0 == 11;
 Parameter Real c1 == 0.3604;
 Parameter Real Jw == 0.1;
```

```
  Parameter Real m_w == 15;
  Modelica.SIunits.AngularVelocity w;
  Modelica.SIunits.AngularAcceleration a_rot;
  Modelica.SIunits.Velocity v;
  Modelica.SIunits.Acceleration a_tr;
  Real s[{Start == 0, Final[max == 1], Final[min == -1]}];
  Modelica.SIunits.Force F_d;
  Modelica.SIunits.Force F_1;
  Modelica.SIunits.Force F_2;
  Modelica.SIunits.Force F_v;
  Modelica.SIunits.Force F_r;
  Equation[
   w == (flange_a.phi)';
   a_rot == w';
   v == (flange_b.s)';
   a_tr == v';
   s == If[Abs[w] < 0.01, 0,
     1 - (Sign[w] v)/((r_w Abs[w] + Modelica.Constants.small))];
   J_w a_rot == flange_a.tau - F_d r_w;
   m_w a_tr == F_d - F_v - F_r;
   table.inPort.signal[[1]] == s;
   F_1 == 3000 (r_w w - v);
   F_2 == ((1520 * 10)/4 + m_w) table.outPort.signal[[1]];
   F_d == If[w ≤ 28, F_1,
     If[w ≥ 32, F_2, F_1 1/2 (Cos[(w - 28)/4 Modelica.Constants.pi] + 1) +
       F_2 1/2 (Cos[(32 - w)/4 Modelica.Constants.pi] + 1)]];
   F_r == If[v < Modelica.Constants.small, Min[F_d, c_0],
     (c_0 + c_1 v)];
   flange_b.f == -F_v;
  ]
]
```

A.4 Chassis

```
Model[Chassis, "Chassis model",
 Modelica.Mechanics.Translational.Interfaces.Flange_a
  flange_a;
 Modelica.Mechanics.Translational.Interfaces.Flange_b
  flange_b;
 Modelica.Blocks.Interfaces.OutPort outPort[{Final[n == 1]}];
 Modelica.Mechanics.Translational.SlidingMass
  mass[{m == 1520}];
 Parameter Real m == 1520;
 Equation[
  Connect[flange_a, mass.flange_a];
  Connect[mass.flange_b, flange_b];
  outPort.signal[[1]] == mass.v;
 ]
]
```

A.5 Air Drag

```
Model[AirDrag, "Air drag model",
 Modelica.Mechanics.Translational.Interfaces.Flange_a
  flange_a;
 Modelica.Mechanics.Translational.Force force;
 Parameter Real c2 == 0.3896;
 Equation[
  Connect[flange_a, force.flange_b];
  force.inPort.signal[[1]] == -c2 * ((force.flange_b.s)')^2;
 ]
]
```

A.6 Engine

```
Model[Engine,
 "Simple Engine Model with characteristic torque",
 Modelica.Blocks.Interfaces.InPort
  inPort_pedal[{Final[n == 1]}];
 Parameter Real m_{ac,min} == 0.0021;
 Parameter Real m_{ac,max} == 0.0759;
 Parameter Real p_{amb} == 101.3 * 10^3;
 Parameter Real p_{i,min} == 1.222 10^4;
 Parameter Real T_{max} == 270;
 Modelica.SIunits.AngularVelocity ω_e;
 Real m_{ac,demand};
 Real p_{i,demand};
 Real p_i;
 Modelica.SIunits.Torque T_{demand};
 Modelica.Blocks.Continuous.TransferFunction
  transferFunction[{a == {0.1, 1}}];
 Modelica.Mechanics.Rotational.Torque torque;
 Modelica.Mechanics.Rotational.Inertia J_m[{J == 0.1}];
 Modelica.Mechanics.Rotational.Interfaces.Flange_b flange_b;
 Equation[
  m_{ac,demand} == inPort_pedal.signal[[1]] * (m_{ac,max} - m_{ac,min}) + m_{ac,min};
  ω_e == (flange_b.phi)';
  p_{i,demand} == 1/(2 * Modelica.Constants.pi) ω_e;
  p_i == Min[p_{amb}, (7.787 * 10^7 m_{ac,demand})/p_{i,demand}];
  T_{demand} == 0.00266535 p_i - 0.000365633 (p_{amb} - p_i);
  transferFunction.inPort.signal[[1]] == T_{demand};
  Connect[transferFunction.outPort, torque.inPort];
  Connect[torque.flange_b, J_m.flange_a];
  Connect[J_m.flange_b, flange_b];
 ]
]
```

A.7 Driver

```
Model[Driver, "Driver Model",
 ModelicaAdditions.Tables.CombiTableTime
  table[{fileName == "c:/home/perno586/drivertable",
    tableName == "drivertab", icol == {2, 3, 4}}];
 Modelica.Blocks.Interfaces.InPort
  inPort_ActualSpeed[{Final[n == 1]}];
 PI_Driver PIDriver;
 Modelica.Blocks.Interfaces.OutPort
  outPort_Gear[{Final[n == 1]}];
 Modelica.Blocks.Interfaces.OutPort
  outPort_Clutch[{Final[n == 1]}];
 Modelica.Blocks.Interfaces.OutPort
  outPort_GasPedal[{Final[n == 1]}];
 Modelica.Blocks.Interfaces.OutPort
  outPort_BrakePedal[{Final[n == 1]}];
 Equation[
  outPort_Gear.signal[[1]] == table.outPort.signal[[2]];
  outPort_Clutch.signal[[1]] == table.outPort.signal[[3]];
  PIDriver.inPort.signal[[1]] ==
   table.outPort.signal[[1]]
   ------------------------- -
              3.6
    inPort_ActualSpeed.signal[[1]];
  PIDriver.inPort_Reset.signal[[1]] ==
   table.outPort.signal[[3]];
  outPort_BrakePedal.signal[[1]] ==
   -Min[0, PIDriver.outPort.signal[[1]]];
  outPort_GasPedal.signal[[1]] ==
   -(Min[0, -PIDriver.outPort.signal[[1]]]) *
    table.outPort.signal[[3]];
 ]
]
```

```
Model[PI_Driver,
 "Resettable PI-regulator with limited output",
 Modelica.Blocks.Interfaces.InPort inPort[{Final[n == 1]}];
 Modelica.Blocks.Interfaces.InPort
  inPort_Reset[{Final[n == 1]}];
 Modelica.Blocks.Interfaces.OutPort outPort[{Final[n == 1]}];
 Modelica.Blocks.Nonlinear.Limiter limiter;
 ResetIntegrator resetIntegrator;
 Equation[
  Connect[inPort, resetIntegrator.inPort];
  Connect[inPort_Reset, resetIntegrator.inPort_Reset];
  limiter.inPort.signal[[1]] ==
   0.2 * (inPort.signal[[1]]) +
    0.5 * (resetIntegrator.outPort.signal[[1]]);
  Connect[limiter.outPort, outPort];
 ]
]
```

```
Model[ResetIntegrator, "Resettable Integrator",
 Modelica.Blocks.Interfaces.InPort inPort[{Final[n == 1]}];
 Modelica.Blocks.Interfaces.InPort
  inPort_Reset[{Final[n == 1]}];
 Modelica.Blocks.Interfaces.OutPort outPort[{Final[n == 1]}];
 Parameter Real outMax == 30;
 Parameter Real outMin == 0;
 Protected[
  Parameter Real p_outMax == outMax;
  Parameter Real p_outMin == outMin;
 ];
 Equation[
  (outPort.signal[[1]])' ==
   If[
    (outPort.signal[[1]] < p_outMin &&
       inPort.signal[[1]] < 0) ||
     (outPort.signal[[1]] > p_outMax &&
       inPort.signal[[1]] > 0), 0, inPort.signal[[1]]];
  When[inPort_Reset.signal[[1]] < Modelica.Constants.small,
   Reinit[outPort.signal[[1]], 0]]
 ]
]
```

The drivertable consists of information about the speed limits of the road and which gear and the clutch position the driver should have, all with respect to time. Below, as an example of a drivertable, the beginning of the used table can be seen

```
#1
double drivertab(76,4)
#1:st column = time [s], 2:nd column 2 = speed limit [km/h], 3:rd column =
gear,4:th column = clutch.
0          0  0   0
10.6       0  0   0
10.601     0  1   0
15        15  1   1
23        15  1   1
25        10  1   1
25.001    10  1   0
25.002    10  0   0
28         0  0   0
48.6       0  0   0
48.601     0  1   0
54        15  1   1
54.001    15  1   0
54.002    15  2   0
56        15  2   1
61        32  2   1
```

A.8 Driveline

```
Model[Driveline,
 Driver driver;
 Engine engine;
 Modelica.Mechanics.Rotational.Clutch
  Clutch[{mue_pos == {{0, 0.3}}, fn_max == 6973, peak == 1.1,
    cgeo == 0.176}];
 Gearbox gearbox;
 Modelica.Mechanics.Rotational.BearingFriction friction;
 Modelica.Mechanics.Rotational.SpringDamper
  DriveShaftLeft[{c == 2 * 10^4, d == 10^4}];
 Modelica.Mechanics.Rotational.SpringDamper
  DriveShaftRight[{c == 2 * 10^4, d == 10^4}];
 Modelica.Mechanics.Rotational.Brake
  brakeLeft[{mue_pos == {{0, 0.3}}, fn_max == 2.932 * 10^5,
    peak == 1.1, cgeo == 0.05}];
 Modelica.Mechanics.Rotational.Brake
  brakeRight[{mue_pos == {{0, 0.3}}, fn_max == 2.932 * 10^5,
    peak == 1.1, cgeo == 0.05}];
 FinalDrive finalDrive;
 Wheel wheelLeft;
 Wheel wheelRight;
 Chassis chassis;
 AirDrag airDrag;
 Equation[
  Connect[driver.outPort_GasPedal, engine.inPort_pedal];
  Connect[driver.outPort_BrakePedal, brakeLeft.inPort];
  Connect[driver.outPort_BrakePedal, brakeRight.inPort];
  Connect[ engine.flange_b, Clutch.flange_a];
  Connect[driver.outPort_Clutch, Clutch.inPort];
  Connect[Clutch.flange_b, gearbox.flange_a];
  Connect[gearbox.flange_b, friction.flange_a];
  Connect[friction.flange_b, finalDrive.flange_a];
  Connect[finalDrive.flange_left, DriveShaftLeft.flange_a];
  Connect[DriveShaftLeft.flange_b, brakeLeft.flange_a];
  Connect[brakeLeft.flange_b, wheelLeft.flange_a];
  Connect[wheelLeft.flange_b, chassis.flange_a];
  Connect[finalDrive.flange_right,
   DriveShaftRight.flange_a];
```

```
    Connect[DriveShaftRight.flange␣b, brakeRight.flange␣a];
    Connect[brakeRight.flange␣b, wheelRight.flange␣a];
    Connect[wheelRight.flange␣b, chassis.flange␣a];
    Connect[chassis.flange␣b, airDrag.flange␣a];
    Connect[driver.outPort␣Gear, gearbox.inPort];
    Connect[chassis.outPort, driver.inPort␣ActualSpeed];
   ]
  ]
```

Appendix B

Simulations

This appendix shows some additional plots from the simulations during a european driving cycle, described in section 9.2.

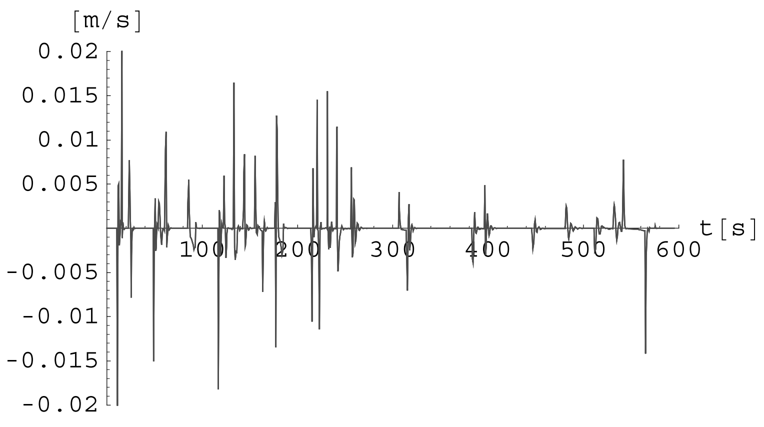


Figure B.1. Relative speed difference, $\omega_{d,in} - \omega_{d,out}$, for the drive shaft.

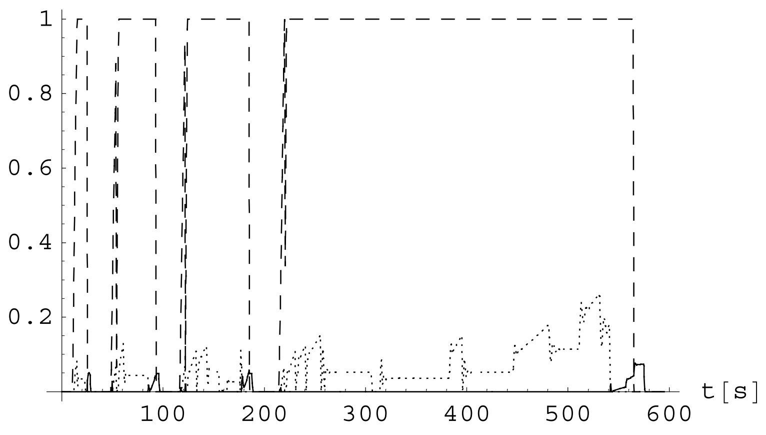


Figure B.2. Gas pedal (dotted), brake pedal (solid) and clutch pedal (dashed) signals.

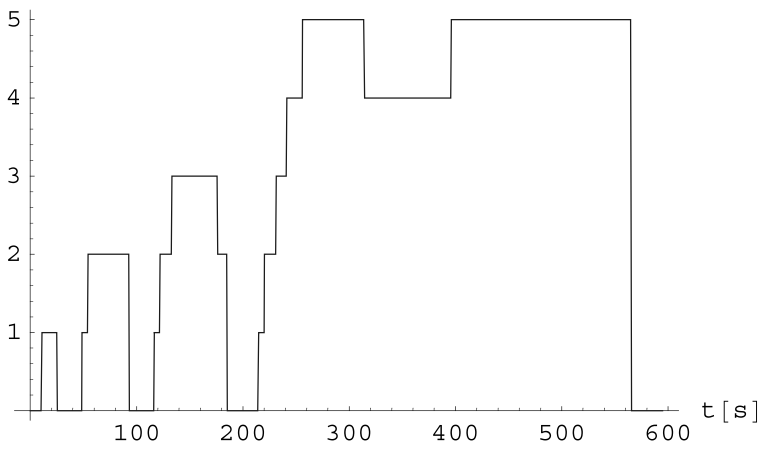


Figure B.3. Gear signal.

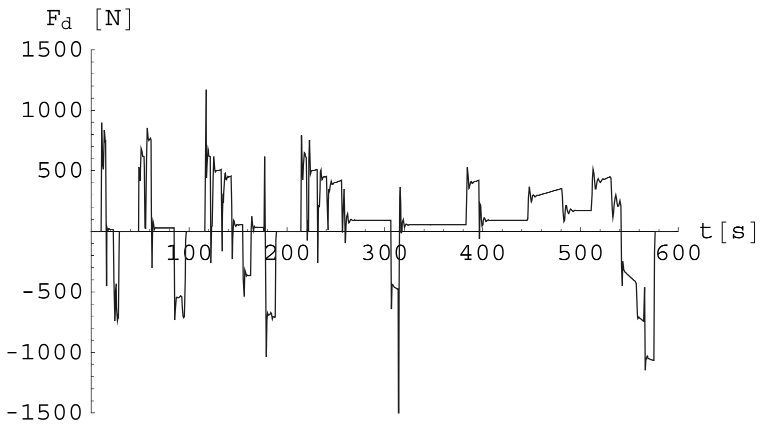


Figure B.4. Longitudinal driving force, F_d, from tyre-ground contact.

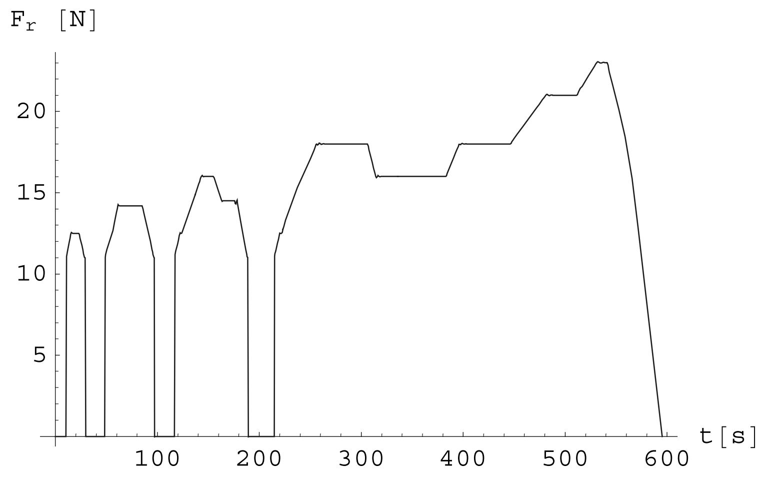


Figure B.5. Rolling resistance, F_r, acting on wheel.