Modeling a vehicle with use of partial vehicles and implementation in MATLAB/Simulink

Iwaarden, M.E. van

Modeling a Vehicle with Use of Partial Vehicles and Implementation in MATLAB/Simulink

A thesis submitted in partial fulfilment of the
requirements for the degree of
Master of Science at the

University of Leiden

by

Martijn van Iwaarden

Supervisor:

Prof. dr. S.M. Verduyn Lunel

Readers:

Prof. dr. ir. L.A. Peletier
Dr. J.A. van de Griend
Drs. W.F. Lammen (NLR)

This thesis is publiced as Memorandum IW-2003-025 of the National Aerospace Laboratory (NLR) in Amsterdam.

Department of Mathematics

Summary

The department Mathematical Models and Methods (IW) of the NLR is among other things occupied with simulation of vehicles. Typical examples of simulations are landing or taxiing aircrafts or personal cars. A vehicle is a typical example of a multibody system. The behaviour of a multibody system can be described by a multibody model with use of the appropriate equations of motion. These dynamical behaviour models are developed for use in training simulators.

The target of the research described in this report, is the development of a numerical stable behaviour model in MATLAB/Simulink, which describes the behaviour of a vehicle realistically. The model has to be applicable for different terrain types.

The vehicle is modeled with use of partial vehicles. One partial vehicle consists of a part of the chassis and the accompanying wheel. A partial vehicle can be modeled as a mass-spring system, consisting of three point masses and two springs connecting the point masses to each other. The equations of motion of a partial vehicle can be formulated by use of the Newton/Euler laws.

Several partial vehicles can be coupled to each other and the same they make up a vehicle. When coupling two partial vehicles to each other, one obtains a behaviour model of a motorcycle. When coupling to motorcycle to each other, one obtains a behaviour model of a four-part vehicle such as an automobile. In this way it is possible to analyse the behaviour of a multiple-wheeled vehicle. When modeling a vehicle with use of partial vehicles it is possible to extend existing behaviour models without completely reformulating the model.

The behaviour models of multiple-wheeled vehicles all have the same structure. Behaviour models with this structure are numerically stable solved by the Discrete Lagrange Multiplier Method. This numerical method, described in this report, is explicit and therefore appropriate for use in training simulators.

The simulation of wheel road contact is also a topic in this report. An algorithm that describes the wheel road contact is formulated. This algorithm is appropriate for different terrain types.

The model for a three dimensional four-part vehicle such as an automobile, is formulated. A two-part vehicle, such as a motorcycle, in two dimensions, is fully implemented in MATLAB/Simulink. The motorcycle can move in horizontal

and vertical direction. Steering is not possible. Several tests are performed to test the implemented model for realistic behaviour, and multiple simulations are done to show the usefulness of the model for designers of motorcycles.

Contents

Chapter 1

Introduction and Overview

1.1 Background Information and Goal of the Research

The National Aerospace Laboratory (NLR) states its mission in the following words: The mission of the foundation NLR is to provide expert contributions to activities in aerospace and related fields. NLR independently renders services to government departments and international agencies, aerospace industries and aircraft and spacecraft operators. Customers include various organizations based in the Netherlands, in Europe and elsewhere [2]. The NLR is an organization appointed to applications.

The department Mathematical Models and Methods (IW) is among other things occupied with modeling dynamic behaviour models for use in training simulators. This kind of models is implemented in software that calculates positions, velocities and orientation from signals as trottle, brake and steering. The calculated values can be passed to a visual system or motion platform. Possible applications of dynamic behaviour models are simulating trucks, personal cars and tractors, landing and taxiing aircraft, planetary vehicles and cranes.

The results of the research, described and discussed in this report, are built on earlier research. The reports [24],[9] and [27] contain results on formulating behaviour models for vehicles. In the reports [27],[15] and [8] special attention is given to simulate the wheel-road contact of vehicles. It was the task to combine all earlier results into one integrated behaviour model of a vehicle. Most of the research on physical items was done and some mathematical problems were risen. This gave rise to a mathematical contribution into the project. The final result of this research is a Simulink model implementation [1]. For the NLR it is possible to convert this Simulink model to model software for training simulators with the tool MOSAIC. MOSAIC is a conversion program, developed by the NLR, which can convert Simulink models to real-time simulation environments [21]. For the NLR, it is a matter of developing a new technique of modeling of multibody systems. In this approach of modeling it is easy to extend existing

behaviour models of vehicles. When from the research it appears that this technique is appropriate for modeling vehicles, it can be applied in simulation software and be used for various applications.

A vehicle is a typical multibody system, a system that consists of multiple bodies. The behaviour of multibody system can be described by the appropriate equations of motion. The resulting dynamical system can be implemented for training simulators. In formulating the behaviour model, main attention is paid to the vehicle suspension system. To use the behaviour model in training simulators it is necessary that the model can be solved in a stable and fast way. Therefore, explicit numerical methods, which can handle the model equations, are preferable. Several methods are described, discussed and tested. The most appropriate one is selected to use in the further development of the Simulink model.

The modeling of vehicles is done with use of partial vehicles. A partial vehicle is a part of a vehicle that consists of a tyre, a rim, and a part of the chassis, with appropriate shock absorber system. In the model all these parts of a partial vehicle are modeled by point masses. Partial vehicles can be connected to each other to obtain more complicated vehicles. This approach of developing behaviour models for vehicles leads to a way of modeling where it is easy to extend existing behaviour models, because of its modularity. When modeling vehicles by formulating the moment equations, extension to larger vehicles is much more complicated, because the whole ODE has to be completely reformulated. When using the approach of partial vehicles, the behaviour model for a vehicle consisting of 2 partial vehicles can easily be extended to a multiple-wheeld vehicle by adding extra equations.

Besides formulating the dynamical model, specific attention is paid to simulate wheel-road contact. After the more theoretical research, the implementation of a vehicle consisting of two partial vehicles, such as a motorcycle, is done.

1.2 Structure of this Report

The organisation of this report is as follows. First, in the remainder of the present chapter the final results of the research are presented. Each section contains the results of a part of the research. The whole research is divided into the following parts.

1. **Formulation of the behaviour model of a vehicle modeled with use of partial vehicles** The formulation of the behaviour model of a four-wheeled vehicle is done in Section 1.3. There the final behaviour model is presented. The final behaviour model is formulated on base of the results of earlier research. For more details on this subject is referred to Chapter 2.

2. **Stable numerical solving of the behaviour model of a vehicle** The method, which is used to solve the behaviour model is presented in

Section 1.4. The presented method was already used in earlier research. In Chapter 3 however, the method is compared with several other methods.

3. **Simulating wheel-road contact** Simulating wheel-road contact is done by an algorithm. This so-called *contact algorithm* is described in Section 1.5. A description of the development of this algorithm on base of earlier results is given in Chapter 4.

4. **Implementation of the behaviour model of a two-part vehicle in MATLAB/Simulink** A global description of the implementation of the behaviour model for a two-part vehicle in two dimensions, such as a motorcycle, is given in Section 1.6. The Simulink model is discussed in detail in Chapter 5. A manual to use the Simulink model is given in appendix A.

5. **Testing the implemented behaviour model** A single test of the implemented behaviour model is presented in Section 1.7. More tests are performed and discussed in Chapter 6.

6. **Using the behaviour model for design purposes** In Section 1.8 some remarks are made on using behaviour models for design purposes. An example of using the behaviour model for designers of vehicles is given in Chapter 7. There the behaviour model is used to find that setting of the different parameters that leads to most comfortable riding on a certain predefined terrain.

7. **Conclusions** Some global conclusions are given in Section 1.9. More conclusions and recommendations in detail are given in Chapter 8.

1.3 Modeling a Vehicle with use of Partial Vehicles

Modeling the undercarriage of a four-wheeled vehicle, like a personal car, can be realized with multibody dynamics. One can construct the vehicle as a combination of mass-spring systems with the upper parts connected to each other. The multibody model can be divided into four parts. Each part represents a partial vehicle. This is shown in Figure 1.1.

A partial vehicle is constructed by three point masses, which represent a part of the chassis, a wheel and the wheel contact point on an accompanying tyre. These three point masses are interrelated by two springs, which represent the shock absorber system and the elasticity of the tyre. Therefore a partial vehicle can be modeled as a mass-spring system with damping forces. The schematic representation of a mass-spring system is represented in Figure 1.2 [26, page 8].

As a basis for the description of the motion in the system, the Newton/Euler laws from classical mechanics are used [14, Chapter 6]. For each point mass in the mass-spring system, the equations of motion are formulated. To formulate the constraint motion of a point mass, the following ingredients are used

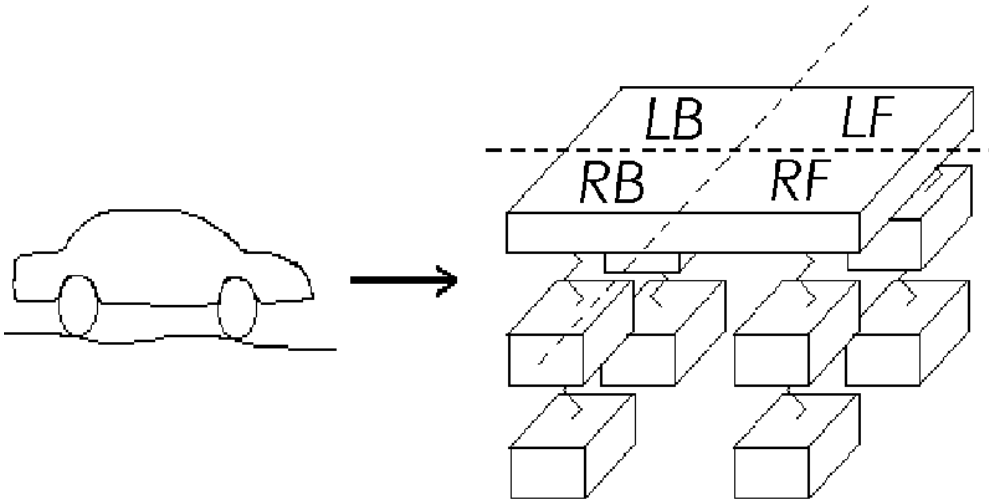

Figure 1.1: *A schematic representation of a personal car divided into four partial vehicles. The abbreviation RF means that it represents the right-front side of the car, RB the right-backside, LF the left-front side and LB the left-backside.*

- the second law of Newton, i.e. $F = ma$, or in words, force is mass times acceleration [14, page 72],
- the force in a spring is approximately given by a spring constant k times the deviation of the length of the spring in relaxed state. Let z_{rel} be the length of the spring in relaxed state and z_1 and z_2 the positions of the connection of the objects which are interrelated by the spring, then the force F which is caused by the spring is [20, page 256],

$$F = k(z_{rel} - (z_1 - z_2)). \tag{1.1}$$

- The force caused by the damping element is opposite to the force of the spring and depends on the velocity and a damping constant c in according to [23, page 15],

$$F = c(\dot{z}_1 - \dot{z}_2)), \tag{1.2}$$

  where $\dot{z}_i$ is the velocity of the object connected to the spring.

In the model, the following notations are used

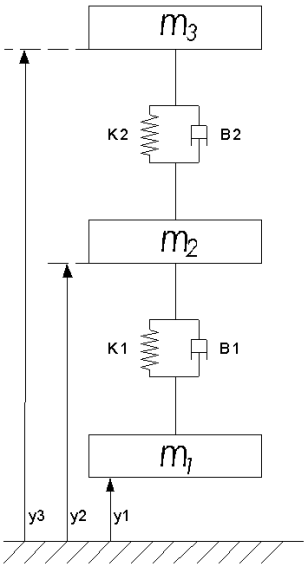

Figure 1.2: *The schematic representation of a mass-spring system with damping forces. K_i represents spring i, B_j represents damping element j and y_k represents the height of mass k.*

- m_i the mass of point mass i,
- x or $\mathbf{x}$ the state vector (the positions of all point masses),
- h the ground height,
- $F_c(t)$ the control input force, function of time,
- F_g gravitational force,
- k_{v_i} spring constant of spring i,
- d_{v_i} damping constant of spring i,
- $\delta_{min}, \delta_{max}$ resp the minimal and maximal spring length,
- $z_{rel_{vi}}$ relaxed spring length of spring i,
- g gravitational constant.

Before formulating the constraint motion of a partial vehicle in two dimensions, the angles α and β have to be defined. These angles are represented in Figure 1.3. With these ingredients the constraint motion of a partial vehicle can be

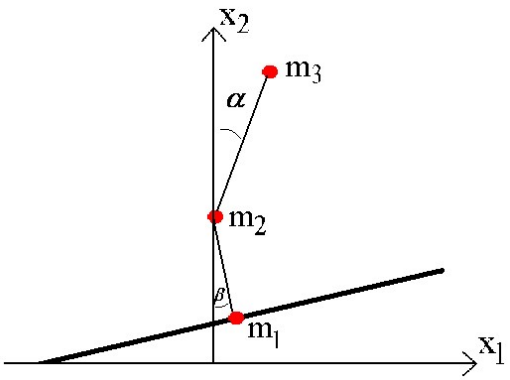

Figure 1.3: *The angles α and β for a partial vehicle in two dimensions.*

formulated. The constraint motion of a partial vehicle reads

$$\begin{aligned}
m_1\ddot{x}_1 &= -\sin(\alpha)(F_{v_1} - F_{d_1}) + F_{C_1} \\
m_1\ddot{x}_2 &= -\cos(\alpha)(F_{v_1} - F_{d_1}) - m_1 g + F_{C_2} \\
m_2\ddot{x}_3 &= \sin(\alpha)(F_{v_1} - F_{d_1}) - \sin(\beta)(F_{v_2} - F_{d_2}) + F_{C_3} \\
m_2\ddot{x}_4 &= \cos(\alpha)(F_{v_1} - F_{d_1}) - \cos(\beta)(F_{v_2} - F_{d_2}) - m_2 g + F_{C_4} \\
m_3\ddot{x}_5 &= \sin(\beta)(F_{v_2} - F_{d_2}) + F_{C_5} \\
m_3\ddot{x}_6 &= \cos(\beta)(F_{v_2} - F_{d_2}) - m_3 g + F_{C_6} \\
x_2 &\geq h(x_1),
\end{aligned} \tag{1.3}$$

where

- the two translational degrees of freedom for point mass m_i are denoted by $x_{(i-1)2+1}$ and $x_{(i-1)2+2}$;
- spring 1 is the spring damping element connecting point mass m_1 to point mass m_2 and spring 2 the spring damping element connecting point mass m_2 to point mass m_3;
- $d(m_i, m_j)$ is the distance between point mass i and point mass j, $i, j = 1, 2, 3$;
- $F_{v_i} = k_{vi}(z_{rel} - d(m_i, m_{i+1})$ is the spring force in spring i for $i = 1, 2$,
- and $F_{d_i} = d_{v_i}(\frac{\partial}{\partial t} d(m_i, m_{i+1})$ is the damping force in spring i for $i = 1, 2$.

One can construct a 'motorcycle model' by connecting two partial vehicle by a massless rod of length L. When looking closely to a motorcycle, one can see that the angle between the upper spring damping elements and the chassis keeps equal. Therefore, to simulate the behaviour of a motorcycle in a realistic way, this has to be implemented in the model.

To obtain the behaviour model for a motorcycle in two dimensions, one can take

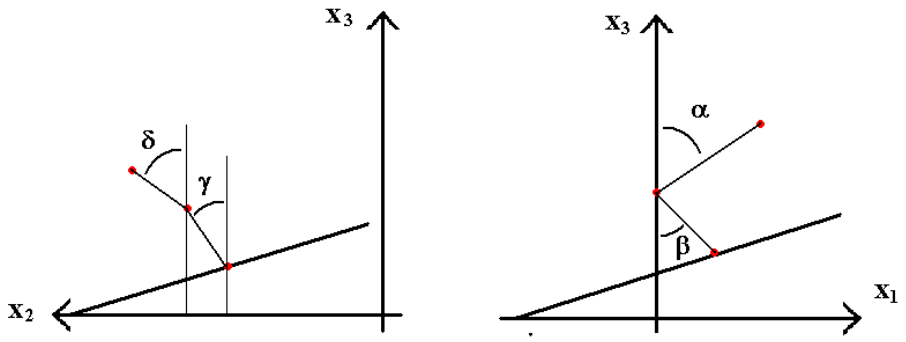

Figure 1.4: *The schematic representations of the (three dimensional) states of a partial vehicle which contains the angles α, β, γ and δ.*

two copies of system (1.3) extended with the constraints

$$\begin{aligned} 0 &= d(m_3, m_6) - L, && \text{(1.4a)} \\ 0 &= d(m_2, m_3)^2 + L^2 - d(m_2, m_6)^2 && \text{(1.4b)} \\ 0 &= d(m_5, m_6)^2 + L^2 - d(m_5, m_3)^2. && \text{(1.4c)} \end{aligned}$$

Constraint (1.4a) is the constraint needed to keep the distance L between the chassis parts of the two partial vehicles constant. The constraints (1.4b) and (1.4c) are set to keep the upper spring damping elements acting perpendicular to the chassis. These constraints follow from Pythagoras' theorem.

Before formulating the model of the mass-spring system extended for three dimensions, some definitions are needed. The three translational degrees of freedom for point mass m_i are denoted by $x_{(i-1)3+1}, x_{(i-1)3+2}$ and $x_{(i-1)3+3}$. The angles α, β, γ and δ, needed for the model, are defined in Figure 1.4. The equations of motion become

$$\begin{aligned} m_1\ddot{x}_1 &= \cos(\gamma)(-\sin(\alpha)(F_{v_1} - F_{d_1})) + F_{C_1} \\ m_1\ddot{x}_2 &= \sin(\gamma)(-\sin(\alpha)(F_{v_1} - F_{d_1}) - \cos(\alpha)(F_{v_1} - F_{d_1})) + F_{C_2} \\ m_1\ddot{x}_3 &= \cos(\gamma)(-\cos(\alpha)(F_{v_1} - F_{d_1})) - m_1 g + F_{C_3} \\ m_2\ddot{x}_4 &= \cos(\gamma)(\sin(\alpha)(F_{v_1} - F_{d_1})) + F_{C_4} \\ m_2\ddot{x}_5 &= \sin(\gamma)(\sin(\alpha)(F_{v_1} - F_{d_1}) + \cos(\alpha)(F_{v_1} - F_{d_1})) \\ &\quad - \sin(\delta)(\sin(\beta)(F_{v_2} - F_{d_2}) + \cos(\beta)(F_{v_2} - F_{d_2})) + F_{C_5} \\ m_2\ddot{x}_6 &= \cos(\gamma)(\cos(\alpha)(F_{v_1} - F_{d_1})) - m_2 g + F_{C_6} \\ m_3\ddot{x}_7 &= \cos(\delta)(\sin(\beta)(F_{v_2} - F_{d_2})) + F_{C_7} \\ m_3\ddot{x}_8 &= \sin(\delta)(\sin(\beta)(F_{v_2} - F_{d_2}) + \cos(\beta)(F_{v_2} - F_{d_2})) + F_{C_8} \\ m_3\ddot{x}_9 &= \cos(\delta)(\cos(\beta)(F_{v_2} - F_{d_2})) - m_3 g + F_{C_9} \end{aligned} \qquad (1.5)$$

The motorcycle model with three translational degrees of freedom is give by two copies of (1.5) and the three constraints (1.4).

For a four-wheeled vehicle, such as a personal car, two copies of the motorcycle model and some extra constraints are needed. Figure 1.5 represents this schematically. The constraints needed for the behaviour model of a personal

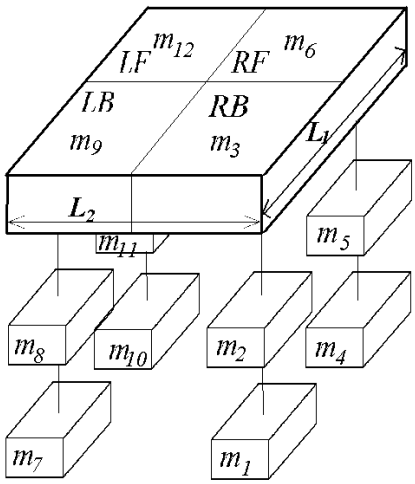

Figure 1.5: *The schematic representation of a four-wheeled vehicle, with numbered masses and definitions of L_1 and L_2.*

car are

$$\begin{aligned}
0 &= d(m_3, m_6) - L_1, \\
0 &= d(m_2, m_3)^2 + L_1^2 - d(m_2, m_6)^2, \\
0 &= d(m_5, m_6)^2 + L_1^2 - d(m_5, m_3)^2, \\
0 &= d(m_9, m_{12}) - L_1, \\
0 &= d(m_8, m_{11})^2 + L_1^2 - d(m_8, m_{12})^2, \\
0 &= d(m_{11}, m_{12})^2 + L_1^2 - d(m_{11}, m_9)^2, \\
0 &= d(m_2, m_3)^2 + L_2^2 - d(m_2, m_9)^2, \\
0 &= d(m_8, m_9)^2 + L_2^2 - d(m_8, m_3)^2, \\
0 &= d(m_5, m_6)^2 + L_2^2 - d(m_5, m_12)^2, \\
0 &= d(m_{11}, m_{12})^2 + L_2^2 - d(m_{11}, m_6)^2, \\
0 &= d(m_3, m_6)^2 + d(m_6, m_{12})^2 - d(m_3, m_{12})^2.
\end{aligned} \tag{1.6}$$

The constraints (1.6) are the equality constraints. However, there are some natural inequality constraints. There are two kinds of inequality constraints. The following constraints imply that the wheel contact points are always above or on the terrain and that the rims are always above the wheel contact points

$$\begin{cases}
x_6 > x_3 &\geq h(x_1, x_2), \\
x_{15} > x_{12} &\geq h(x_{10}, x_{11}), \\
x_{24} > x_{21} &\geq h(x_{19}, x_{20}), \\
x_{33} > x_{30} &\geq h(x_{28}, x_{29}).
\end{cases} \tag{1.7}$$

The next set of inequality constraints imply that the point mass representing

the chassis part is always above the rim

$$\begin{cases} x_6 & < & x_9, \\ x_{15} & < & x_{18}, \\ x_{24} & < & x_{27}, \\ x_{33} & < & x_{36}. \end{cases} \tag{1.8}$$

The inequality constraints (1.7) are implemented in the contact algorithm, in which the wheel-road contact is modeled. Satisfying the constraints (1.8) is just a case of choosing the right spring and damping constants.

As one can see, it is easy to extend the model with more partial vehicles. Adding one partial vehicle means adding of the equations of motion (1.5) and the appropriate inequality constraints. The development of the model is further discussed in Chapter 2.

In Section 1.4 numerical solving of systems such as system (1.5) subject to equality constraints like the constraints (1.6) is described. Section 1.5 describes the simulation of wheel-road contact, where the inequality constraints (1.7) are satisfied with the use of an algorithm.

1.4 Stable Numerical Solving of the Behaviour Model

Because the behaviour model, consisting of the equations (1.5), (1.6), (1.7) and (1.8), has to be implemented in training simulators there is need for an efficient numerical method to solve the behaviour model in time. Therefore explicit numerical methods are preferable. The system of equations that has to be solved for the behaviour model of a vehicle consist of second-order ordinary differential equations with equality constraints. These systems are a special class of differential-algebraic equations (DAEs).

A Differential-Algebraic Equation (DAE) is a differential equation

$$\mathbf{F}(t, \mathbf{y}, \dot{\mathbf{y}}) = 0, \tag{1.9}$$

where the Jacobian matrix $\partial \mathbf{F}/\partial \dot{\mathbf{y}}$ is singular. In this definition $\mathbf{F} : [0, \infty] \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$, $\mathbf{y}$ is a time dependent vector function in $\mathbb{R}^n$, $\dot{\mathbf{y}}$ is the time derivative of $\mathbf{y}$.

The next DAE system can be considered as an extension of an explicit Ordinary Differential Equation (ODE). It is an *ODE with constraints* or a *semi-explicit* system of differential-algebraic equations

$$\dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x}, \mathbf{z}), \tag{1.10a}$$

$$\mathbf{0} = \mathbf{g}(t, \mathbf{x}, \mathbf{z}). \tag{1.10b}$$

One can reformulate (1.10) into the form (1.9) for the unknown vector $\mathbf{y} = \binom{\mathbf{x}}{\mathbf{z}}$ with the nonsingular Jacobian matrix

$$\frac{\partial \mathbf{F}(t, \mathbf{u}, \mathbf{v})}{\partial \mathbf{v}} = \begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix}.$$

If in system (1.10) the matrix $\partial \mathbf{g}/\partial \mathbf{z}$ is nonsingular then, by the implicit function theorem, $\dot{\mathbf{z}}$ is obtained by one differentation of (1.10b). If this is done the DAE is transformed to an explicit ODE system for all the unknowns. In general the number of differentiations needed to obtain an explicit ODE from a DAE is called the *index* of a DAE. A more formal definition of the *index* is given by [7, page 236] and listed here.

Definition
For general DAE systems (1.9), the *index* along a solution $\mathbf{y}(t)$ is the minimum number of differentiations of the system which would be required to solve for $\dot{\mathbf{y}}$ uniquely in terms of $\mathbf{y}$ and t (i.e. to define an ODE for $\mathbf{y}$). Thus, the index is defined in terms of the overdetermined system

$$\left\{\begin{array}{rcl} \mathbf{F}(t,\mathbf{y},\dot{\mathbf{y}}) &=& 0 \\ \frac{d\mathbf{F}}{dt}(t,\mathbf{y},\dot{\mathbf{y}},\ddot{\mathbf{y}}) &=& 0 \\ & \vdots & \\ \frac{d^p\mathbf{F}}{dt^p}(t,\mathbf{y},\dot{\mathbf{y}},\ldots,\mathbf{y}^{(p+1)}) &=& 0 \end{array}\right. \tag{1.11}$$

to be the smallest integer p so that $\dot{\mathbf{y}}$ in (1.11) can be solved for in terms of $\mathbf{y}$ and t .

In the present literature with respect to numerical integration DAEs are characterized by their index. The index can be viewed upon as a measure of how far a DAE is from being an ODE [6, page 315].

To illustrate the definition of the index the following DAE is considered

$$\begin{aligned} x_1' &= y_1, && \text{(1.12a)}\\ x_2' &= y_2, && \text{(1.12b)}\\ y_1' &= -\lambda x_1, && \text{(1.12c)}\\ y_2' &= -\lambda x_2 - g, && \text{(1.12d)}\\ 0 &= x_1^2 + x_2^2 - 1. && \text{(1.12e)} \end{aligned}$$

In this system $\lambda = \lambda(t)$ is an unknown function and g is a known constant. When differentiating the constraint (1.12e) the equation

$$x_1 x_1' + x_2 x_2' = 0$$

is obtained. Substituting for x_1' from (1.12a) and x_2' from (1.12b) the equation

$$x_1 y_1 + x_2 y_2 = 0 \tag{1.13}$$

is obtained. Differentiating (1.13) and again substituting for x_1' and x_2' yields

$$x_1 y_1' + x_2 y_2' + y_1^2 + y_2^2 = 0.$$

Substituting for y_1' from (1.12c) and y_2' from (1.12d) and simplifying using (1.12e) yields

$$-\lambda - x_2 g + y_1^2 + y_2^2 = 0. \tag{1.14}$$

This yields λ, which can be substituted into (1.12c) and (1.12d) to obtain an ODE for $\mathbf{x}$ and $\mathbf{y}$. To obtain an ODE for *all* the unknowns, it is needed to

differentiate (1.14) one more time. Then an ODE for λ is obtained. In the processing of getting to the explicit ODE system, the position constraints were differentiated three times. Hence, the index of this system is 3 [7, page 241,242].

Modeling mechanical systems is usually done by formulating the equations of motion. The second law of Newton, force is mass times acceleration, plays an important role in this formulation. The general form of the model of an unconstrained mechanical system is therefore [23, page 14]

$$M\ddot{\mathbf{x}} = F(t, \mathbf{x}, \dot{\mathbf{x}}) \tag{1.15}$$

With M a positive definite square mass matrix and $F(t, \mathbf{x}, \dot{\mathbf{x}})$ the applied forces. Time is denoted by t and $\mathbf{x}$, $\dot{\mathbf{x}}$ and $\ddot{\mathbf{x}}$ resp denote position, velocity and acceleration.

The general form of the equations of motion of a constrained mechanical system is given by [23, page 18]

$$\begin{aligned} M\ddot{\mathbf{x}} &= F(t, \mathbf{x}, \dot{\mathbf{x}}) + F_r(\mathbf{x}, \lambda) && \text{(1.16a)} \\ 0 &= P(\mathbf{x}, t), && \text{(1.16b)} \end{aligned}$$

where $P(\mathbf{x}, t)$ is a vector valued function describing the constraints; F_r describes additional forces acting on the system. These so-called *generalized constraint forces* are responsible for the constraint to be satisfied. The constraint defines a manifold of free motion. By basic principles, like *d'Alembert's principle* ([20, Chapter 12] and [25, page 91-97]), it can be shown that constraint forces are orthogonal to this manifold [25, page 92]. This leads to

$$F_r(\mathbf{x}, \lambda) = C(\mathbf{x})^T \lambda$$

with the *constraint matrix* $C := \frac{d}{d\mathbf{x}} P(\mathbf{x}, t)$ and λ the vector with the so-called *Lagrange multipliers*. When considering the term $F(t, \mathbf{x}, \dot{\mathbf{x}})$ in (1.15), one can divide this term into a part $B(t, \mathbf{x}, \dot{\mathbf{x}})$, representing the Coriolis, gravitational and centrifugal force/torque vector and a part $F_c(t)$ representing the control forces as trottle, brake and steering. The resulting equations of motion of a constrained mechanical system are now given by

$$M\ddot{\mathbf{x}} = B + F_c + C^T \lambda. \tag{1.17}$$

This formulation is called the classical formulation of constrained mechanical systems. Note that the original DAE (1.16) is transformed into an ODE that satisfies the constraint.

With the substitution $\mathbf{y} = \dot{\mathbf{x}}$, the constrained mechanical system (1.16) can be transformed to the *dynamical system*

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{y}, && \text{(1.18a)} \\ M\dot{\mathbf{y}} &= B + F_c, && \text{(1.18b)} \\ 0 &= P(\mathbf{x}), && \text{(1.18c)} \\ 0 &= C\mathbf{y} - d, && \text{(1.18d)} \end{aligned}$$

where $d = \frac{\partial}{\partial t} P(\mathbf{x}, t) - C\mathbf{y}$. Constraint (1.18d) is the time derivative of constraint (1.18c).

The following assumptions are made

Assumption A: the matrix M is positive definite.

Assumption B: $\forall(\mathbf{x},t) \in \mathbb{R}^{n+1}$ such that $P(\mathbf{x},t) = 0 : \text{rank}(C(\mathbf{x},t))) = m, m \leq n$.

Assumption B': $(\exists\varepsilon > 0)$ such that $\forall\delta$ with $|P(\mathbf{x},t)| \leq \delta \Rightarrow \text{rank}(C(\mathbf{x},t)) = m, m \leq n$.

Assumption C: let R denote the stability region of the numerical method under consideration. Let $\sigma(.)$ denote the collection of eigenvalues of the controlled system. Then $\forall\nu \in \sigma(.)$, choose Δt such that $\nu\Delta t \in R$.

Assumption D: the time step Δt is such that the variation in the constraint Jacobian matrix C is small on the interval $[t_n, t_{n+1}]$ for all n.

The manifold S is defined as

$$S = \{(\mathbf{x},\mathbf{y},t) \in \mathbb{R}^{2n+1} | P(\mathbf{x},t) = \mathbf{0} \quad \text{and} \quad C(\mathbf{x},t)\mathbf{y} = d(\mathbf{x},t)\}$$

By the theorem about the formulation of constraint mechanical systems [5, Theorem 4.1] , under assumptions A and B, the following formulations are equivalent

i $\mathbf{x}(t)$ is a trajectory of the dynamical system (1.18), with $(\mathbf{x}(t_0),\mathbf{y}(t_0)) = (x_0, y_0)$,

ii $\exists\mu_X \in \mathbb{R}^m$ and $\exists\lambda_Y \in \mathbb{R}^m$ such that $\mathbf{x}(t)$ is a trajectory of the dynamical system

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{y} + XC^T\mu_X, && \text{(1.19a)} \\ M\dot{\mathbf{y}} &= B + F_c + MYC^T\lambda_Y, && \text{(1.19b)} \\ 0 &= P(\mathbf{x}), && \text{(1.19c)} \\ 0 &= C\mathbf{y} - d, && \text{(1.19d)} \end{aligned}$$

with $(\mathbf{x}(t_0),\mathbf{y}(t_0)) = (x_0, y_0)$, X and Y are matrices such that $\forall(\mathbf{x},\mathbf{y},t) \in S, rank(CXC^T) = rank(C) = rank(CYC^T)$ and $\mathbf{x}_0$, $\mathbf{y}_0$ and t_0 are vectors such that $(\mathbf{x}_0,\mathbf{y}_0,t) \in S$.

By the theorem about stable numerical integration of constrained mechanical systems [5, Theorem 4.2] stable numerical integration of (1.19) is obtained for the classical fourth-order Runge Kutta method, when

$$\mu_X^d = -(CXC^T)^{-1}(C\mathbf{y} - d + P_n/\Delta t),$$

$$\lambda_Y^d = -(CYC^T)^{-1}(CM^{-1}(B + F_c) + \dot{C}\mathbf{y} - \dot{d} + (C_n y_n - d_n)/\Delta t).$$

This method is called the Discrete Lagrange Multiplier Method (DLMM). Just as in [6] and [5], we set $X = I$ and $Y = M^{-1}$. The justification to choose this method among the other DAE-solvers is given in Chapter 3. In Chapter 3 several methods are compared and tested.

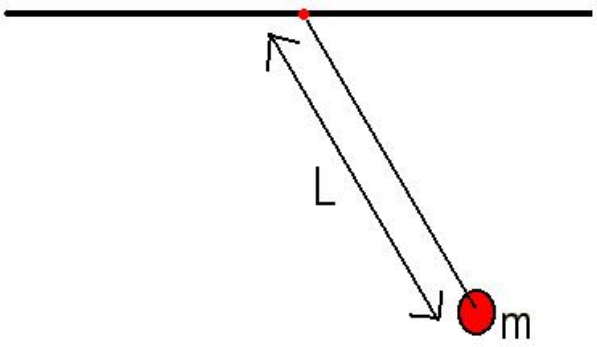

Figure 1.6: *The planar pendulum*

To illustrate the stable solving of the DLMM the model of a planar pendulum is considered. A planar pendulum as in Figure 1.6 is modeled as a falling point mass subject to a superimposed constraint.

The only external force on the system is the gravitational force. Therefore the equations of motion of this mechanical system are given by

$$\begin{aligned} m\ddot{x}_1 &= 0, && \text{(1.20a)} \\ m\ddot{x}_2 &= -mg, && \text{(1.20b)} \\ 0 &= x_1^2 + x_2^2 - L^2 = P(\mathbf{x}), && \text{(1.20c)} \end{aligned}$$

where m is the mass of the point mass, L is the length of the rod, and g is the gravitational constant. The state-vector $\mathbf{x} = (x_1, x_2)$ represents the Cartesian coordinates of the point mass.

The system (1.20) is solved with the DLMM and in Figure 1.7 the found trajectory is plotted. In Figure 1.8 the deviation of the constraint (1.20c) is plotted and as one can see in this figure, the deviation of the constraint keeps very small and is not enhancing.

With these results it is possible to solve the behaviour model of a vehicle in a stable way, with an explicit numerical method. On this point, the first two parts of the research as described in Section 1.2 are done. The next section proceeds with the simulation of wheel-road contact.

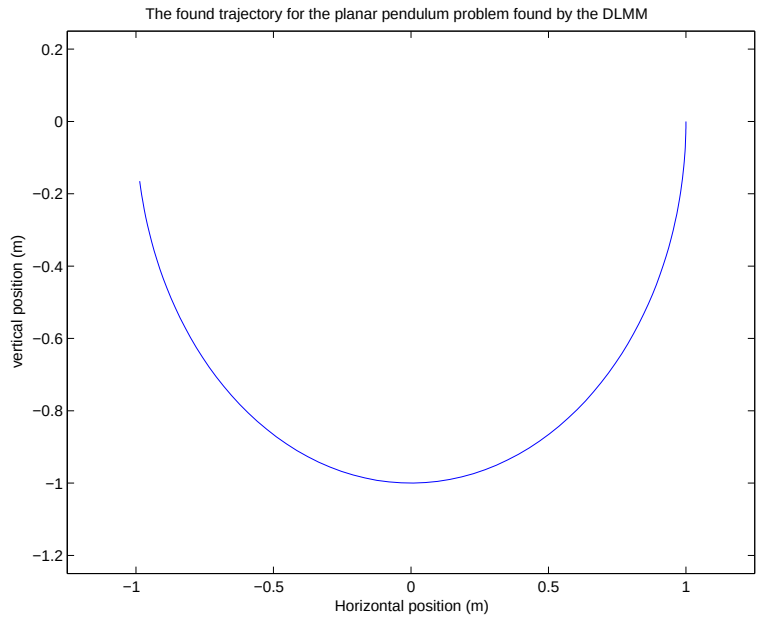

Figure 1.7: *The found trajectory for the planar pendulum model (1.20). The midpoint of the circle is (0,0) The mass m starts form rest on (1,0). Simulation time is 1 sec.*

1.5 Simulating Wheel-Road Contact

Simulating wheel-road contact means satisfying the constraints (1.7). The modeling of the wheel-road contact is done by an algorithm. This algorithm models the behaviour of the wheel subject to a given terrain. This algorithm is called *the contact algorithm*. Next the algorithm is described for one partial vehicle in two dimensions. The partial vehicle can move horizontal and vertical.

The equations of motion are solved subject to the equality constraints. After each time-step of the DAE-solver the mode of the wheel subject to the terrain has to be detected. Three different modes can be distinguished.

1. If on time t_n the whole wheel is above the terrain the wheel contact point is also above the terrain. The constraints are then satisfied. Note that, when the whole wheel is above the terrain it is not known where the wheel contact point is. Because each point on the tyre can become the wheel contact point. It all depends on the terrain.

2. If on time t_{n-1} the wheel contact point is on the terrain and on time t_n the wheel contact point is under the terrain, the mode *rolling* is detected. The inequality constraints (1.7) are not satisfied and the algorithm has to do this.

3. If on time t_{n-1} the whole wheel is above the terrain and on time t_n it is

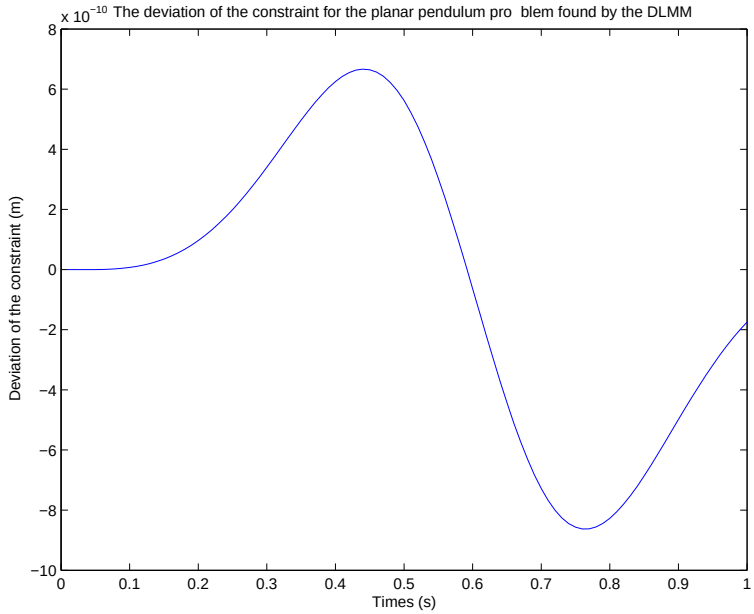

Figure 1.8: *The deviation of the constraint for the planar pendulum model (1.20). This figure belongs to Figure 1.7*

not, an *impact* has occurred and the constraints has to be satisfied. This has to be done by the contact algorithm.

Detecting the mode is the first task of the contact algorithm. To detect the mode, it is not possible to verify the vertical position of the wheel contact point and the height of the terrain on that position. This is because it is unknown where the wheel contact point is. However, it is possible to compute an approximation of the radius of the tyre. This is the distance from the rim to the foregoing wheel contact point. The foregoing wheel contact point moves due to the spring forces in the tyre. The whole wheel can be seen as a circle with midpoint the positions of the rim and radius the distance from the rim to the foregoing wheel contact point. This can be represented in a formula, which represents the whole wheel. Let (x_0, y_0) represent the horizontal and vertical position of the rim and d the distance between the rim and the foregoing wheel contact point. The formula then reads

$$(x - x_0)^2 + (y - y_0)^2 - d^2 = 0. \tag{1.21}$$

If the equation representing the terrain is known, it is possible to detect the mode analytically. Because usually the terrain equation is not known, it is local approximated linearly by $y = ax + b$. To detect the mode, the linear approximation of the terrain is substituted in (1.21). The resulting equation becomes

$$(x - x_0)^2 + (ax + b - y_0)^2 - d^2 = 0. \tag{1.22}$$

This is a polynomial of second degree that can be solved analytically. When

using the ABC-formula, the discriminant D becomes

$$D = (-2x_0 + 2a(b - y_0))^2 - 4(1 + a^2)(x_0^2 + (b - y_0)^2 - d^2).$$

This information can be used to detect the mode. If on time t_n, one has $D < 0$, then there is nothing to change. The whole wheel is above the terrain. If on time t_{n-1} the wheel contact point is on the terrain and on time t_n one has $D \geq 0$, then the mode rolling is detected, and if on time t_{n-1} the wheel contact point is above the terrain and on time t_n one has $D \geq 0$, an impact is detected.

If one of the modes *rolling* or *impact* is detected, the inequality constraints (1.7) are not satisfied and the contact algorithm has to change the state such that wheel-road contact is simulated in a realistic way and the inequality constraints become satisfied. Section 1.5.1 contains a description of the computations of the contact algorithm if the mode is rolling. Section 1.5.2 contains a description of the computations needed for simulating an impact.

1.5.1 Simulating a Rolling Wheel

When the wheel is rolling the wheel contact point has to be found. The wheel contact point is located on the terrain and is located such that the spring in the tyre acts perpendicular to the terrain. When the linear approximation of the terrain under the wheel is $y = ax + b$, the line perpendicular to the terrain and through the rim (x_0, y_0) can be formulated. This line is then given by the formula $y = 1/ax + c$, where $c = (y_0 + \frac{x_0}{a})$. The intersection of these two lines is set as the wheel contact point. Therefore the horizontal position of the wheel contact point is set to $\frac{c-b}{a-1/a}$ if $a \neq 0$. If $a = 0$ then the horizontal position of the wheel contact point is the same as the horizontal position of the rim x_0. The vertical position of the wheel contact point is now given by the linear approximation of the terrain $y = ax + b$. The velocities of the wheel contact point are determined to be equal to the velocities of the rim. After these computations, the inequality constraints (1.7) are satisfied and a rolling wheel is simulated in a realistic way.

1.5.2 Simulating an Impact

If the mode impact is detected, the following sequence of steps has to be done to simulate this wheel-road contact

Step 1, compute impact time t^* - To know when exactly the impact was, the terrain height is linearly approximated in time on the interval $[t_{n-1}, t_n]$. The vertical position of the future wheel contact point is also linear approximated in time. On the intersection of these two lines, the impact time t^* is found.

Step 2, compute impact position - From the impact time t^* one can compute the horizontal and vertical position of the wheel contact point. The

horizontal position is approximately given by

$$x(t^*) = x(t_{n-1}) + \dot{x}(t_{n-1})(t^* - t_{n-1}) \tag{1.23}$$

and the vertical impact position is given by the linear approximation of the terrain $y = ax + b$.

Step 3, compute velocities just before impact - The velocities just before the impact can be computed by the following formula

$$y_{i_{bc}} = y_i(t_{n-1}) + \dot{y}_i(t^* - t_{n-1}), i = 1, 2. \tag{1.24}$$

In this formula $y_{i_{bc}}$ is the computed velocity of point mass m_1 just before impact, y_i is the velocity of point mass m_1 and $\dot{y}_i$ is the acceleration of point mass m_i. This formula can be applied to both horizontal ($i = 1$) and vertical ($i = 2$) velocity of the appropriate point mass.

Step 4, compute velocities just after impact - When an impact occurs, the impact in reality will not be a completely elastic impact or a completely plastically impact. The component of the velocity perpendicular to the terrain would be changed to a velocity in opposite direction, but not with the same value. Because it is not expected that all the kinetic energy will be maintained elasticity constants e_{vert} and e_{hor} are introduced. The elasticity constant $e_{vert} \in [0, 1]$ determines which part of the velocity perpendicular to the terrain will be maintained. The elasticity constant $e_{hor} \in [0, 1]$ determines which part of the velocity parallel to the terrain will be maintained.

To apply this elasticity constants to the velocities of the wheel contact point just before impact, the factorisation of the velocities parallel with the terrain and perpendicular to the terrain has to be performed. The following steps can do this factorisation. The angle β is the angle of the terrain, computed as $\beta = arctan(a)$ where a is the tangent of the linear approximation of the terrain. The vertical and horizontal velocities in the original orientation are denoted by y_{vert} and y_{hor}. The vertical and horizontal velocities in the new orientation are denoted by yp_{vert} and yp_{hor}.

$$\begin{aligned} nr_1 &= y_{hor}\cos(\beta) \\ nr_2 &= y_{vert}\sin(\beta) \\ nr_3 &= -y_{hor}\sin(\beta) \\ nr_4 &= y_{vert}\cos(\beta) \\ yp_{vert} &= nr_1 + nr_2 \\ yp_{hor} &= nr_3 + nr_4 \end{aligned}$$

Applying the elasticity constants gives for the velocities just after the impact

$$\begin{aligned} yp_{vert} &= -e_{vert} \quad yp_{vert} \\ yp_{hor} &= e_{hor} yp_{hor} \end{aligned}$$

The factorisation back to the original orientation can be done by the fol-

lowing steps

$$\begin{aligned} nr_5 &= yp_{hor}\cos(\beta) \\ nr_6 &= -yp_{vert}\sin(\beta) \\ nr_7 &= yp_{hor}\sin(\beta) \\ nr_8 &= yp_{vert}\cos(\beta) \\ y_{hor_{ac}} &= nr_5 + nr_6 \\ y_{vert_{ac}} &= nr_7 + nr_8 \end{aligned}$$

The resulting $y_{hor_{ac}}$ and $y_{vert_{ac}}$ are the horizontal and vertical velocities of the wheel contact point just after the impact.

Step 5, compute velocities atn time t_n - The velocities at time t_n can be computed by the formula

$$y_i(t_n) = y_{i_{ac}} + \dot{y}_i(t_n - t^*), i = 1, 2. \tag{1.25}$$

For a free falling body the term $\dot{y}_i$ is usually $-g$ for the vertical velocity ($i = 2$), where g is the gravitational constant ($g = 9.81$).

Step 6, compute positions on time t_n - The positions of the wheel contact point on time t_n can be computed by the formula

$$x_i(t_n) = x_i(t^*) + y_{i_{ac}}(t_n - t^*) + \dot{y}_i(t_n - t^*)^2, i = 1, 2. \tag{1.26}$$

Step 7, check for second impact in the same interval - It is possible that after correcting the velocities and positions the wheel contact point is not above the terrain, but due to the gravitational forces again under the terrain. Therefore after correcting the positions and velocities is checked if the corrected positions of the wheel contact point satisfy the inequality constraints (1.7). If not, it is stated that the wheel is going to roll and the wheel contact point is set on the terrain. The position and velocities are set exactly the same as described in Section 1.5.1.

To set the wheel rolling in this case is realistic, because then the distance between the wheel contact point and the terrain must be small. An upper bound of the distance between the wheel contact point and the terrain is given by $\frac{1}{2}g\Delta t^2$.

After the execution of the contact algorithm the inequality constraints (1.7) are satisfied. The development of this algorithm is further discussed in Chapter 4.

All ingredients needed for the implementation in MATLAB/Simulink are present on this point of the research. Therefore, the implementation of a vehicle can be done. In the next section the implementation of a two-part vehicle in two dimensions in a Simulink model is described.

1.6 Implementation of the Model

The behaviour model of a two-part vehicle in two dimensions is implemented in a Simulink model. The top level of the Simulink model is represented in Figure 1.9. In the appendix a manual is given to handle and change the model.

1.6.1 The MATLAB Compiler and Simulink S-functions

First some basic information about MATLAB, Simulink and the MATLAB compiler is given [1].

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation.

Simulink has become the most widely used software package in academia and industry for modeling and simulating dynamic systems. In Simulink one can easily build models from scratch, or take an existing model and add to it. Simulations are interactive, therefore it is possible to change parameters on the fly and immediately see what happens. One has instant access to all the analysis tools in MATLAB, therefore the results can taken, analyzed and visualized. A Simulink model is built with blocks. Many blocks are predefined, but it is also possible to build blocks, for instance by the use of Simulink S-functions, which is done. A Simulink S-function is a computer language description of a Simulink block. S-functions can be written in MATLAB, C, C++, Ada, or Fortran. In the implementation of the behaviour model of the vehicle, the S-function are written in C.

The MATLAB Compiler takes M-files as input and generates C or C++ source code or P-code as output. The MATLAB Compiler can generate various kinds of source code for instance C code S-functions for use with Simulink and C shared libraries (dynamically linked libraries, or DLLs, on Microsoft Windows) and C++ static libraries.

With these tools it is possible to program all in M-files and use the MATLAB Compiler to convert it to Simulink C Mex S functions. This is done because a Simulink model is wanted and the C mex S functions can be invoked in Simulink. This Simulink model can be converted for real-time simulating purposes by an especially developed tool of the NLR, called MOSAIC [21].

1.6.2 The Simulink Model

The Simulink model is represented in Figure 1.9. The main parts of the model are the four colored blocks.

The yellow block contains the subsystem where the forces acting on the motorcycle are computed, the red one contains the subsystem where the needed terrain information is computed and the green block represents the subsystem where the contact algorithm is applied to the computed state vector.

The integrator block can be seen as the center of the model. There, on base of the output of the subsystem "Equations of motion and DLMM", the next state-vector is computed by the specified numerical method. The model starts with the initial condition given by the IC-block right under. The state for the following time-step is computed in the integrator block. The state port copies the computed state and sends it to the subsystems "Terrain information" and

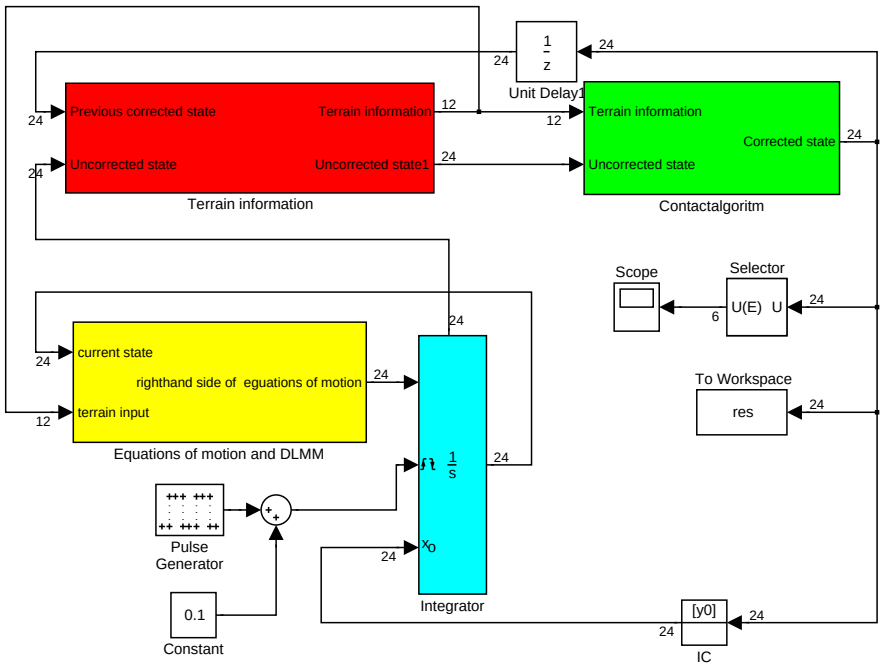

Figure 1.9: *The top-level of the Simulink model of the vehicle.*

"Contact". The contact algorithm corrects the state, subject to the terrain information obtained from the red subsystem. The corrected state is sent to the initial condition port of the integrator block. The Pulse Generator and the Constant block are designed to reset the state of the integrator block each time-step to the corrected state. Then the next time-step can start.

The Unit Delay block in the model is set to avoid a loop. The initial condition of this block is the same as the initial condition of the whole system. Right under in the model a Selector block, a scope block and a block, called To Workspace, are found. The Selector selects the vertical positions computed by the model and represents them in the scope. All output needed for application in training simulators can be obtained from the complete set of data that is sent to the workspace in an array. Changing parameters can easily be done by changing them in the masks of the subsystems.

More information about the Simulink model in detail is given in Chapter 5. The behaviour model of a two-part vehicle in two dimensions is implemented in MATLAB/Simulink and it is now possible to test the model. A single test is given in the next section.

1.7 A Test Result

The following test illustrates the working of the model. In this test the riding of a motorcycle over a non-flat terrain is simulated. The different items in the simulations are all present. Riding over a slope, riding over a hill and falling on a slope.

Test objective
Test for stability and behaviour properties of the motorcycle model. The objective is to test how the model will act on a non-flat rough terrain.

Description
The terrain function for this test is given by the following formula

$$h(x) = \begin{cases} 0.1x - 2 & \text{for} \quad x \in [20, 25] \\ 0.05x - 1.25 & \text{for} \quad x \in [25, 35] \\ 0.5 - 0.5\cos(0.25(x - 50)) & \text{for} \quad x \in [50, 50 + 8\pi] \\ 0.15x - 12 & \text{for} \quad x \geq 80 \\ 0 & \text{otherwise} \end{cases} \tag{1.27}$$

and is graphical represented by Figure 1.10.

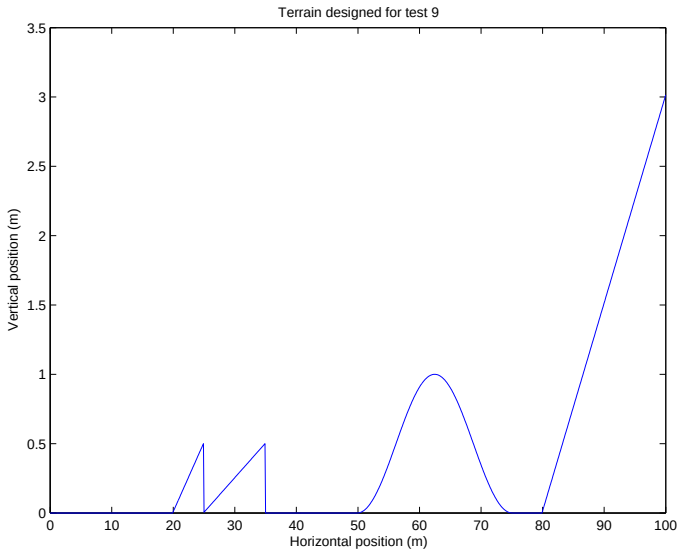

Figure 1.10: *The terrain height defined by the function* $h(x)$

The following initial conditions are taken

i	position	velocities	i	position	velocities
1	(0,0)	(20,0)	4	(1,0)	(20,0)
2	(0,3588)	(20,0)	5	(1,3588)	(20,0)
3	(0,7392)	(20,0)	6	(1,7392)	(20,0)

In this table, i denotes the number of the pointmass as defined in Figure 1.11. After the pointmass i, the positions and the horizontal and vertical velocity of the point mass is given.

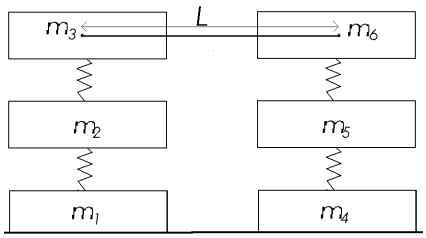

Figure 1.11: *The motorcycle model consisting of two partial vehicles connected by a rod of length L*

The other parameters are as follows:
Mass of point mass i is $m_i = 5$ kg for $i = 1, 2, 3, 4$.
Relaxed spring lengths $z_{rel_{v_i}} = 0.4$ for $i = 1, 2, 3, 4$.
Spring constant k_{v_i} of spring i is $k_{v_i} = 10000$ for $i = 1, 2, 3, 4$.
Damping constant d_{v_i} of spring i is $d_{v_i} = 500$ for $i = 1, 2, 3, 4$.
The elasticity constants for the contact algorithm are both chosen as 0.9.
Friction constant is chosen as $c_w = 1$.
The time-interval is $t \in [0, 6]$.
The step size is chosen as $\Delta t = 0.001$

Results
The results are represented in Figure 1.12. It is seen that when the front part of the motorcycle reaches the first obstacle, it moves upward and the backside follows. Because this obstacle is modeled as a ski jump, the motorcycle jumps and then falls on rising terrain after the obstacle, then the motorcycle falls on a flat terrain, and then the motorcycle passes the hill. After the hill is passed the motorcycle moves upward the final slope. This obstacle is not passed, but there the simulation stopped.

Conclusions
This test gives reason to state that the model has good behaviour and stability properties. The trajectories of the different point masses seem to be realistic.

More tests are represented in Chapter 6. It is seen that the behaviour model is realistic and that the implementation in MATLAB/Simulink is successfully performed. The implementation of a behaviour model as is present here can be used by designers of vehicles. Some remarks on this subject are made in the next section.

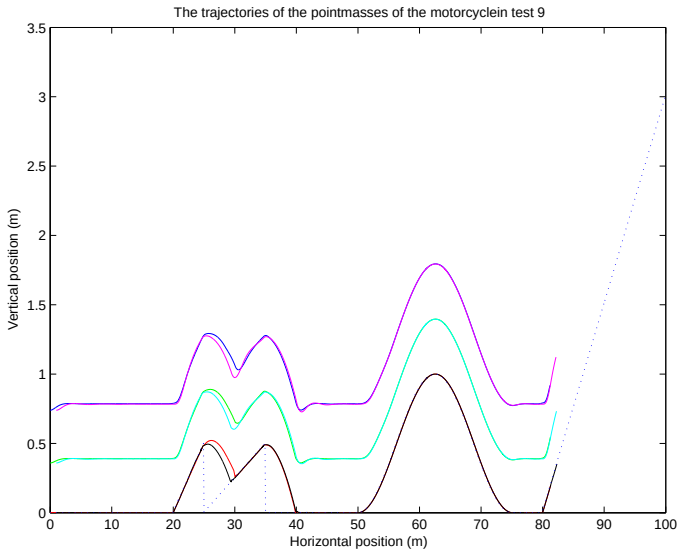

Figure 1.12: *The found trajectories of the point masses of the vehicle when simulating the riding of a motorcycle on a rough terrain.*

1.8 Using the Behaviour Model for Design Studies

The implemented behaviour model can be used to get a feeling how the behaviour of the motorcycle will change when the different parameters change. The simulations performed in Chapter 7 illustrate how this kind of models can be useful for designers. With the model, they can test the design they made for behaviour properties. The model developed and implemented in this research however is not really according to a real motorcycle, but the technique implemented in the model is the same as for a model of real motorcycle. The parameter studies performed in this chapter are only done to show how a model as developed in this research can help designers. With some simulations it is possible to calculate the effects of multiple alterations in a design in less time. In Chapter 7 the results of multiple simulations are represented.

1.9 Conclusions

On base of the results of this research one can conclude that modeling vehicles with use of partial vehicles leads to realistic behaviour models. The Discrete Lagrange Multiplier Method is very appropriate for solving the behaviour models. Implementation in MATLAB/Simulink is possible.

It is recommended to do further research on modeling vehicles with use of partial vehicles. The next step in the research could be the implementation of a four-

wheeled vehicle in three dimensions.

Chapter 2

Modeling a Vehicle with use of Partial Vehicles

Modeling the behaviour of a vehicle is done by use of partial vehicles as explained in Section 1.3. The resulting model equations of a partial vehicle in three dimensions is build up on base of research done in [24],[9] and [27]. The same notations as in Section 1.3 are used.

The mass-spring system will be modeled with three translational degrees of freedom. For point mass i these are the variables $x_{(i-1)3+1}, x_{(i-1)3+2}$ and $x_{(i-1)3+3}$. The moment equations are not formulated here, but all different parts of the vehicle are represented by point masses and the equations of motion of each point mass are formulated.

Now the model is stepwise build up. First is looked for one point mass, see fig 2.1 .

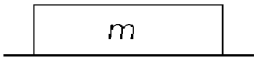

Figure 2.1: *One point mass*

The constraint motion of one point mass ($\mathbf{x} = (x_1, x_2, x_3)^T$) is:

$$\begin{aligned} &m\ddot{\mathbf{x}} = F_c(t) + F_g, \\ &x_3 \geq h(x_1, x_2) \end{aligned} \tag{2.1}$$

One can add a point mass (m_2) to this model and connect this point mass directly above the first one (m_1) by a spring/damping element, see Figure 2.2.

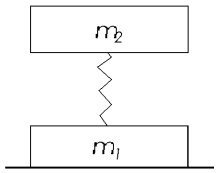

Figure 2.2: *Two point masses directly above each other, connected by a vertical oriented spring.*

The constraint motion of two point masses connected by a spring/damping element (Figure 2.2) is

$$\begin{aligned} &m_1\ddot{x}_3 = -k(z_{rel} - (x_6 - x_3)) + d(\dot{x}_6 - \dot{x}_3) - m_1 g + F_{C_3}(t) \\ &m_2\ddot{x}_6 = k(z_{rel} - (x_6 - x_3)) - d(\dot{x}_6 - \dot{x}_3) - m_2 g + F_{C_6}(t) \\ &m_1\ddot{x}_j = F_{C_j}(t), j = 1, 2 \\ &x_3 \geq h(x_1, x_2) \\ &x_3 + \delta_{max} > x_6 > x_3 + \delta_{min} \\ &x_1 = x_4, x_2 = x_5 \end{aligned} \tag{2.2}$$

Again a point mass (m_3) is added directly above the second point mass (m_2) to complete the partial vehicle, see Figure 2.3.

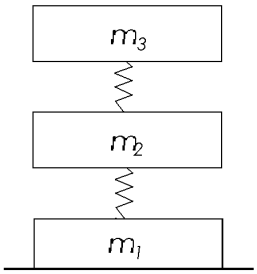

Figure 2.3: *Three point masses directly above each other, connected by two vertical oriented springs. The same the point masses make up a partial vehicle*

The constraint motion of this partial vehicle model is:

$$
\begin{aligned}
&m_1\ddot{x}_3 = -k_{v1}(z_{rel_{v1}} - (x_6 - x_3)) + d_{v1}(\dot{x}_6 - \dot{x}_3) - m_1 g + F_{C_3}(t) \\
&m_2\ddot{x}_6 = k_{v1}(z_{rel_{v1}} - (x_6 - x_3)) - d_{v1}(\dot{x}_6 - \dot{x}_3) \\
&\qquad -k_{v2}(z_{rel_{v2}} - (x_9 - x_6)) + d_{v2}(\dot{x}_9 - \dot{x}_6) - m_2 g + F_{C_6}(t) \\
&m_3\ddot{x}_9 = k_{v2}(z_{rel_{v2}} - (x_9 - x_6)) - d_{v2}(\dot{x}_9 - \dot{x}_6) - m_3 g + F_{C_9}(t) \\
&m_1\ddot{x}_j = F_{C_j}(t), j = 1, 2 \\
&x_3 \geq h(x_1, x_2) \\
&x_3 + \delta_{max_{v1}} > x_6 > x_3 + \delta_{min_{v1}} \\
&x_6 + \delta_{max_{v2}} > x_9 > x_6 + \delta_{min_{v2}} \\
&x_1 = x_4 = x_7, x_2 = x_5 = x_8
\end{aligned}
\tag{2.3}
$$

Constructing the 'motorcycle model' existing of two partial vehicles, can be done by connecting the two partial vehicles by a massless rod of length L (see Figure 2.4) .

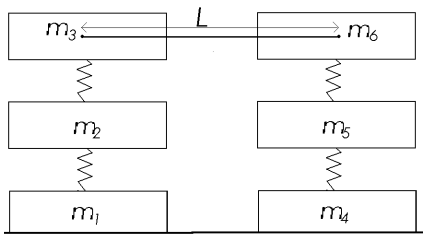

Figure 2.4: *The motorcycle model consisting of two partial vehicles connected by a rod of length L*

The result is a model with six point masses and four spring/damping elements with relaxed spring lengths $z_{rel_{v1}}$, $z_{rel_{v2}}$, $z_{rel_{v3}}$ and $z_{rel_{v4}}$. The behaviour model consists of two copies of (2.3) with extra constraint:

$$(x_{16} - x_7)^2 + (x_{17} - x_8)^2 + (x_{18} - x_9)^2 = L^2 \tag{2.4}$$

The complete model of the four-wheeled vehicle, as represented in Figure 1.1, can be modeled as two copies of the above 'motorcycle model' with extra constraint:

$$(x_{25} - x_7)^2 + (x_{26} - x_8)^2 + (x_{27} - x_9)^2 = L_1^2. \tag{2.5}$$

Here, L_1 is the length of the diagonal, the distance between the point masses m_3 and m_9.

After developing this model one can see that extending the model with more partial vehicles can be done by adding some equations. The structure of the model for a two-wheeled vehicle and the structure of the model for a four-wheeled vehicle is the same. Extending the model means adding known equations.

When considering a real motorcycle and the motorcycle model, one can see that, in real, the angle between the upper springs damping elements and the chassis

keeps equal. However, this is not implemented in the model. In the model the horizontal point mass are located directly above each other and only the height of the three different point masses differs. Therefore the constraints

$$x_1 = x_4 = x_7 \quad \text{and} \quad x_2 = x_5 = x_8$$

are replaced by constraints which yield that the springs act perpendicular to the chassis. When this is performed, it also necessary to apply the spring and damping forces, not only in vertical direction, but also in three dimensions. The orientation of the spring is no longer fixed. This has to be applied too.

The adapted behaviour model for a partial vehicle will be constructed. Other and greater vehicles then can be constructed by adding known equations of motion of a partial vehicle.

Before formulating the model of a partial vehicle with only two translational degrees of freedom some definitions are made

- $d(m_i, m_j)$ is the distance between point mass i and point mass j, $i, j = 1, 2, 3$,
- $F_{v_i} = k_{vi}(z_{rel} - d(m_i, m_{i+1})$ is the spring force in spring damping element i for $i = 1, 2$,
- $F_{d_i} = d_{v_i}(\frac{\partial}{\partial t} d(m_i, m_{i+1})$is the damping force in spring damping element i for $i = 1, 2$,
- the angles α and β are defined as in Figure 2.5.

$$\begin{aligned}
m_1\ddot{x}_1 &= -\sin(\alpha)(F_{v_1} - F_{d_1}) + F_{C_1} && \text{(2.6a)}\\
m_1\ddot{x}_2 &= -\cos(\alpha)(F_{v_1} - F_{d_1}) - m_1 g + F_{C_2} && \text{(2.6b)}\\
m_2\ddot{x}_3 &= \sin(\alpha)(F_{v_1} - F_{d_1}) - \sin(\beta)(F_{v_2} - F_{d_2}) + F_{C_3} && \text{(2.6c)}\\
m_2\ddot{x}_4 &= \cos(\alpha)(F_{v_1} - F_{d_1}) - \cos(\beta)(F_{v_2} - F_{d_2}) - m_2 g + F_{C_4} && \text{(2.6d)}\\
m_3\ddot{x}_5 &= \sin(\beta)(F_{v_2} - F_{d_2}) + F_{C_5} && \text{(2.6e)}\\
m_3\ddot{x}_6 &= \cos(\beta)(F_{v_2} - F_{d_2}) - m_3 g + F_{C_6} && \text{(2.6f)}\\
x_2 &\geq h(x_1). && \text{(2.6g)}
\end{aligned}$$

For the motorcycle model with two translational degrees of freedom two copies of the above model and the following constraints are needed.

$$\begin{aligned}
0 &= d(m_3, m_6) - L, && \text{(2.7a)}\\
0 &= d(m_2, m_3)^2 + L^2 - d(m_2, m_6)^2 && \text{(2.7b)}\\
0 &= d(m_5, m_6)^2 + L^2 - d(m_5, m_3)^2 && \text{(2.7c)}
\end{aligned}$$

The constraint (2.7a) defines the length of the massless rod between the two upper point masses. The constraints (2.7b) and (2.7c) define the orientation of the upper springs to the chassis. The springs are set to act perpendicular to the chassis. The constraints follow from the theorem of Pythagoras. This model is implemented in MATLAB/Simulink in this research. The partial vehicle with

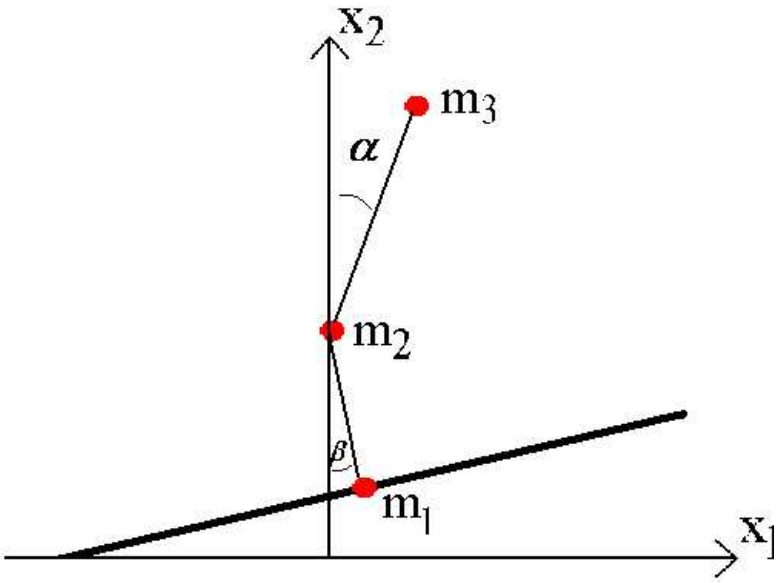

Figure 2.5: *The schematic representation of the state of a partial vehicle which contains the angles α and β.*

point masses 1,2, and 3 is determined to be the backside of the motorcycle and the other partial vehicle is determined to be the front side of the motorcycle.

To define the model of the mass-spring system with 3 translational degrees of freedom. In Figure 2.6 the used angles α, β, γ and δ are defined.

The equations of motion for a partial vehicle in three dimensions become

$$\begin{array}{rcl}
m_1\ddot{x}_1 &=& \cos(\gamma)(-\sin(\alpha)(F_{v_1} - F_{d_1})) + F_{C_1}, \\
m_1\ddot{x}_2 &=& \sin(\gamma)(-\sin(\alpha)(F_{v_1} - F_{d_1}) - \cos(\alpha)(F_{v_1} - F_{d_1})) + F_{C_2}, \\
m_1\ddot{x}_3 &=& \cos(\gamma)(-\cos(\alpha)(F_{v_1} - F_{d_1})) - m_1 g + F_{C_3}, \\
m_2\ddot{x}_4 &=& \cos(\gamma)(\sin(\alpha)(F_{v_1} - F_{d_1})) + F_{C_4}, \\
m_2\ddot{x}_5 &=& \sin(\gamma)(\sin(\alpha)(F_{v_1} - F_{d_1}) + \cos(\alpha)(F_{v_1} - F_{d_1})) \\
&& -\sin(\delta)(\sin(\beta)(F_{v_2} - F_{d_2}) + \cos(\beta)(F_{v_2} - F_{d_2})) + F_{C_5}, \\
m_2\ddot{x}_6 &=& \cos(\gamma)(\cos(\alpha)(F_{v_1} - F_{d_1})) - m_2 g + F_{C_6}, \\
m_3\ddot{x}_7 &=& \cos(\delta)(\sin(\beta)(F_{v_2} - F_{d_2})) + F_{C_7}, \\
m_3\ddot{x}_8 &=& \sin(\delta)(\sin(\beta)(F_{v_2} - F_{d_2}) + \cos(\beta)(F_{v_2} - F_{d_2})) + F_{C_8}, \\
m_3\ddot{x}_9 &=& \cos(\delta)(\cos(\beta)(F_{v_2} - F_{d_2})) - m_3 g + F_{C_9}.
\end{array} \tag{2.8}$$

The motorcycle-model with three translational degrees of freedom is give by 2 copies of the above model and the three constraints (2.7).

The four-wheeled vehicle needs two copies of the motorcycle model and some extra constraints. For the numbering of the point masses and L_1 and L_2, see Figure 2.7

The constraints are given in (2.9)

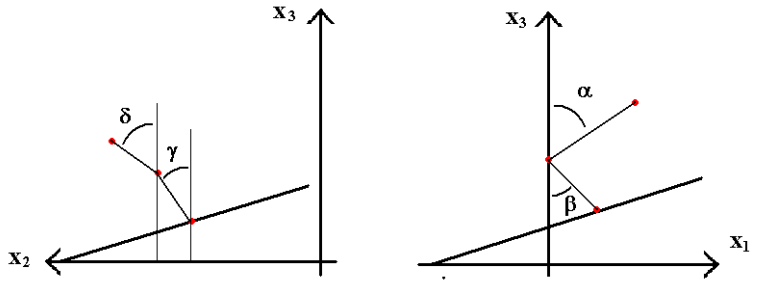

Figure 2.6: *The schematic representations of the (three dimensional) states of a partial vehicle which contains the angles α, β, γ and δ.*

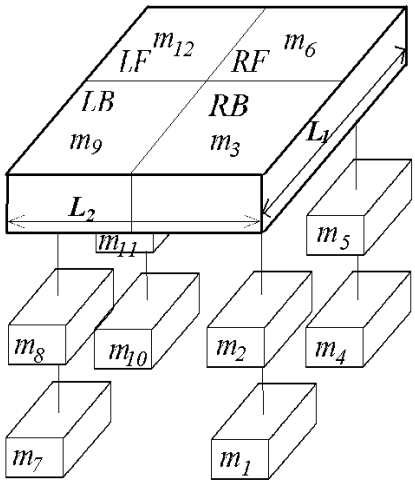

Figure 2.7: *The schematic representation the four-wheeled vehicle, with numbered masses and definition of L_1 and L_2.*

$$
\begin{aligned}
0 &= d(m_3, m_6) - L_1, && \text{(2.9a)}\\
0 &= d(m_2, m_3)^2 + L_1^2 - d(m_2, m_6)^2 && \text{(2.9b)}\\
0 &= d(m_5, m_6)^2 + L_1^2 - d(m_5, m_3)^2 && \text{(2.9c)}\\
0 &= d(m_9, m_{12}) - L_1, && \text{(2.9d)}\\
0 &= d(m_8, m_{11})^2 + L_1^2 - d(m_8, m_{12})^2 && \text{(2.9e)}\\
0 &= d(m_{11}, m_{12})^2 + L_1^2 - d(m_{11}, m_9)^2 && \text{(2.9f)}\\
0 &= d(m_2, m_3)^2 + L_2^2 - d(m_2, m_9)^2 && \text{(2.9g)}\\
0 &= d(m_8, m_9)^2 + L_2^2 - d(m_8, m_3)^2 && \text{(2.9h)}\\
0 &= d(m_5, m_6)^2 + L_2^2 - d(m_5, m_12)^2 && \text{(2.9i)}\\
0 &= d(m_{11}, m_{12})^2 + L_2^2 - d(m_{11}, m_6)^2 && \text{(2.9j)}\\
0 &= d(m_3, m_6)^2 + d(m_6, m_{12})^2 - d(m_3, m_{12})^2 && \text{(2.9k)}
\end{aligned}
$$

The constraints (2.9) are the equality constraints. However, there are some natural inequality constraints. There are two kinds of inequality constraints. The following constraints imply that the wheel contact points are always above or on the terrain and that the rims are always above the wheel contact points

$$\begin{cases} x_6 > x_3 & \geq & h(x_1, x_2), \\ x_{15} > x_{12} & \geq & h(x_{10}, x_{11}), \\ x_{24} > x_{21} & \geq & h(x_{19}, x_{20}), \\ x_{33} > x_{30} & \geq & h(x_{28}, x_{29}). \end{cases} \tag{2.10}$$

The next set of inequality constraints imply that the point mass representing the chassis part is always above the rim

$$\begin{cases} x_6 & < & x_9, \\ x_{15} & < & x_{18}, \\ x_{24} & < & x_{27}, \\ x_{33} & < & x_{36}. \end{cases} \tag{2.11}$$

The resulting equations (2.8), (2.9), (2.10) and (2.11) are also represented in Chapter 1.

Chapter 3

Solving Differential Algebraic Equations

3.1 Introduction

In Section 1.4 the Discrete Lagrange Multiplier Method (DLMM) is presented as a solver that solves Ordinary Differential Equations (ODEs) subject to equality constraints in a stable way. This chapter contains the justification of the choice for the DLMM. The following methods are described and discussed in this chapter

- the Continue Lagrange Multiplier Method (CLMM),
- the Discrete Lagrange Multiplier Method (DLMM),
- the Stabilized ODE Method,
- the Post Stabilization Method,
- the Coordinate Projection Method,
- the Half-Explicit Runge Kutta Methods.

These methods are all explicit methods. The Half-Explicit Runge Kutta Methods are for some kind of problems explicit as well. There are many more methods which can solve DAEs, but before the model has to be implemented in a real-time simulator, explicit methods are preferable, because of there efficiency. The methods do not only need to be explicit, a fixed step size and a fixed method is also desired. Because of these requirements the often called method DASSL is not appropriate for the purposes of this research. The method DASSL is based on the backward differentiation formulas. These are implicit solvers with variable step size and variable order [10, Chapter 2] .

The main of this chapter consists of the describing, testing and discussing of the several methods. All these methods are discussed and tested for the behaviour model (1.20) of a planar pendulum. The stability properties of the different methods will be explained.

3.2 Lagrange Multiplier Methods

3.2.1 The Continue Lagrange Multiplier Method

Consider the dynamical system

$$G(\mathbf{x})\dot{\mathbf{x}} = B(\mathbf{x}) + F \tag{3.1a}$$
$$\mathbf{0} = P(\mathbf{x}, t). \tag{3.1b}$$

In this system $\mathbf{x} \in \mathbb{R}^n$ is the state-vector of the system. The matrices G $(n \times n)$ and B $(n \times 1)$ are assumed to be known. It is assumed that $G(\mathbf{x})$ is non-singular $\forall \mathbf{x} \in \mathbb{R}^n$. This means that (3.1a) is a true ODE. The vector F represents the control forces. The matrix B represents the Coriolis, gravitational and centrifugal force/torque vector of the system. The matrix $C = P_x(\mathbf{x}, t)$ is introduced. It is assumed that $\forall (\mathbf{x}, t) \in \mathbb{R}^{n+1}$ such that $P(\mathbf{x}, t) = 0$: $\text{rank}(C(\mathbf{x}, t)) = m, m \leq n$.

The manifold S is defined as

$$S = (\mathbf{x}, t) \in \mathbb{R}^{n+1} : P(\mathbf{x}, t) = 0$$

From the theorem about the formulation of dynamical systems described by a DAE [6, page 321] , it follows that the system (3.1) is equivalent with

$$G(\mathbf{x})\dot{\mathbf{x}} = B(\mathbf{x}) + F + GZC^T\lambda_Z, \tag{3.2a}$$
$$\lambda_Z = -(CZC^T)^{-1}(CG^{-1}(B + F) - d), \tag{3.2b}$$

where $\mathbf{x}(t_0) = \mathbf{x}_0$, $d = P_t(\mathbf{x}, t)$, and the matrix $Z = Z(\mathbf{x}, t)$ such that

$$\forall (\mathbf{x}, t) \in S : rank(CZC^T) = rank(C)$$

and $(\mathbf{x}_0, t_0) \in S$.

The matrix Z can be used for optimization purposes and is referred to as a weighting matrix, λ is called the *generalized Lagrange multiplier*. This name is motivated by the Lagrange multiplier that is present in the formulation of mechanical systems (1.17). In this case however, the Lagrange multiplier also depends on the matrix Z.

An ODE-solver can be used to solve the ODE (3.2a), with for λ_Z the expression as in (3.2b). This defines the CLMM.

3.2.2 The Discrete Lagrange Multiplier Method

The DLMM is derived from the CLMM. In [6, page 323] is shown that the CLMM leads to error accumulation once an error is made in the calculation of λ_Z.

To avoid this error accumulation the method has changed to the DLMM. When first discretizing the model equations and then solving λ for the discretization of the model equations, the *discrete generalized Lagrange multiplier* λ_{nZ}^d

$$\lambda_{nZ}^d = -(CZC^T)^{-1}(CG_n^{-1}(B_n + F_{cn}) - d + P_n/\Delta t) \tag{3.3}$$

is obtained [6, page 325]. The difference between the CLMM and the DLMM is the sequence of steps to obtain an expression for λ. For the CLMM the expression for λ_Z in (3.2a) is obtained before discretizing, and for the DLMM the expression for λ is obtained after discretizing the ODE (3.2a). Note that the term $GZC^T\lambda$ physically represents the force to let the system satisfy the constraint, as in system (1.17).

For the DLMM, the theorem about stable numerical integration of DAEs with Forward-Euler [5, Theorem 3.6] states that system (3.1) can be solved in stable manner by solving (3.2a) with for λ_Z the discrete generalized Lagrange multiplier (3.3).

3.2.3 Application to Constrained Mechanical Systems

The CLMM and DLMM can be applied to constrained mechanical systems. The dynamical system (1.18) is considered.

$$\begin{aligned}
\dot{\mathbf{x}} &= \mathbf{y}, && (3.4a)\\
M\dot{\mathbf{y}} &= B + F_c, && (3.4b)\\
\mathbf{0} &= P(\mathbf{x}, t), && (3.4c)\\
\mathbf{0} &= C\mathbf{y} - d. && (3.4d)
\end{aligned}$$

The manifold S is redefined as

$$S = \{(\mathbf{x}, \mathbf{y}, t) \in \mathbb{R}^{2n+1} | P(\mathbf{x}, t) = \mathbf{0} \quad \text{and} \quad C(\mathbf{x}, t)\mathbf{y} = d(\mathbf{x}, t)\}$$

It follows from the theorem about the formulation of constrained mechanical systems [6, page 329] that (3.4) is equivalent with

$$\begin{aligned}
\dot{\mathbf{x}} &= \mathbf{y} + XC^T\mu_X, && (3.5a)\\
M\dot{\mathbf{y}} &= B + F + MYC^T\lambda_Y, && (3.5b)\\
\mathbf{0} &= P(\mathbf{x}, t), && (3.5c)\\
\mathbf{0} &= C\mathbf{y} - d, && (3.5d)
\end{aligned}$$

where $(\mathbf{x}(t_o), \mathbf{y}(t_0)) = (\mathbf{x}_0, \mathbf{y}_0)$, X and Y are matrices such that $\forall(\mathbf{x}, \mathbf{y}, t) \in S, rank(CXC^T) = rank(C) = rank(CYC^T)$ and $\mathbf{x}_0$, $\mathbf{y}_0$ and t_0 are vectors such that $(\mathbf{x}_0, \mathbf{y}_0, t) \in S$. In this formulation two Lagrange multipliers μ_X and λ_Y are present.

For the CLMM, when solving for the Lagrange multipliers before discretizing (3.5a) and (3.5b), the following generalized Lagrange multipliers are obtained

$$\lambda = -(CM^{-1}C^T)^{-1}(CM^{-1}(B + F_c) + \dot{C}\mathbf{y} - \dot{d}), \quad \mu = 0.$$

The expressions for the Lagrange multipliers μ_X^d and λ_Y^d on the interval $[t_n, t_{n+1}]$ for the DLMM become

$$\mu_X^d = -(CXC^T)^{-1}(C\mathbf{y} - d + P_n/\Delta t),$$
$$\lambda_Y^d = -(CYC^T)^{-1}(CM^{-1}(B + F_c) + \dot{C}\mathbf{y} - \dot{d} + (C_n y_n - d_n)/\Delta t).$$

3.2.4 Test Results

Both the CLMM and the DLMM are tested for the planar pendulum problem (1.20). This system is solved for $t \in [0, 1]$.

Test objective
Search for stability properties for the two Lagrange multiplier methods. Search for the power of the method in handling equality constraints.

Results
In Figure 3.1 both solutions for the planar pendulum problem are plotted in one figure.

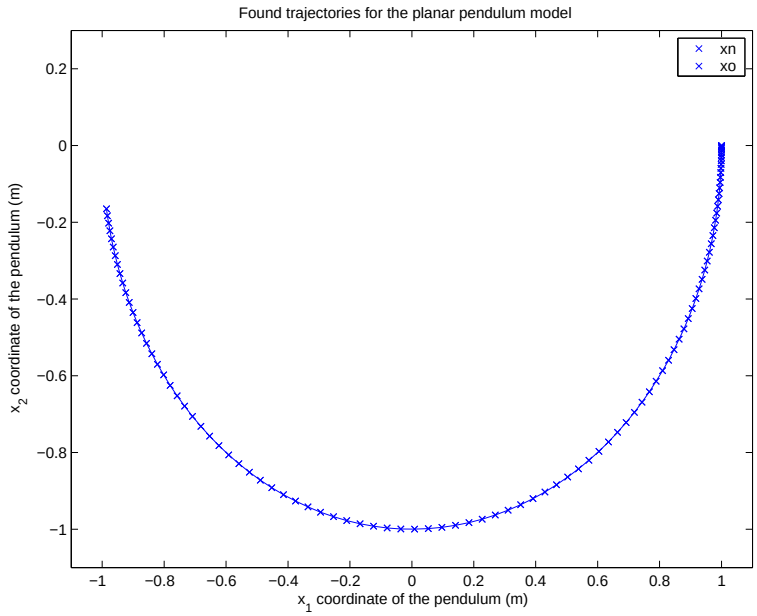

Figure 3.1: *The solution of the planar pendulum problem found by the two Lagrange multiplier methods. The line represents the solution for the CLMM, the crosses represent the solution found by the DLMM. The ODE-solver used was the classical fourth-order Runge Kutta method. Just as in [6] and [5] X and Y were chosen as $X = I$ and $Y = M^{-1}$. The mass of the point mass is $m = 1$ and the length of the rod is $L = 1$. The start position of the mass is (1,0). The simulation starts when the mass is in rest.*

In Figure 3.2 the deviation of the constraint (1.20c) in time is plotted.

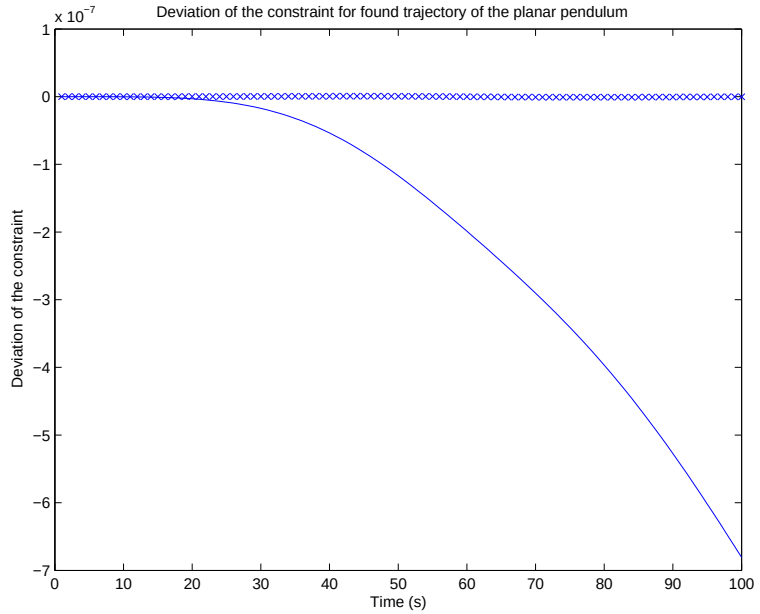

Figure 3.2: *The deviation of the constraint in time. The line represents the deviation of the constraint for the solution found by the CLMM, the crosses represent the deviation of the constraint for the solution found by the DLMM. The ODE-solver used was the classical fourth-order Runge Kutta method. Just as in [6] and [5] X and Y were chosen as $X = I$ and $Y = M^{-1}$. The mass of the point mass is $m = 1$ and the length of the rod is $L = 1$.*

Conclusions
In Figure 3.1 there is no difference visible for the solutions of the two methods. At first sight both methods give the same correct solution. But from Figure 3.2 one can see that the CLMM has an increasing deviation of the constraint while the DLMM has not. Therefore, the DLMM is better on constraint satisfaction then the CLMM. For the interval in which the problem is solved, both methods can be used in a stable way.

Stability Analysis

As already stated, a stability analysis of the CLMM and the DLMM is given in [5, Chapter 3] and [6, Chapter 3]. Here the results of this analysis are given.

For the CLMM is shown that solving the system (3.2) leads to error accumulation once an error is made in the calculation of λ_Z, or if one starts with initial conditions that are not on the manifold S.

In [5, Theorem 3.6] and in [6, page 325,326] it is stated that for the DLMM the deviation of the constraint goes to zero, if the step size goes to zero.

3.3 Solving Ordinary Differential Equations on Manifolds

In this section some methods designed for Ordinary Differential Equations (ODEs) on manifolds are described

- the stabilized ODE method,
- the post stabilization method,
- the coordinate projection method.

The general form of DAEs that can be interpreted as ODEs on manifolds is

$$\dot{\mathbf{x}} = \hat{\mathbf{f}}(\mathbf{x}) \tag{3.6a}$$
$$\mathbf{0} = \mathbf{h}(\mathbf{x}) \tag{3.6b}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the state-vector of the system, $\hat{\mathbf{f}}$ a vector function in $\mathbb{R}^n$ and $\mathbf{h}(\mathbf{x})$ a vector function defining the manifold.

The methods now described are further described and discussed in [7, Chapter 9 and 10]. The stabilized ODE method tries to solve system (3.6) by solving ODE (3.6a) after adding a stabilization term to this ODE. The post stabilization and coordinate projection method, first solve the ODE (3.6a) with a known ODE-solver and after each step of this ODE-solver a stabilization process will be applied. This stabilization process is designed to let the solution (more) satisfy the constraint equations (3.6b) .

3.3.1 Stabilized ODE Method

First the stabilized ODE method is described. Instead of solving (3.6) one can look for an ODE that automatically satisfies the constraint. Therefore, the ODE (3.6a) is stabilized or attenuated with respect to the invariant set $\mathcal{M} = \{\mathbf{x} \mid \mathbf{h}(\mathbf{x}) = \mathbf{0}\}$.

The ODE

$$\dot{\mathbf{x}} = \hat{\mathbf{f}}(\mathbf{x}) - \gamma F(\mathbf{x})\mathbf{h}(\mathbf{x}) \tag{3.7}$$

obviously has the same solutions as (3.6a) on $\mathcal{M}$ (i.e. when $h(\mathbf{x}) = 0$).

For F such that HF, where $H = \mathbf{h}_x$, is positive definite, and if the positive parameter γ large enough, then, solving the ODE (3.7) can be done with asymptotic stability. This means that any trajectory of (3.7) starting from some initial value near $\mathcal{M}$ will tend towards satisfying the constraints, i.e. towards the manifold. Moreover, this tendency is monotonic

$$|\mathbf{h}(\mathbf{x}(t+\alpha))| \leq |\mathbf{h}(\mathbf{x}(t))| \tag{3.8}$$

For example, consider the mechanical system

$$\begin{aligned} \dot{\mathbf{q}} &= \mathbf{v} \\ M(\mathbf{q})\dot{\mathbf{v}} &= \mathbf{f}(\mathbf{q},\mathbf{v}) - G^T(\mathbf{q})\lambda, \\ \mathbf{0} &= \mathbf{g}(\mathbf{q}), \end{aligned} \tag{3.9}$$

in [7, page 251], it is shown that with $F(\mathbf{x}) = H^T(HH^T)^{-1}$ and $\gamma > 1$ asymptotic stability occurs.

Test Results

In this subsection the test results for the stabilized ODE method are given. Again applying to the planar pendulum problem (1.20) tests the method.

Test objective
Search for stability properties of the stabilized ODE method. Search for the power of the method in handling equality constraints.

Results
In the figures 3.3 and 3.4 the results of this test are represented.

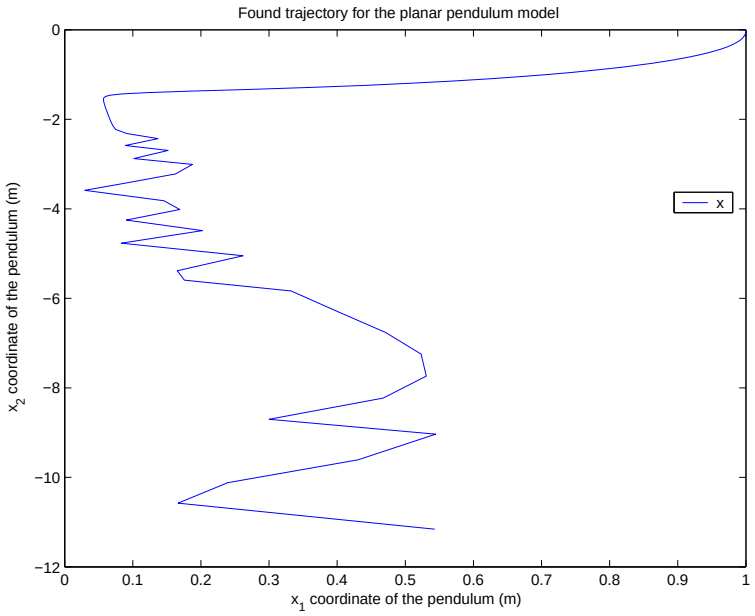

Figure 3.3: *The solution 'x' of the planar pendulum problem (1.20) approximated by the stabilized ODE method. The ODE-solver used was the classical fourth order Runge Kutta method. $F(\boldsymbol{x})$ is chosen as $F(\boldsymbol{x}) = H^T(HH^T)^{-1}$ and for γ is chosen $\gamma = 10$. The start position of the mass is (1,0). The simulation starts when the mass is in rest.*

It can be seen in Figure 3.3 that from a certain point in time instability behaviour occurs. From Figure 3.4 one can see that the deviation of the constraint

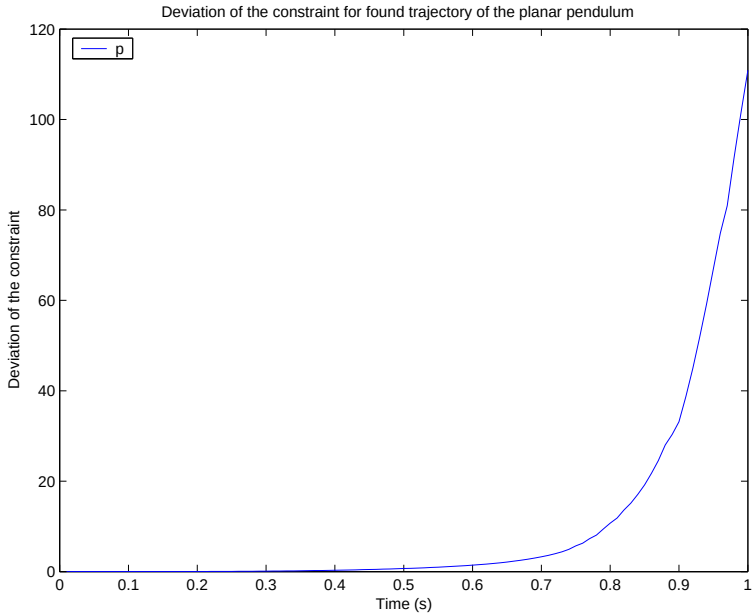

Figure 3.4: *The deviation of the constraint for the planar pendulum problem (1.20) solved by the stabilized ODE method. The ODE-solver used was the classical fourth order Runge Kutta method. Tha matrix $F(\boldsymbol{x})$ is chosen as $F(\boldsymbol{x}) = H^T(HH^T)^{-1}$ and for γ is chosen $\gamma = 10$.*

grows exponentially.

Conclusions
The stabilized ODE method cannot be used for solving the problem (1.20). The deviation of the constraint grows exponentially and the method does not have the required stability properties to solve (1.20).

Stability Analysis

If one applies Forward Euler to the ODE (3.7), one obtains

$$\mathbf{x}(t_{n+1}) = \mathbf{x}_{n+1} + \epsilon = \mathbf{x}_n + h\dot{\mathbf{x}}_n + \epsilon = \mathbf{x}_n + h(\hat{\mathbf{f}}(\mathbf{x}_n) - \gamma F(\mathbf{x}_n)\mathbf{h}(\mathbf{x}_n)) + \epsilon, \quad (3.10)$$

where ϵ is the error due to the numerical method. When interpreting the Forward-Euler method as a first-order Taylor approximation, about the error ϵ is found that

$$\epsilon = \frac{h^2}{2}\ddot{\mathbf{x}}(t_n) + \mathcal{O}(h^3), \quad (3.11)$$

although second order in h, may not decrease and may even grow arbitrarily with h fixed, if $\ddot{\mathbf{x}}$ grows. This is the case now. Remember that a pendulum has periodic behaviour. This means that on some intervals $\ddot{\mathbf{x}}$ shall grow.

This is the explanation for the instability when applying Forward Euler to the system (1.20). The same steps can be done for the fourth-order Runge Kutta method.

Al though instability in this test, the stabilized ODE-method is not useless. In [7, Example 9.10] an example is given, where this method is successfully applied.

3.3.2 Post Stabilization and Coordinate Projection

In this section the post stabilization method and the coordinate projection method are described. The two methods are nearly the same. For solving system (3.6), both methods require two steps each time-step.

The post stabilization method can be described as follows.

Step 1, solve the ODE (3.6a) with a known ODE-solver for one time-step - At time t_{n-1} the approximate solution is $\mathbf{x}_{n-1}$, application of the ODE-solver gives

$$\tilde{\mathbf{x}}_n = \phi_h^f(\mathbf{x}_{n-1}),$$

where ϕ_h^f returns the approximated value on time t_n, with the used ODE-solver. (e.g. forward Euler: $\phi_h^f(\mathbf{x}_{n-1}) = \mathbf{x}_{n-1} + h\hat{f}(\mathbf{x}_{n-1})$).

Step 2, apply a stabilization step to the result $\tilde{\mathbf{x}}_n$ of step 1 - This stabilization step is designed to let the result more satisfy the constraint. The value after stabilizing is given by

$$\mathbf{x}_n = \tilde{\mathbf{x}}_n - F(\tilde{\mathbf{x}}_n)\mathbf{h}(\tilde{\mathbf{x}}_n), \tag{3.12}$$

where $F(\tilde{\mathbf{x}}_n) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ is called the *stabilization matrix*.

The coordinate projection method can be described as follows.

Step 1, solve the ODE (3.6a) with a known ODE-solver for one time-step - At time t_{n-1} the approximate solution is $\mathbf{x}_{n-1}$, application of the ODE-solver gives

$$\tilde{\mathbf{x}}_n = \phi_h^f(\mathbf{x}_{n-1}),$$

where ϕ_h^f returns the approximated value on time t_n, due to the used ODE-solver. (e.g. forward Euler: $\phi_h^f(\mathbf{x}_{n-1}) = \mathbf{x}_{n-1} + h\hat{f}(\mathbf{x}_{n-1})$).

Step 2, apply a stabilization step to the result $\tilde{\mathbf{x}}_n$ of step 1- This stabilization step is designed to let the result more satisfy the constraint. The solution after coordinate projection on time t_n, $\mathbf{x}_n$, is determined as the minimizer of $|\mathbf{x}_n - \tilde{\mathbf{x}}_n|_2$ such that

$$\mathbf{0} = \mathbf{h}(\mathbf{x}_n).$$

(with $|.|_2$ the Euclidean norm in $\mathbb{R}^n$). The result is a constrained least squares minimization problem to be solved for $\mathbf{x}_n$ at each step n. The post

stabilization method with $F = H^T(HH^T)^{-1}$ coincides with one Newton step for this local minimization problem. For more on solving constrained least squares problems is referred to [3]

The two methods almost coincide when the step size h is very small. Since explicit methods are preferable, the iteration process of the local minimization problem of the coordinate projection method is broken off after a certain number n of iteration steps.

A topic of interest to the post stabilization method is how to choose the stabilization matrix F. The smaller $\|I - HF\|$ is, the more effective the post stabilization step will be [7, page 285]. The choice $F = H^T(HH^T)^{-1}$, which was mentioned earlier, or more generally the choice corresponding to one Newton step of coordinate projection $F = D^T(HD^T)^{-1}$, achieves the minimum $HF = I$. However, choices of F satisfying $HF = I$ may be expensive to apply, because $H = \mathbf{h}_\mathbf{x}$ can be very complicated. To avoid such complicated computations more choices of F are possible. For more details about this subject we refer to [7, page 285].

Test Results

Test objective
Search for stability properties and properties concerning the handling of constraints, for both post stabilization and coordinate projection method.

Results
The figures 3.5, 3.6, 3.7 and 3.8 represent the results.

Conclusions
It is seen that, for both methods, from a certain point in time instability occurs. It also visible in Figure 3.8 and Figure 3.6 that the deviation of the constraint for the coordinate projection method is greater than for the post stabilization method.

Stability Problems for the Post Stabilization Method and the Coordinate Projection Method

To explain the unstable behaviour of these methods it is necessary to say something about the iteration process of Newton to minimize $|\mathbf{x}_n - \tilde{\mathbf{x}}_n|_2$. The minimum of $|\mathbf{x}_n - \tilde{\mathbf{x}}_n|_2$, with $\mathbf{h}(\mathbf{x}_n) = 0$ is found when the vector function $\frac{\partial |\mathbf{x}_n - \tilde{\mathbf{x}}_n|_2}{\partial \mathbf{x}} = 0$. To compute this zero the Newton iteration process ([11, Section 2.2], [19, Section 1.5] and [16, Section 4.2]) is used.

To use the iteration process of Newton successfully it is necessary to start with a value sufficiently close to the minimum. For the two-dimensional case, for example when a point of inflection lies between the starting point and the minimum, the iteration process will diverge ([4, page 346], [19, page 11]).

Of course there are other explicit methods for minimizing $|\mathbf{x}_n - \tilde{\mathbf{x}}_n|_2$, subject

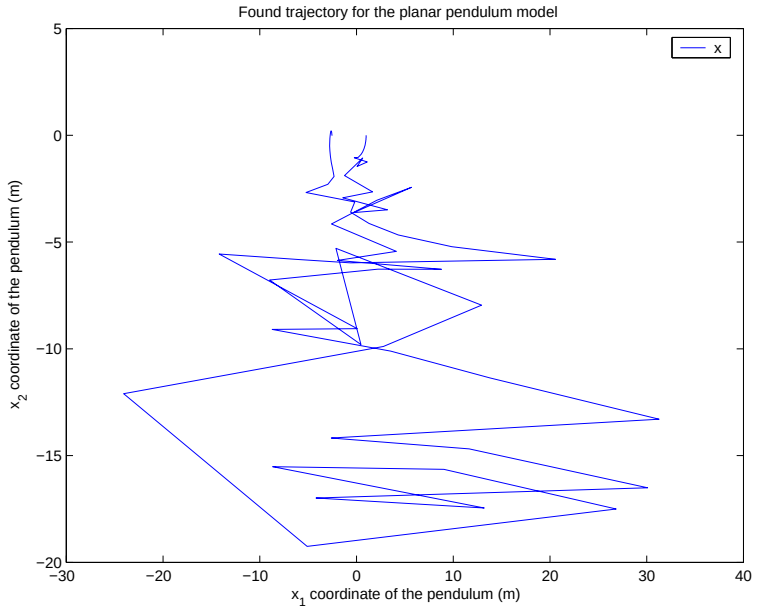

Figure 3.5: *The computed solution for the planar pendulum problem (1.20) by the post stabilization method. The stabilization matrix F is $F = H^T(HH^T)^{-1}$. The start position of the mass is (1,0). The simulation starts when the mass is in rest.*

to $\mathbf{h}(\mathbf{x}_n) = 0$, but for all, they need a starting value sufficiently close to the minimum.

In Section 3.3.1 it was seen that the error due to the numerical method might grow arbitrarily. Therefore, the distance between the starting point of the Newton iteration process and the desired value for the minimum of $|\mathbf{x}_n - \tilde{\mathbf{x}}_n|_2$ grows each time-step. And, dependent on the ODE, the Newton iteration process will diverge. This is the case in this test. It explains why there are no stability problems before a certain point in time and why the coordinate projection method is more instable then the post stabilization method.

In the test case the solution $\mathbf{x}(t)$ of the problem is a periodic function. This means that there are many points of inflection and that the distance between two points of inflection is bounded. For an arbitrarily growing error, due to the numerical method, there will be at some point a wrong starting point for the Newton iteration process.

Though it is concluded that this method is not appropriate enough for this research, this will not mean that these methods are completely ineffective. In [7, Example 10.7] an example is given where these methods give the exact solution of the model.

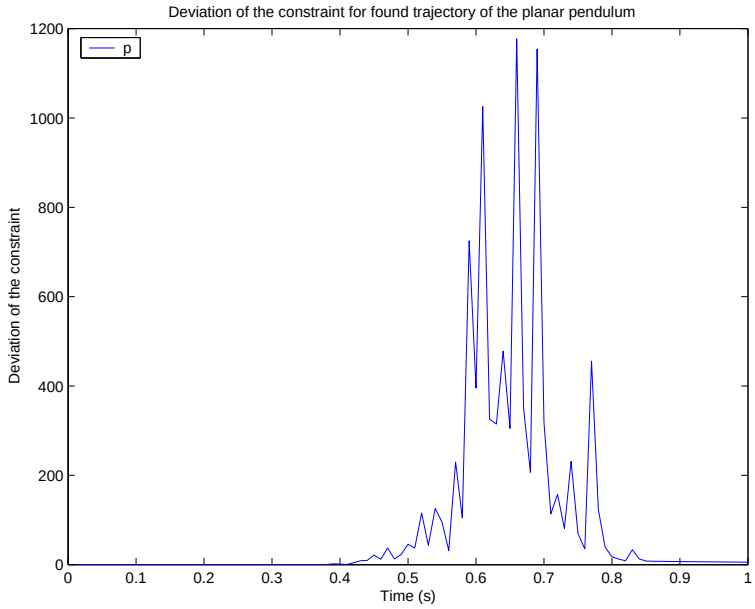

Figure 3.6: *The deviation of the constraint for the planar pendulum problem (1.20) solved by the post stabilization method. The stabilization matrix F is $F = H^T(HH^T)^{-1}$.*

3.4 Half-Explicit Runge Kutta Methods

In this section the half-explicit Runge Kutta methods are described. Half-explicit Runge Kutta methods are explicit Runge Kutta methods that are applied to DAEs in a special way. The methods for DAEs with index 1 and 2 are described here. The application of Runge Kutta methods to ODEs is discussed in any textbook on numerical analysis. For example see [13, Section 8.3.3],[12, Section 2.4] and [22, page 212]. For the description of the half-explicit Runge Kutta methods [17, Chapter 1,2 and 3] is used.

To problems of the form

$$\begin{aligned} \dot{\mathbf{y}} &= \mathbf{f}(\mathbf{y}, \mathbf{z}) \\ \mathbf{0} &= \mathbf{g}(\mathbf{y}, \mathbf{z}) \end{aligned} \tag{3.13}$$

with $\mathbf{y} \in \mathbb{R}^n$, $\mathbf{z} \in \mathbb{R}^m$, $\mathbf{f} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ and $\mathbf{g} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^m$, explicit Runge Kutta methods can be applied as follows:

$$\mathbf{Y}_{ni} = \mathbf{y}_n + \mathbf{h}\sum_{j=1}^{i-1} a_{ij}\mathbf{f}(\mathbf{Y}_{nj}, \mathbf{Z}_{nj}), \qquad i = 1, \ldots, s \tag{3.14a}$$

$$\mathbf{0} = \mathbf{g}(\mathbf{Y}_{ni}, \mathbf{Z}_{ni}) \qquad i = 1, \ldots, s \tag{3.14b}$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \mathbf{h}\sum_{s}^{i-1} b_i\mathbf{f}(\mathbf{Y}_{ni}, \mathbf{Z}_{ni}), \tag{3.14c}$$

$$\mathbf{0} = \mathbf{g}(\mathbf{y}_{n+1}, \mathbf{z}_{n+1}). \tag{3.14d}$$

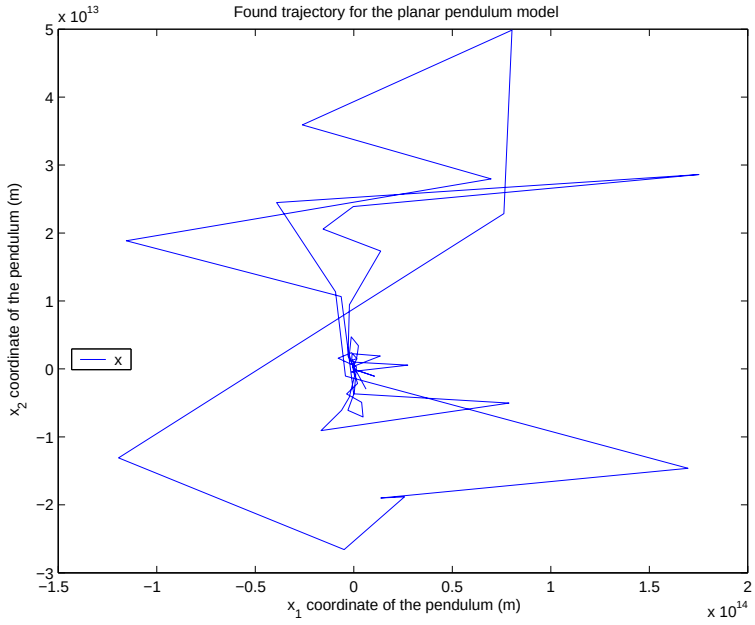

Figure 3.7: *The computed solution for the planar pendulum problem by the coordinate projection method. The least squares minimization is done by 10 steps of the Newton iteration process. The start position of the mass is (1,0). The simulation starts when the mass is in rest.*

In this formulation, $\mathbf{Y}_{ni}$ and $\mathbf{Z}_{ni}$ are variables needed for computing the wanted $\mathbf{y}_{n+1}$ and $\mathbf{z}_{n+1}$.

If $\mathbf{g}_z$ in (3.13) invertible, then the index of (3.13) is 1. In this case, the following sequence of steps gives one step of the numerical method.

Step 1 - Start with $\mathbf{Y}_{n1} = \mathbf{y}_n$

Step 2 - compute $\mathbf{Z}_{n1}$ from (3.14b)

Step 3 - compute $\mathbf{Y}_{n2}$ with (3.14a)

Step 4 - repeat step 2 and 3 till $\mathbf{Y}_{ni}, \mathbf{Z}_{ni}$ are known for $i = 1 \ldots, s$

Step 5 - compute $\mathbf{y}_{n+1}$ with (3.14c)

Step 6 - compute $\mathbf{z}_{n+1}$ from (3.14d).

If $\mathbf{g}$ does not depend on $\mathbf{z}$ and if $\mathbf{g}_y\mathbf{f}_z$ is invertible, then the index of (3.13) is 2, and the above formulation is still applicable. The following sequence of steps defines one time-step of the half-explicit Runge Kutta method.

Step 1 - Start with $\mathbf{Y}_{n1} = \mathbf{y}_n$

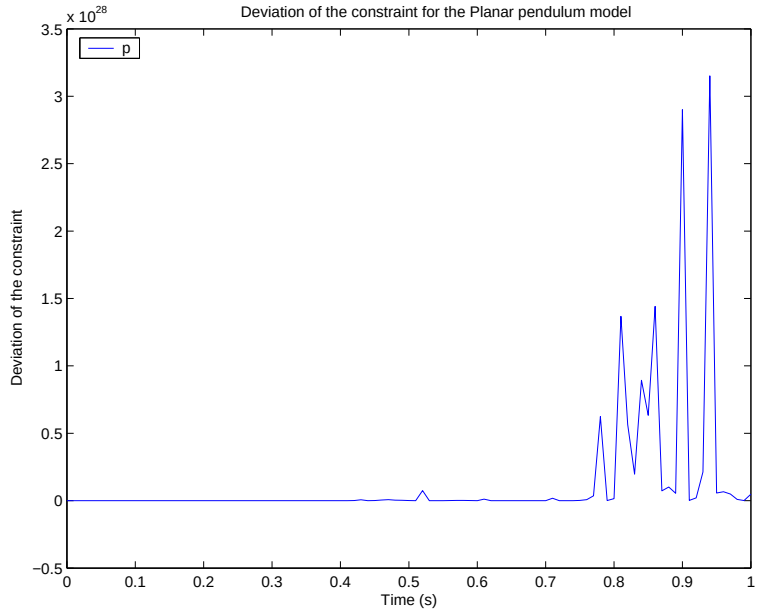

Figure 3.8: *The deviation of the constraint for the planar pendulum problem solved by the coordinate projection method. The least squares minimization is done by 10 steps of the Newton iteration process.*

Step 2 - insert (3.14a) in (3.14b) to find $\mathbf{Z}_{n1}$

Step 3 - compute $\mathbf{Y}_{n2}$ with (3.14a)

Step 4 - repeat step 2 and 3 till $\mathbf{Y}_{ni}, \mathbf{Z}_{ni}$ are known for $i = 1 \ldots, s$

Step 5 - compute $\mathbf{y}_{n+1}$ with (3.14c)

Step 6 - take $\mathbf{z}_{n+1} = \mathbf{Z}_{ns}$.

3.4.1 Applying the Half-Explicit Runge Kutta Methods

To apply a half-explicit Runge Kutta method to the system

$$\begin{aligned} \dot{\mathbf{z}} &= \hat{\mathbf{f}}(\mathbf{z}), & (3.15a) \\ \mathbf{0} &= \mathbf{h}(\mathbf{z}), & (3.15b) \end{aligned}$$

with $\hat{\mathbf{f}}$ and $\mathbf{h}$ as in (3.6) and $\mathbf{z}$ is the state-vector of the system. It is noted that

$$\begin{aligned} \dot{\mathbf{z}} &= \hat{\mathbf{f}}(\mathbf{z}) - D(\mathbf{z})\mathbf{v}, & (3.16a) \\ \mathbf{0} &= \mathbf{h}(\mathbf{z}), & (3.16b) \end{aligned}$$

where $D(\mathbf{z})$ is any bounded matrix function such that HD ($H = \mathbf{h}_z$) is boundedly invertible for all t, has exact the same solution as (3.15). The exact solution

of (3.16) gives $\mathbf{v}(t) \equiv \mathbf{0}$, but this is no longer true in general for a numerical discretization of this system. The choice of the matrix function D defines the direction of the projection onto the constraint manifold. A common choice is $D = H^T$, which yields an orthogonal projection [7]. When trying to apply the half-explicit Runge Kutta method based on the fourth order Runge Kutta method to the system (3.16), one has to solve

$$\mathbf{0} = \mathbf{h}(\mathbf{Y}_{n2}) = \mathbf{h}(\mathbf{y}_n + a_{11}\hat{\mathbf{f}}(\mathbf{Y}_{n1}, \mathbf{Z}_{n1}))$$

for $\mathbf{Z}_{n1}$. In general, it is not easy to find an explicit expression for $\mathbf{Z}_{n1}$. Probable one has to iterate to find an approximation for $\mathbf{Z}_{n1}$. Because of the non-explicit character and the inefficiency of this method for this kind of DAEs, this method is not further pursued. For more on Runge-Kutta methods for DAEs is referred to [18, Chapter 6]

3.5 First Selection of Methods

After describing all the methods, one can see that only the Lagrange Multiplier Methods (Section 3.2) have a low deviation of the constraint. The other methods all have a great deviation of the constraint as results. And more, the other methods show all an unstable behaviour from a certain point in time. Therefore the first conclusion is that the CLMM and DLMM are the most appropriate methods for the models formulated in Chapter 2.

Before a final selection is made, it is noted that the CLMM has error accumulation. Therefore, when once an error is made, this will not be corrected by the method. The DLMM however corrects this. Before a final selection of method is made, both methods are tested on this item.

Before going further, it is noted that all methods are present in the literature. This means that they are useful for other problems as well.

3.6 Further Testing

In this section further testing of the CLMM and the DLMM is done on satisfying the constraint when starting with initial values that not satisfy the constraints. Again the planar pendulum problem model (1.20) is used for testing.

3.6.1 Testing the CLMM

Test objective
Test for the expansion of the deviation of the constraint, when starting with initial conditions that not satisfy the constraints.

Results
In this test the initial conditions are set such that the constraint is not satisfied.

A pendulum usually has a circular motion, in this case with midpoint (0,0). The initial position of the mass is (1.25,0) and the length if the rod is set to $L = 1$.

The results are shown in Figure 3.9 and Figure 3.10. Figure 3.9 shows the found trajectory of the planar pendulum and Figure 3.10 shows the deviation of the constraint in time.

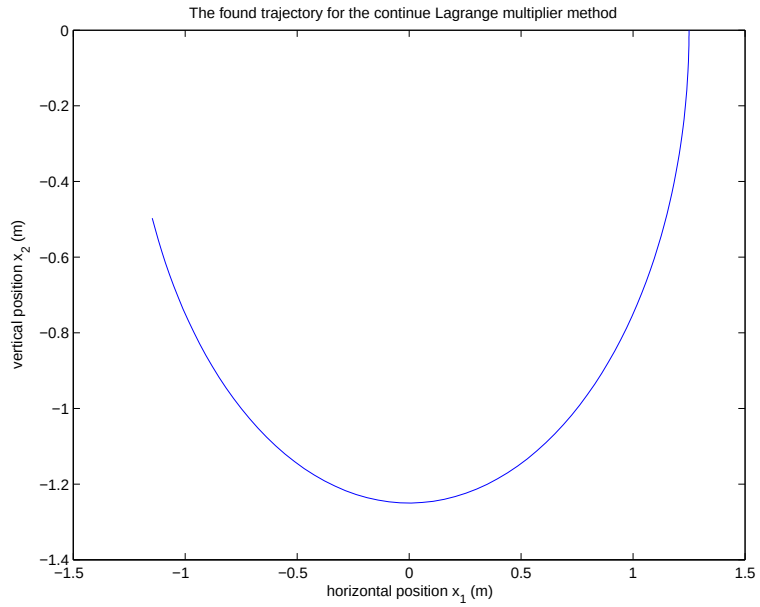

Figure 3.9: *The computed trajectory for the planar pendulum problem (1.20) solved by the CLMM, with initial conditions that not satisfy the constraint. The CLMM is applied as in section 3.2. The initial position is chosen as (1.25,0) and the mass starts from rest.*

Conclusions
As given in the theory one can see that an error once made, will not be corrected, but all errors will be added to each other.

3.6.2 Testing the DLMM

Test objective
Test for the expansion of the deviation of the constraint, when starting with initial conditions that not satisfy the constraints.

Results
In this test the initial conditions are set such that the constraint is not satisfied. A pendulum usually has a circular motion, in this case with midpoint (0,0). The initial position of the mass is (1.25,0) and the length if the rod is set to $L = 1$.

The results are shown in Figure 3.11 and Figure 3.12. Figure 3.11 shows the

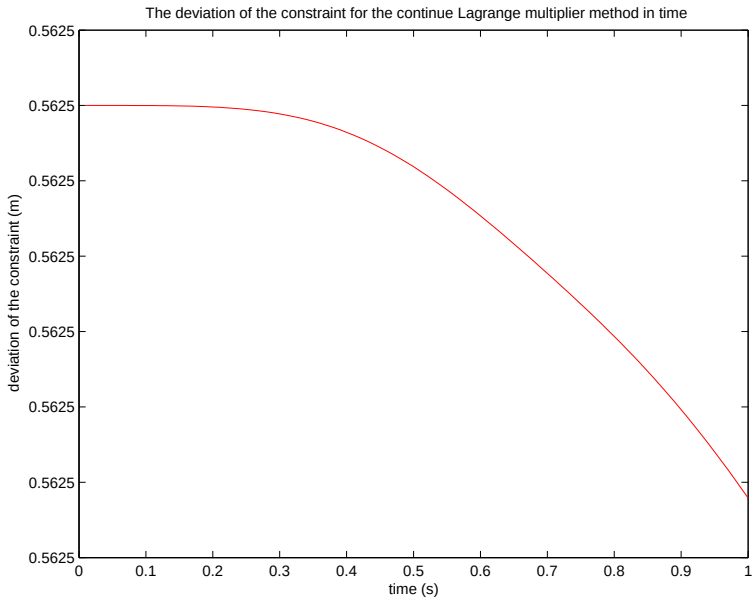

Figure 3.10: *The deviation of the constraint for the planar pendulum problem (1.20) solved by the CLMM, with initial conditions that not satisfy the constraint. The CLMM is applied as in section 3.2. The initial position is chosen as (1.25,0) and the mass starts from rest.*

found trajectory of the planar pendulum problem (1.20) and Figure 3.12 shows the deviation of the constraint in time.

Conclusions
As expected by the theory, the deviation of the constraint will be corrected by the method.

3.7 Final Selection of the Method

After these tests and on base of the test results in section 3.6, a final selection can be done. As suggested by theory and found by the tests, the DLMM is more accurate in handling errors. For this method, there is no error accumulation in the deviation of the constraint, but the DLMM corrects this deviation. Therefore, the DLMM is selected to solve the equations of motion subject to the equality constraints.

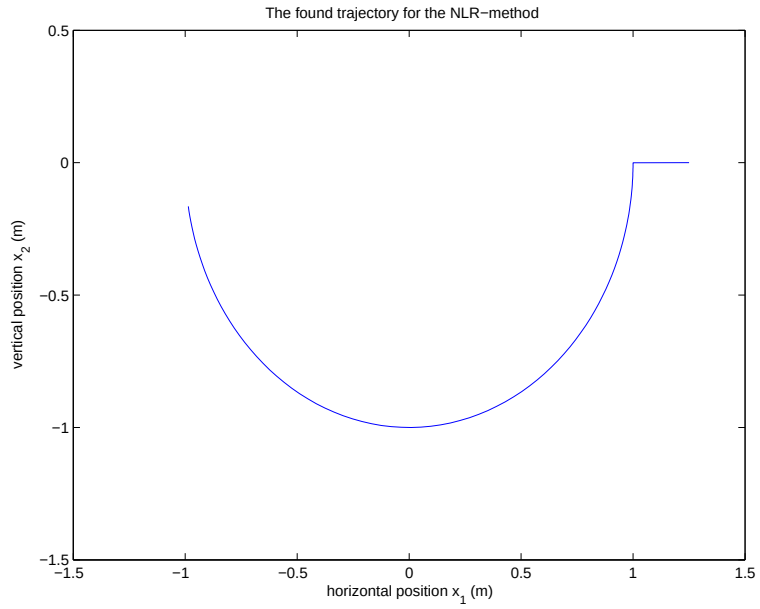

Figure 3.11: *The computed trajectory for the planar pendulum problem (1.20) solved by the DLMM, with initial conditions that not satisfy the constraint. As initial position is taken (1.25,0), so the constraint is not satisfied. The mass starts from rest.*

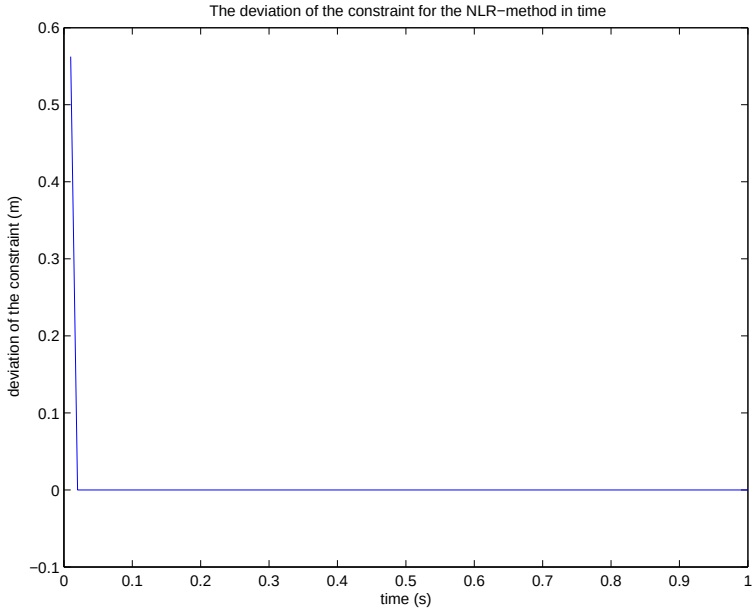

Figure 3.12: *The deviation of the constraint for the planar pendulum problem (1.20) solved by the DLMM, with initial conditions that not satisfy the constraint. As initial position is taken (1.25,0), so the constraint is not satisfied. The initial velocities are taken as zero.*

Chapter 4

Simulating Wheel-Road Contact

4.1 Introduction

A special part of the modeling in this report is the modeling of the wheel-road contact. This means that the inequality constraints (1.7) have to be satisfied. Previous work is done on simulating wheel-road contact, for various road types ([27], [8], [28], [9] and [15]). This research is the basis on which the wheel-road contact simulation is developed. This basis is described in Section 4.2. However, it will appear that this implementation is not realistic enough for simulating wheel-road contact on an arbitrary terrain, the wheel-road contact simulation has to be modified. This will be done in Section 4.3. The algorithm that simulates the wheel-road contact is called the *contact algorithm* and it is described for one partial vehicle in two dimensions.

4.2 The Basis

First, to model wheel-road contact, the partial vehicle is approached as three point masses interconnected by spring damping elements. For simulating wheel-road contact only the lower point mass has to be considered. A number of modes have to be detected. These modes correspond with the possible states subject to the inequality constraints (1.7). After each time step of the DAE-solver that is used, the mode can be different from the previous mode. Therefore, after each time step the contact algorithm is applied and will start with detecting the mode. The possible modes are

1. When the lower point mass on time t_n is above the road, there is nothing to do after this n-th time step of the integration method, because the inequality constraints are satisfied.

2. When on time t_{n-1} the lower point mass is on the road and on time t_n the lower point mass is under or on the road, the inequality constraints (1.7) are violated and the mode rolling is detected. This means that the wheel of this partial vehicle is rolling in the interval $[t_{n-1}, t_n]$. The corrections now needed are discussed in subsection 4.2.1 and after corrections the inequality constraints have to be satisfied.

3. When on time t_{n-1} the lower point mass is above the road and on time t_n the lower point mass is under or on the road again the inequality constraints (1.7) are violated and an impact has to be simulated. The corrections needed are discussed in subsection 4.2.2. And after correction the inequality constraints have to be satisfied.

Before discussing the different modes we remark that the basis contact algorithm described in this section is only appropriate for a flat terrain. The contact algorithm is discussed for one partial vehicle. The partial vehicles is schematically represented by three point masses.

4.2.1 A Rolling Wheel

When the lower point mass in on the road on time t_{n-1} and under or on the road on time t_n the mode is rolling. For this mode the vertical position and velocity of the lower point mass has to be changed. The horizontal position does not need to change and the vertical position of the lower point mass can be determined from the terrain function. The vertical velocity is defined as zero. However this is only true for riding on a flat terrain.

4.2.2 Impact

When an impact occurs much more has to be corrected. When on time t_n an impact is detected, the following sequence of steps has to be done for the wheel contact point (the lower point mass 1).

Step 1, compute impact time t^* - The impact time t^* lies in the interval $[t_{n-1}, t_n]$ and can be given by the following general formula ([27, page 14] and [15, page 12]):

$$t^* = t_{n-1} + \sqrt{\dot{x}_2(t_{n-1})^2 - 2\ddot{x}_2(t_{n-1})(x_2(t_{n-1}) - h(t^*))} \tag{4.1}$$

In this formula $h(t^*)$ is the terrain height on impact time t^*, $x_2(t_{n-1})$ is the vertical position of point mass 1 on time t_{n-1}, $\dot{x}_2(t_{n-1})$ is the velocity of the wheel contact point, $\ddot{x}_2(t_{n-1})$ is the acceleration of the wheel contact point.

Step 2, compute positions for $t = t^*$ - The horizontal position $x(t^*)$ of the wheel contact point on $t = t^*$ can be computed from the formula

$$x(t^*) = x(t_{n-1}) + y(t_{n-1})(t^* - t_{n-1}) + \dot{y}(t_{n-1})(t^* - t_{n-1})^2.$$

In this formula, y denotes the horizontal velocity of the wheel contact point. The vertical impact position is then given by the terrain height.

Step 3, compute velocities just before impact - The velocities just before impact can be compute by the following formula [8, page 15]

$$y_{i_{bc}} = y_i(t_{n-1}) + \dot{y}_i(t^* - t_{n-1}), i = 1, 2 \tag{4.2}$$

In this formula $y_{i_{bc}}$ is the computed velocity of the wheel contact point just before impact, y_i is the velocity of the wheel contact point. and $\dot{y}_i$ is the acceleration ofthe wheel contact point. This formula can be applied to both horizontal ($i = 1$) and vertical ($i = 2$) velocity of the appropriate point mass.

Step 4, compute velocities just after impact - When an impact occurs, the impact in real will not be a completely elastic impact or a completely plastically impact. The vertical velocity would be changed to a velocity in opposite direction, but not with the same value. Because it is not expected that all the kinetic energy will be maintained, an elasticity constant $e_{vert} \in [0, 1]$ which determines which part of the velocity will be maintained, is introduced. The vertical velocity for point mass 1 $y_{vert_{ac}} = -e_{vert}y_{vert_{bc}}$, where $y_{vert_{ac}}$ is the vertical velocity for mass 1 just after impact.

For the horizontal velocity a similar approach is taken. An elasticity constant e_{hor} for the horizontal velocity is introduced and the horizontal velocity $y_{hor_{ac}}$ for mass 1 after impact is be set to $y_{hor_{ac}} = e_{hor}y_{hor_{bc}}$. The value of the elasticity constants depends on physical factors as materials and terrain properties.

Step 5, compute velocities for $t = t_n$ - The velocities at time t_n can be computed by the formula [8, page 15]

$$y_1(t_n) = y_{1_{ac}} + \dot{y}_1(t_n - t^*). \tag{4.3}$$

For a free falling body the term $\dot{y}_1$ is usually $-g$ for the vertical velocity, where g is the gravitational constant $g = 9.81$. It is assumed that the horizontal acceleration is 0, on the interval $[t_{n-1}, t_n]$

Step 6, compute positions for $= t_n$ - The horizontal and vertical position of the wheel contact point on time t_n can be computed by the formula [8, page 15]

$$x_1(t_n) = x_1(t^*) + y_{1_{ac}}(t_n - t^*) + \dot{y}_1(t_n - t^*)^2 \tag{4.4}$$

Step 7, check for second impacts in the same interval - It is possible that after correcting the velocities and positions the wheel contact point is not above the terrain, but due to the gravitational forces again under the terrain. Therefore after correcting the positions and velocities is checked if the corrected positions of the wheel contact point satisfy the inequality constraints (1.7). If not, it is stated that the wheel is going to roll and the wheel contact point is set on the terrain. The position and velocities are set exactly the same as described in section 4.2.1.

To set the wheel rolling in this case is realistic, because then the distance between the wheel contact point and the terrain must be small. An upper bound of the distance between the wheel contact point and the terrain is given by $\frac{1}{2}g\Delta t^2$. This upperbound is the distance that a mass falls within a period of length Δt when the initial vertical velocity is zero.

4.3 The Adapted Contact Algorithm

The contact algorithm has to be modified in order to ride over a non-flat terrain. For the modes *rolling* (Section 4.2.1) and *impact* (Section 4.2.2), the contact algorithm as developed in Section 4.2 is not appropriate enough to simulate wheel-road contact on a non-flat terrain.

The following items in the contact algorithm have to be modified.

- Detecting the mode of the wheel cannot simply be done by comparing position of the wheel contact point with the terrain height. This is because it is unknown where the wheel contact point is. A tyre surrounds the rim and theoretically each point of the tyre can become the wheel contact point. The position of the wheel contact point depends on the slope of the terrain under the wheel.
- For rolling on a slope the vertical velocities cannot simply be set to zero and the position of the wheel contact point is not exactly determined by the equations of motion. The position of the wheel contact point has to be determined dependent on the slope, such that the spring in the tyre acts perpendicular to the terrain.
- If an impact occurs on an arbitrary terrain, again the wheel contact point cannot exactly be determined by the equations of motion. Therefore, the algorithm that simulates an impact on a non-flat terrain has to be modified.

These three points are discussed in the remainder of this chapter.

4.3.1 Detect Mode

To detect the mode, it is not possible to verify the vertical position of the wheel contact point and the height of the terrain on that position. This is because it is unknown where the wheel contact point is. However it is possible to compute an approximation of the radius of the tyre. This is the distance from the rim to the foregoing wheel contact point. The foregoing wheel contact point moves due to the spring forces in the tyre. The whole wheel can be seen as a circle with midpoint the positions of the rim and the radius the distance from the rim to the foregoing wheel contact point. This can be represented in a formula that represents the whole wheel. Let (x_0, y_0) represent the horizontal and vertical

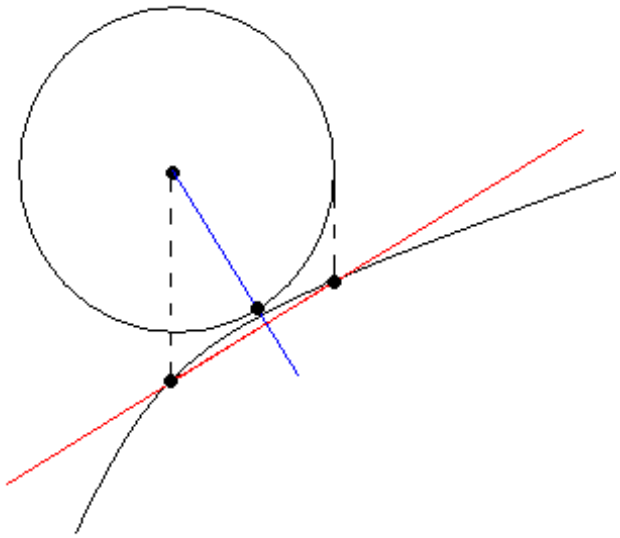

Figure 4.1: *The schematic representation of the linear approximation of the terrain under the wheel contact point and the position of the detected wheel contact point.*

position of the rim and d the distance between the rim and the foregoing wheel contact point, the formula then reads

$$(x - x_0)^2 + (y - y_0)^2 - d^2 = 0. \tag{4.5}$$

If the equation representing the terrain is known, it is possible to detect the mode analytically. Because usually the terrain equation is not known, it is local approximated linearly by $y = ax + b$. This linear approximation is illustrated by Figure 4.1. To detect the mode, the linear approximation of the terrain is substituted in (4.5). The resulting equation becomes

$$(x - x_0)^2 + (ax + b - y_0)^2 - d^2 = 0. \tag{4.6}$$

This is a polynomial of second degree that can be solved analytically. When using the ABC-formula, the discriminant D becomes

$$D = (-2x_0 + 2a(b - y_0))^2 - 4(1 + a^2)(x_0^2 + (b - y_0)^2 - d^2).$$

This information can be used to detect the mode. If on time t_n, one has $D < 0$, then there is nothing to change. The whole wheel is above the terrain. If on time t_{n-1} the wheel contact point is on the terrain and on time t_n one has $D \geq 0$, then the mode rolling is detected, and if on time t_{n-1} the wheel contact point is above the terrain and on time t_n, one has $D \geq 0$, an impact is detected.

4.3.2 Rolling

When the wheel is rolling the wheel contact point has to be found. The wheel contact point is located on the terrain and is located such that the spring in the tyre acts perpendicular to the terrain. When the linear approximation of the terrain under the wheel is $y = ax + b$, the line perpendicular to the terrain and trough the rim (x_0, y_0) can be formulated. This line is then given by the formula $y = 1/ax + c$, where $c = (y_0 + \frac{x_0}{a})$. The intersection of these two lines is set as the wheel contact point. Therefore the horizontal position of the wheel contact point is set to $\frac{c-b}{a-1/a}$ if $a \neq 0$. If $a = 0$ then the horizontal position of the wheel contact point is the same as the horizontal position of the rim x_0. The vertical position of the wheel contact point is now given by the linear approximation of the terrain $y = ax + b$. The velocities of the wheel contact point are determined to be equal to the velocities of the rim. After these computations, the inequality constraints (1.7) are satisfied and a rolling wheel is simulated realistically.

4.3.3 Impact

If the mode impact is detected, the following sequence of steps has to be done to simulate this wheel-road contact. All steps relate to the wheel contact point (point mass 1).

Step 1, compute impact time t^* - To know when exactly the impact was, the terrain height is linear approximated in time on the interval $[t_{n-1}, t_n]$. The vertical position of the future wheel contact point is also linear approximated in time. On the intersection of these two lines, the impact time t^* is found.

Step 2, compute impact position - From the impact time t^*, one can compute the horizontal and vertical position of the wheel contact point. The horizontal position $x(t^*)$ is approximately given by

$$x(t^*) = x(t_{n-1}) + \dot{x}(t_{n-1})(t^* - t_{n-1}) \tag{4.7}$$

and the vertical impact position is given by the linear approximation of the terrain $y = ax + b$.

Step 3, compute velocities just before impact - The velocities just before the impact can be computed by the following formula

$$y_{i_{bc}} = y_i(t_{n-1}) + \dot{y}_i(t^* - t_{n-1}), \quad i = 1, 2. \tag{4.8}$$

In this formula $y_{i_{bc}}$ is the computed velocity of the wheel contact point just before impact and $\dot{y}_i$ is the acceleration of the wheel contact point. This formula can be applied to both horizontal ($i = 1$) and vertical ($i = 1$) velocity of the appropriate point mass.

Step 4, compute velocities just after impact - When an impact occurs, the impact in reality will not be a completely elastic impact or a completely plastically impact. The component of the velocity perpendicular

to the terrain will change to a velocity in opposite direction, but not with the same value. Because, it is not expected that all the kinetic energy will be maintained, elasticity constants e_{vert} and e_{hor} are introduced. The elasticity constant $e_{vert} \in [0, 1]$ determines which part of the velocity perpendicular to the terrain will be maintained. The elasticity constant $e_{vert} \in [0, 1]$ determines which part of the velocity parallel to the terrain will be maintained.

To apply this elasticity constants to the velocities of the wheel contact point just before impact, the factorisation of the velocities parallel with the terrain and perpendicular to the terrain has to be performed. The following steps can do this factorisation. The angle β is the angle of the terrain, computed as $\beta = arctan(a)$ where a is the tangent of the linear approximation of the terrain. The vertical and horizontal velocities in the original orientation are denoted by y_{vert} and y_{hor}. The vertical and horizontal velocities in the new orientation are denoted by yp_{vert} and yp_{hor}.

$$\begin{aligned} nr_1 &= y_{hor}\cos(\beta) \\ nr_2 &= y_{vert}\sin(\beta) \\ nr_3 &= -y_{hor}\sin(\beta) \\ nr_4 &= y_{vert}\cos(\beta) \\ yp_{vert} &= nr_1 + nr_2 \\ yp_{hor} &= nr_3 + nr_4 \end{aligned}$$

Applying the elasticity constants gives for the velocities just after the impact

$$\begin{aligned} yp_{vert} &= -e_{vert} \quad yp_{vert} \\ yp_{hor} &= e_{hor} yp_{hor} \end{aligned}$$

The factorisation back to the original orientation can be done by the following steps

$$\begin{aligned} nr_5 &= yp_{hor}\cos(\beta) \\ nr_6 &= -yp_{vert}\sin(\beta) \\ nr_7 &= yp_{hor}\sin(\beta) \\ nr_8 &= yp_{vert}\cos(\beta) \\ y_{hor_{ac}} &= nr_5 + nr_6 \\ y_{vert_{ac}} &= nr_7 + nr_8 \end{aligned}$$

The resulting $y_{hor_{ac}}$ and $y_{vert_{ac}}$ are the horizontal and vertical velocities of the wheel contact point just after the impact.

Step 5, compute velocities on time t_n - The velocities at time t_n can be computed by the formula

$$y_i(t_n) = y_{i_{ac}} + \dot{y}_i(t_n - t^*), i = 1, 2. \tag{4.9}$$

For a free falling body the term $\dot{y}_i$ is usually $-g$ for the vertical velocity ($i = 2$), where g is the gravitational constant ($g = 9.81$).

Step 6, compute positions on time t_n - The positions of the wheel contact point on time t_n can be computed by the formula

$$x_i(t_n) = x_i(t^*) + y_{i_{ac}}(t_n - t^*) + \dot{y}_i(t_n - t^*)^2, i = 1, 2. \tag{4.10}$$

Step 7, check for second impact in the same interval - It is possible that after correcting the velocities and positions the wheel contact point is not above the terrain, but due to the gravitational forces again under the terrain. Therefore after correcting the positions and velocities is checked if the corrected positions of the wheel contact point satisfy the inequality constraints (1.7). If not, it is stated that the wheel is going to roll and the wheel contact point is set on the terrain. The position and velocities are set exactly the same as described in Section 4.3.2.

To set the wheel rolling in this case is realistic, because then the distance between the wheel contact point and the terrain must be small. An upper bound of the distance between the wheel contact point and the terrain is given by $\frac{1}{2}g\Delta t^2$.

After the execution of the contact algorithm the inequality constraints (1.7) are satisfied.

4.3.4 Remarks

After this all it is noted that the contact algorithm has fundamentally changed. The wheel contact point cannot longer be modeled as an independent point mass. The location of the wheel contact point is dependent of the terrain and the position of the rim. Forces defined on the wheel contact point are not taken into account when the mode of the partial vehicle is rolling or impact. This means, that when one wants to model an accelerating motorcycle, he has to apply the external horizontal force on the rim instead of the wheel contact point.

Chapter 5

Implementing the Behaviour Model in MATLAB/Simulink

The behaviour model for a motorcycle in two dimensions (horizontal and vertical) is implemented in MATLAB/Simulink. In this model it is not possible to steer the motorcycle. The behaviour model consists of the equations (2.6), (2.7) and the inequality constraints

$$\begin{aligned} x_4 > x_2 &\geq h(x_1), \\ x_{10} > x_8 &\geq h(x_7), \\ x_4 &< x_6, \\ x_{10} &< x_{12}. \end{aligned}$$

The top level of the Simulink model is represented in Figure 5.1. In the appendix, a manual is given to handle and change the model. First some basic information about MATLAB, Simulink and the MATLAB compiler is given in Section 5.1 and then Simulink model and the subsystems is described.

5.1 The MATLAB Compiler and Simulink S-functions

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation.

Simulink has become the most widely used software package in academia and industry for modeling and simulating dynamic systems. In Simulink one can

easily build models from scratch, or take an existing model and add to it. Simulations are interactive, therefore it is possible to change parameters on the fly and immediately see what happens. One has instant access to all the analysis tools in MATLAB, therefore the results can taken, analyzed and visualized. A Simulink model is built with blocks. Many blocks are predefined, but it is also possible to build blocks, for instance by the use of Simulink S-functions, which is done. A Simulink S-function is a computer language description of a Simulink block. S-functions can be written in MATLAB, C, C++, Ada, or Fortran. In the implementation of the behaviour model of the vehicle, the S-function are written in C.

The MATLAB Compiler takes M-files as input and generates C or C++ source code or P-code as output. The MATLAB Compiler can generate various kinds of source code for instance C code S-functions for use with Simulink and C shared libraries (dynamically linked libraries, or DLLs, on Microsoft Windows) and C++ static libraries.

With these tools it is possible to program all in M-files and use the MATLAB Compiler to convert it to Simulink C Mex S functions. This is done because a Simulink model is wanted and the C mex S functions can be invoked in Simulink. This Simulink model can be converted for real-time simulating purposes by an especially developed tool of the NLR, called MOSAIC [21].

5.2 The Simulink Model

The Simulink model is represented in Figure 5.1. The main parts of the model are the four colored blocks. The yellow block "Equations of motion and DLMM" contains the subsystem where the forces acting on the motorcycle are computed (discussed in Section 5.3), the red one, "Terrain information", contains the subsystem where the needed terrain information is computed (discussed in Section 5.4) and the green block, "Contact algorithm", represents the subsystem where the contact algorithm is applied to the computed state vector (discussed in Section 5.5).

The integrator block can be seen as the center of the model. There, on base of the output of the subsystem "Equations of motion and DLMM", the next state-vector is computed by the specified numerical method. The model starts with the initial condition given by the IC-block right under. The state for the following time-step is computed in the integrator block. The state port copies the computed state and sends it to the subsystems Terrain information and Contact. The contact algorithm, presented in Chapter 2 corrects the state, subject to the terrain information obtained from the red subsystem. The corrected state is sent to the initial condition port of the integrator block. The Pulse Generator and the Constant block are designed to reset the state of the integrator block each time-step to the corrected state. Then the next time-step can start.

The Unit Delay block in the model is set to avoid a loop. The initial condition of this block is the same as the initial condition of the whole system. Right under in the model a Selector block, a scope block and a block, called To Workspace,

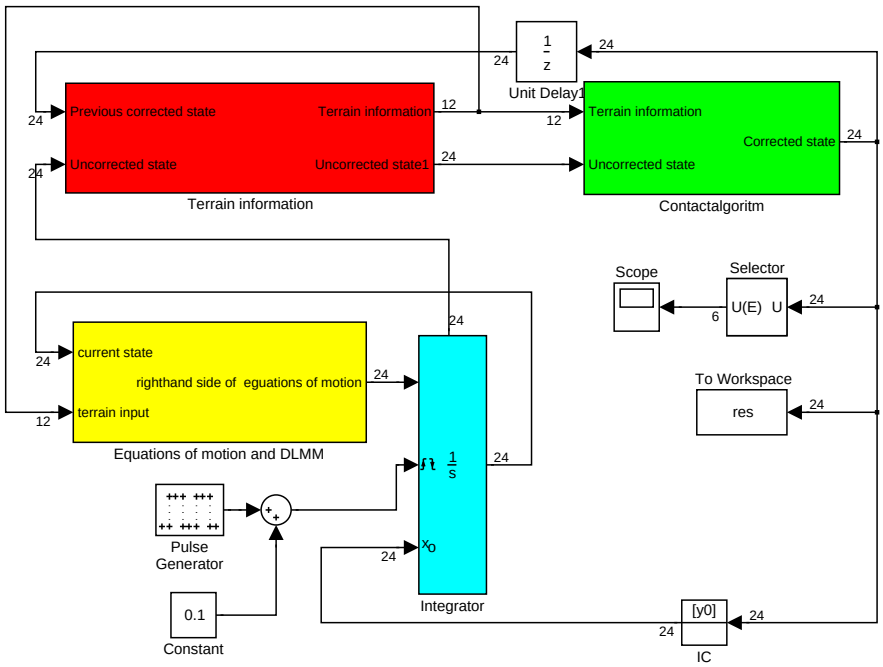

Figure 5.1: *The top-level of the Simulink model of the vehicle.*

are found. The Selector selects the vertical positions computed by the model and represents them in the scope. All output needed for application in training simulators can be obtained from the complete set of data that is sent to the workspace in an array. Changing parameters can easily be done by changing them in the masks of the subsystems.

5.3 Subsystem "Equations of motion and DLMM"

In this subsystem the forces on the masses and the equations of motion are computed. The subsystem is represented in Figure 5.2.

The input of this subsystem consists of the output of the integrator block and some terrain information. This terrain information is needed to determine whether friction has to be applied or not. This determination is performed in the S-function Friction. This S-function gets the previous state and the terrain information as input and detect if the wheel contact points are on the terrain or not. A two-dimensional vector of booleans, which corresponds to the different wheel contact points, gives the output of this S-function.

Many other sources are needed for computing the forces and the right hand side of the equations of motion and the needed computations for DLMM. All the sources are found on the left side of the multiplexer block. The S-function "Motoreq" performs the main of the work in this subsystem. This S-function

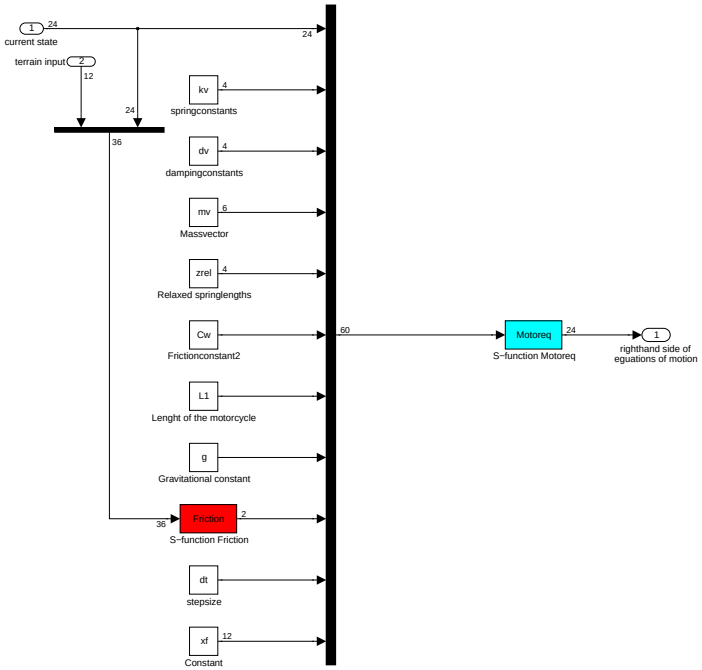

Figure 5.2: *The schematic representation of the subsystem "Equations of motion and DLMM"*

computes the desired output of the subsystem.

5.4 Subsystem "Terrain information"

In this subsystem the terrain information needed for the subsystems "Contact" and "Equations of motion and DLMM" is computed. Therefore the terrain is defined in this subsystem. The subsystem is represented in Figure 5.3.

The input of this subsystem consists of the computed state and the previous state after correction by the contact algorithm. Simulation time and step-size are also necessary for the computation of the needed terrain information. The main task of this subsystem is again performed by a C mex S-function. This S-function defines the terrain properties and, based on these terrain properties and subject to the computed and previous state, it computes for each wheel a linear approximation of the terrain height. A linear approximation of the terrain height in time, under the future wheel contact point of each wheel is also computed. Finally it computes the terrain height under the wheel contact points during t_n and t_{n-1}.

This S-function is obtained from running the MATLAB Compiler on a MATLAB M-file. Three terrain functions are present: a flat plane, a rising plane, and a hill. For more information about the available terrain functions is referred to

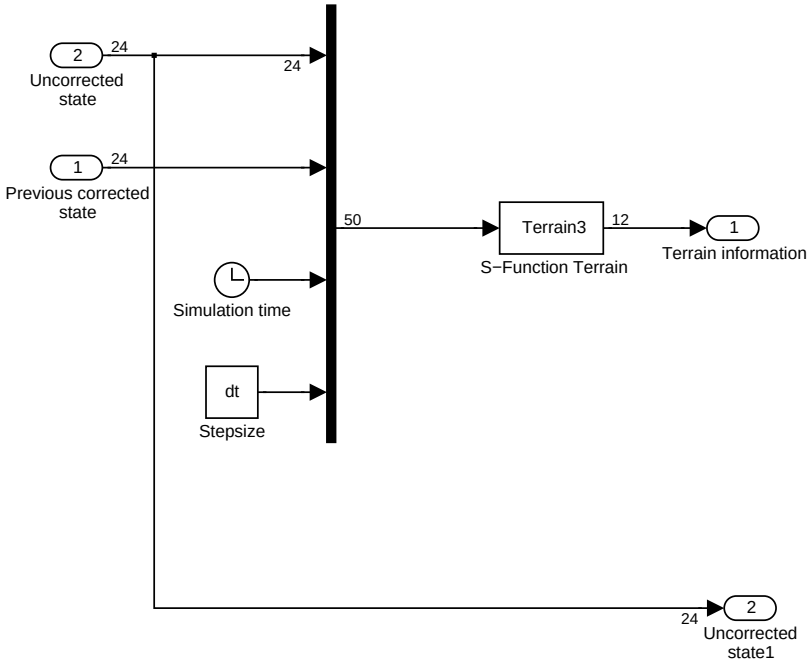

Figure 5.3: *The schematic representation of the subsystem "Terrain information"*

the appendix.

5.5 Subsystem "Contact"

The schematic representation of this subsystem is given in Figure 5.4 In this subsystem, the contact algorithm developed in Chapter 4 is implemented. The input of this subsystem consists of the computed state and the previous state after correction. Other inputs are the vertical and horizontal elasticity constants and the terrain information as described in Section 5.4.

All this information is sent to the S-function Contact which performs all the work to correct the state according to the algorithm as developed in Chapter 4. This S-function is also obtained from a MATLAB M-file by running the MATLAB Compiler. The S-function first detects the input and then first it detect the state for the wheel on the backside of the motorcycle. After the detection of the case and the correcting of the state corresponding to this state, the S-function performs the same for the wheel at the front side of the motorcycle.

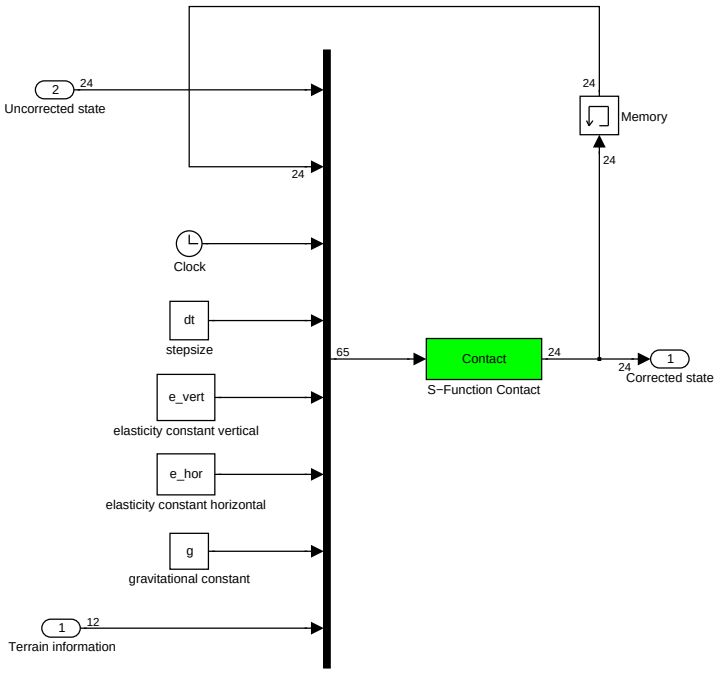

Figure 5.4: *The schematic representation of the subsystem "Contact"*

Chapter 6

Testing the Behaviour Model

6.1 Introduction

In Chapter 2 the behaviour model for a four-wheeled vehicle is developed. In Chapter 3 a numerical method that can solve the behaviour model of such a vehicle is selected. Chapter 4 contains the development of the contact algorithm for a partial vehicle in two dimensions. The behaviour model of a two-part vehicle in two dimensions, such as a motorcycle, is implemented in MATLAB/Simulink (Chapter 5). The two dimensions include that with this model one can simulate the riding of a motorcycle over obstacles. However, steering is not possible. In this chapter several tests are described.

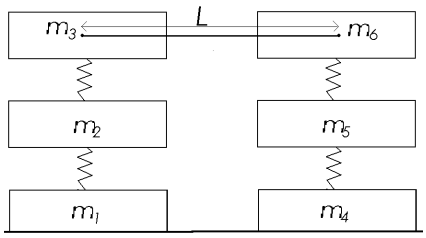

Figure 6.1: *The motorcycle model consisting of two partial vehicles connected by a rod of length L.*

First, the schematical representation of the motorcycle is given in Figure 6.1. During the development of the behaviour model of the behaviour model of a multiple-part vehicle two behaviour models were developed. *Model 1* is the

behaviour model consisting of two copies of (2.3) and the constraint (2.4). *Model 2* consists of two copies of (2.6) and the constraints (2.7).

In Chapter 2 it is already stated that model 1 is not sufficiently realistic. In Section 6.2 three basic tests are done with model 1 to illustrate why model 1 is not realistic enough. The same tests as in Section 6.2 are done in Section 6.3, but now for model 2. The results of the tests are used to compare model 1 and model 2. This comparison is done in Section 6.4. Model 2 is further tested in Section 6.5. A conclusion in Section 6.6 completes the chapter.

All tests in this chapter are done in MATLAB with the use of M-files. The Discrete Lagrange Multiplier Method (DLMM), described in Chapter 3 is used for solving the DAEs and the used ODE-solver is the fourth-order Runge Kutta method. In all tests the length of the motorcycle is $L = 1$ (See Figure 6.1). All other parameters vary and are given in the test descriptions.

6.2 Testing Model 1

In this section, model 1 is tested. Three basic tests are done and described. The results of the tests are figures that represent the found trajectories of the point masses $m_i, i = 1, \ldots, 6$, or sometimes the deviation of the constraints. For model 1, the horizontal position of the point masses $m_i, i = 1, 2, 3$ is represented by x_1, for the point masses $m_i, i = 4, 5, 6$ the horizontal position is represented by x_5. The vertical position of point mass m_1, is denoted by x_2, for point mass m_2 it is x_3, for point mass m_3 it is x_4, for point mass m_4 it is x_6, for point mass m_5 it is x_7, and for point mass m_6, the vertical position is denoted by x_8.

6.2.1 Test 1 Going to the Equilibrium State

Test objective
Test the stability and behaviour of this model when the model does not start in the equilibrium state.

Description
This test consists of two parts. The description of the first part is given first.

The initial conditions are set such that the vertical positions of the point masses 3 and 6 (see Figure 6.1) are the same, and the vertical positions of the point masses 2 and 5 are the same. The most bottom point masses 1 and 4 are always on the ground. Therefore, their vertical position is 0. The following initial conditions are taken.

i	position	velocities	i	position	velocities
1	(0,0)	(0,0)	4	(1,0)	(0,0)
2	(0,3)	(0,0)	5	(1,3)	(0,0)
3	(0,4)	(0,0)	6	(1,4)	(0,0)

The parameters are as follows.

- Mass of point mass i is $m_i = 1$ kg, for $i = 1, 2, 3, 4$.
- Relaxed spring lengths $z_{rel_{v_i}} = 0.4$ for $i = 1, 2, 3, 4$.
- Spring constant k_{v_i} of spring i is $k_{v_i} = 1000$ for $i = 1, 2, 3, 4$.
- Damping constant d_{v_i} of spring i is $d_{v_i} = 0$ for $i = 1, 2, 3, 4$.
- The time-interval is $t \in [0, 1]$.
- The step size is chosen as $\Delta t = 0.01$.

A periodic behaviour like a sine wave is expected.

In the second part of this test the damping constants are chosen as $d_{v_i} = 50$. For this test a damped periodic behaviour is expected.

Results
Figure 6.2 shows the results for part 1 of this test. This figure shows the vertical positions of the point masses of model 1 in time. As expected, a certain periodic behaviour is visible. The point masses m_3 and m_6 do not behave as a sine wave. They are also influenced by the lower springs. This causes it not to be a sine-wave like motion. The time plots of the vertical positions of m_1 and m_4, m_2 and m_5 and m_3 and m_6 are the same. This is to be expected since the initial conditions were chosen this way. This also yields that the constraint (2.4) is not violated.

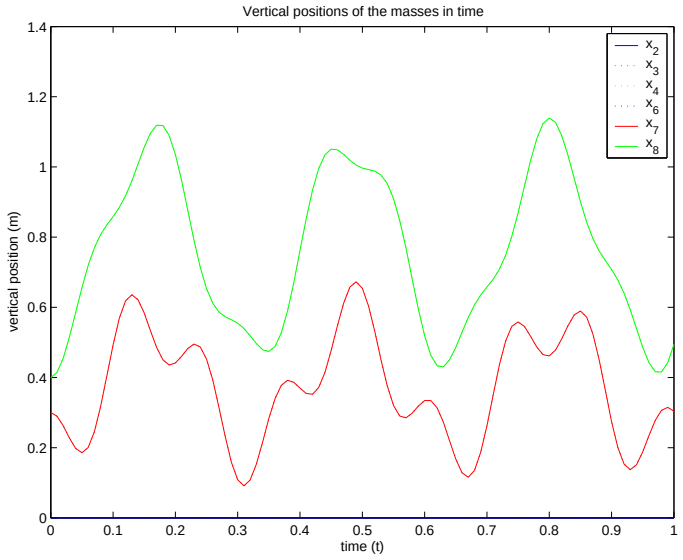

Figure 6.2: *The vertical positions of the point masses in model 1 in time, for part 1 of test 1. The values for x_2 and x_5, x_3 and x_7 and x_4 and x_8 coincide.*

The results of the second part are represented in Figure 6.3. Again the figure shows the vertical positions of the point masses in time. In this figure a periodic damping behaviour, just what was expected.

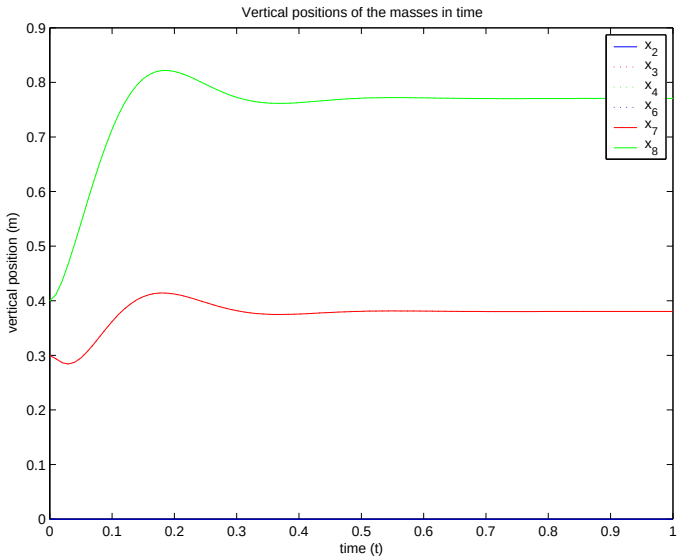

Figure 6.3: *The vertical positions of the point masses in model 1 in time, for part 2 of test 1. The values for x_2 and x_5, x_3 and x_7 and x_4 and x_8 coincide.*

Conclusions
Based on this test, it follows that the model remains stable with realistic behaviour when it does not start in the equilibrium state.

6.2.2 Test 2 Applying an External Vertical Force

Test objective
Test how the constraint (2.4) will be satisfied when applying an external vertical force to the model in a nearly stable state. The stability and behaviour properties are also taken into account.

Description
For this test, from $t = 1$ an external vertical force of $100N$ is applied on point mass 6. The following initial conditions are taken

i	position	velocities	i	position	velocities
1	(0,0)	(0,0)	4	(1,0)	(0,0)
2	(0,3)	(0,0)	5	(1,3)	(0,0)
3	(0,4)	(0,0)	6	(1,4)	(0,0)

The parameters are as follows.

- Mass of point mass i is $m_i = 1$ kg, for $i = 1, 2, 3, 4$.
- Relaxed spring lengths $z_{rel_{v_i}} = 0.4$ for $i = 1, 2, 3, 4$.
- Spring constant k_{v_i} of spring i is $k_{v_i} = 1000$ for $i = 1, 2, 3, 4$.

- Damping constant d_{v_i} of spring i is $d_{v_i} = 50$ for $i = 1, 2, 3, 4$.
- The time-interval is $t \in [0, 3]$.
- The step size is chosen as $\Delta t = 0.01$

It is expected that the front of the motorcycle (the side which includes point mass 6) moves down wards while the backside remains its vertical position.

Results
In the figures 6.4 and 6.5 the results are presented. Figure 6.4 presents the vertical positions of all point masses in the model. Figure 6.5 represents the deviation of the constraint for this case.

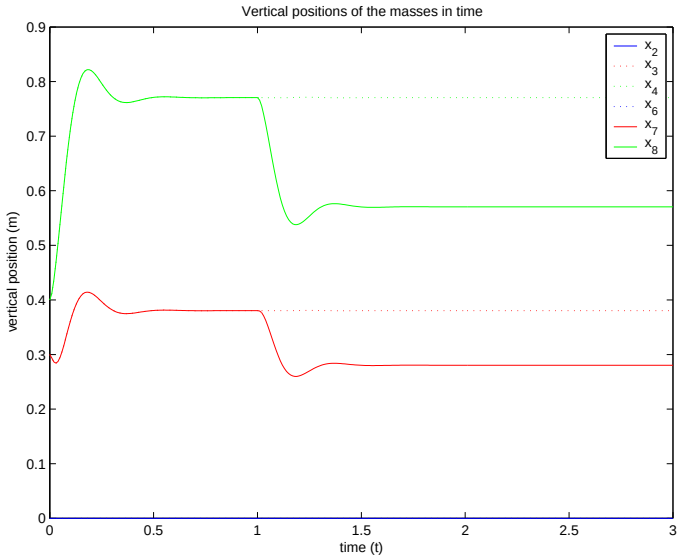

Figure 6.4: *The vertical positions of the point masses in model 1 for test 2. The values for x_2 and x_5, x_3 and x_7 and x_4 and x_8 coincide.*

First in Figure 6.4, from $t \in [0, 0.5]$, it is seen that the model turns into the equilibrium state. From $t = 1$, the extra vertical force is applied and it causes a downward movement of first point mass m_6 and this causes a downward movement of m_5. In Figure 6.5 the deviation of the constraint in time is represented. As can be seen in this figure, from $t = 1$, there is a little deviation, but it is corrected in short time. This demonstrates the power of the DLMM in satisfying the constraints.

Conclusions
From this test, it can be concluded that the constraint is satisfied and that the point masses m_6 and m_5 move downward. The stability and behaviour properties in this test are realistic.

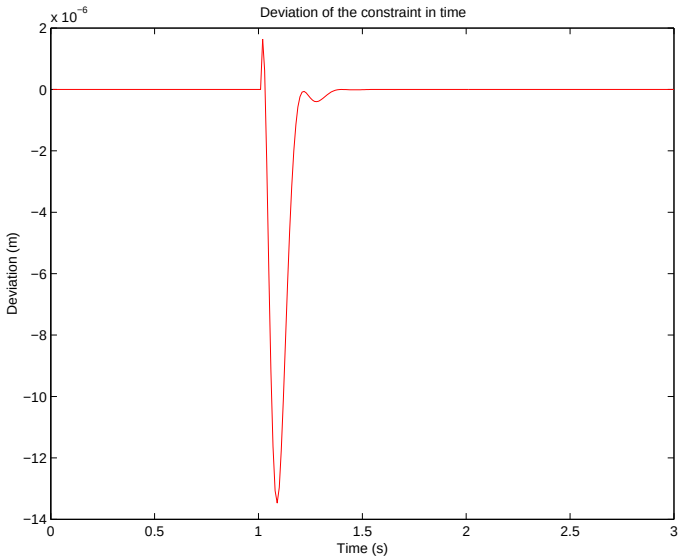

Figure 6.5: *The deviation of the constraint (2.4) in time for model 1 in test 2.*

6.2.3 Test 3 Accelerating and Decelerating

Test objective
Test the model behaviour when applying external horizontal forces on point mass 1, which represent acceleration and deceleration forces.

Description
For $t \in [0, 1]$, an external horizontal force of $100N$ is applied to point mass 1, which represents acceleration. For $t \in [2, 3]$ an external horizontal force of $-100N$ is applied to point mass 1, which represents deceleration. The following initial conditions are taken.

i	position	velocities	i	position	velocities
1	(0,0)	(0,0)	4	(1,0)	(0,0)
2	(0,3)	(0,0)	5	(1,3)	(0,0)
3	(0,4)	(0,0)	6	(1,4)	(0,0)

The parameters are

- Mass of point mass i is $m_i = 1$ kg, for $i = 1, 2, 3, 4$.
- Relaxed spring lengths $z_{rel_{v_i}} = 0.4$ for $i = 1, 2, 3, 4$.
- Spring constant k_{v_i} of spring i is $k_{v_i} = 1000$ for $i = 1, 2, 3, 4$.
- Damping constant d_{v_i} of spring i is $d_{v_i} = 50$ for $i = 1, 2, 3, 4$.
- The time-interval is $t \in [0, 1]$.
- The step size is chosen as $\Delta t = 0.001$

During acceleration, it is expected that the front of the motorcycle (represented by the point masses $4, 5$ and 6) becomes higher than the backside. During deceleration the front has to become lower than the backside.

Results
The angle of the orientation of the chassis subject to the terrain is given in Figure 6.6. Figure 6.7 shows the vertical positions of the point masses in the model.

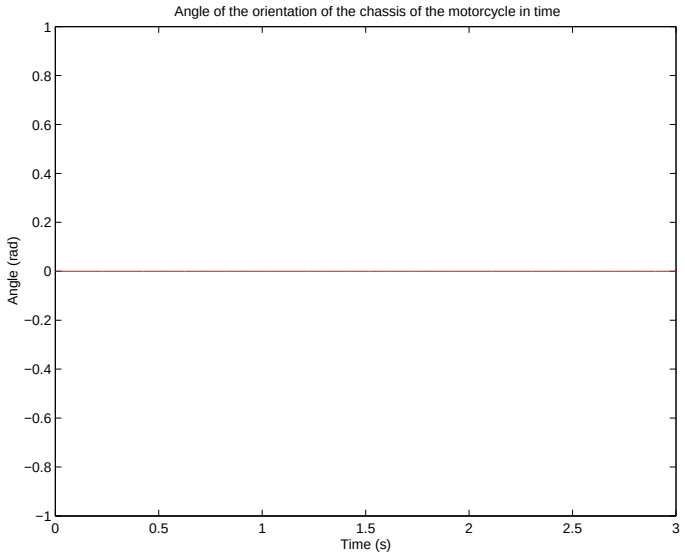

Figure 6.6: *The angle of the orientation of the chassis of the motorcycle subject to the terrain for model 1 in test 3.*

As can be seen the angle does not change either when accelerating or decelerating. This is not the expected behaviour. From Figure 6.7 it is seen that the vertical positions move to an equilibrium state and do not react to acceleration or deceleration.

Conclusions
From the results of this test it has to be concluded that model does not have the desired behaviour properties. The motorcycle is riding as a result of the external horizontal force, but the front of the motorcycle does not go up wards due to the horizontal force on point mass 1. And the front does not go down when decelerating.

6.3 Testing Model 2

In this section, model 2 is tested. The same tests as in section 6.2 are performed. The tests are performed with the completely developed model, including the contact algorithm as described in section 4.3. This means that though realistic

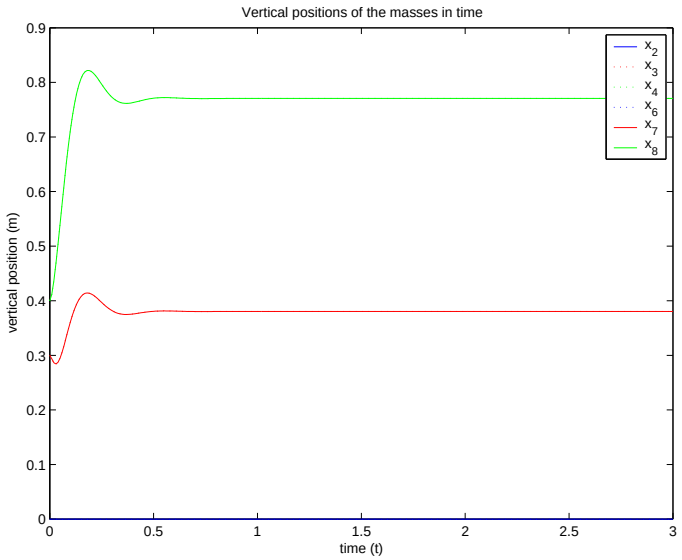

Figure 6.7: *The vertical positions of the point masses in model 1 for test 3. The values for x_2 and x_5, x_3 and x_7 and x_4 and x_8 coincide.*

behaviour for some test cases for model 1, the results in the same cases for this model can differ. The tests performed for model 1 are performed without a contact algorithm implemented. To keep the lower point masses on the ground for model 1, the lower vertical positions of the lower point masses are set to zero. To keep the lower point masses on the ground for model 2, the initial conditions are chosen such that this is nearly the case.

The results of the tests are figures that represent the found trajectories of the point masses $m_i, i = 1, \ldots, 6$, or sometimes the deviation of the constraints. For model 1, the horizontal position of point mass m_i is represented by $x_{2i-1}, i = 1, \ldots, 6$, the vertical position of point mass m_i is represented by $x_{2i}, i = 1, \ldots, 6$.

6.3.1 Test 1 Going to an Equilibrium State

Test objective
Test the stability and behaviour of this model when the model does not start in the equilibrium state.

Description
This test consists of two parts. The description of part 1 is given first.

The initial conditions are set such that the vertical positions of the point masses 3 and 6 are the same, and the vertical positions of the point masses 2 and 5 are the same. The most bottom point masses 1 and 4 are always on the ground. Therefore, their vertical position is 0. The following initial conditions are taken.

i	position	velocities	i	position	velocities
1	(0,0)	(0,0)	4	(1,0)	(0,0)
2	(0,3588)	(0,0)	5	(1,3588)	(0,0)
3	(0,7392)	(0,0)	6	(1,7392)	(0,0)

The parameters are as follows.

- Mass of point mass i is $m_i = 1$ kg.
- Relaxed spring lengths $z_{rel_{v_i}} = 0.4$ for $i = 1, 2, 3, 4$.
- Spring constant k_{v_i} of spring i is $k_{v_i} = 500$ for $i = 1, 2, 3, 4$.
- Damping constant d_{v_i} of spring i is $d_{v_i} = 0$ for $i = 1, 2, 3, 4$.
- The elasticity constants for the contact algorithm are both chosen as 0.9.
- Friction constant is chosen as $c_w = 0$.
- The time-interval is $t \in [0, 3]$.
- The step size is chosen as $\Delta t = 0.001$.

A periodical a sine wave like behaviour is expected.

In the second part of this test, the spring constants are chosen as $k_{v_i} = 500$ for $i = 1, 2, 3, 4$, the damping constants as $d_{v_i} = 50$ for $i = 1, 2, 3, 4$ and the initial conditions as

i	position	velocities	i	position	velocities
1	(0,0)	(0,0)	4	(1,0)	(0,0)
2	(0,35)	(0,0)	5	(1,35)	(0,0)
3	(0,7)	(0,0)	6	(1,7)	(0,0)

All other parameters and settings are the same as for part 1.

A damped periodic behaviour is expected.

Results
Figure 6.8 represents the vertical positions of the point masses of model 2 in part 1 of this test. Figure 6.8 represents the vertical positions of the point masses of model 2 in part 2. For both figures only the upper four point masses are represented. This is done for better representing the periodical behaviour, because of low amplitude.

As one can see, Figure 6.8 represents a periodical behaviour nearly like a sine wave. It has to be noted that the initial conditions are chosen very near to the equilibrium heights, otherwise the lower point masses were not on the terrain, during simulation. The results of part 2 represented in Figure 6.9 show a damping behaviour. For $t \in [0, 1]$ a damped swing is visible and then the point masses are nearly on their equilibrium.

Conclusions
Both, Figure 6.8 and Figure 6.9 show the expected and realistic behaviour. It

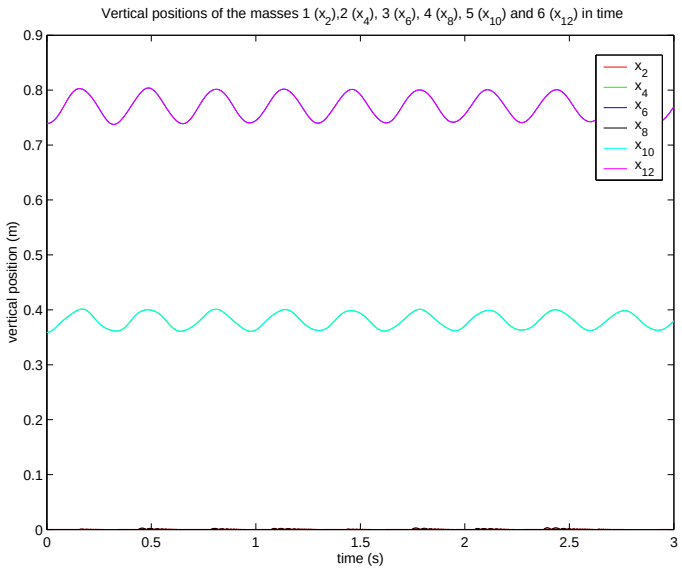

Figure 6.8: *The vertical positions of the upper four point masses in model 2 for test 1, part 1. The values for x_4 and x_{10}, and for x_6 and x_{12} coincide.*

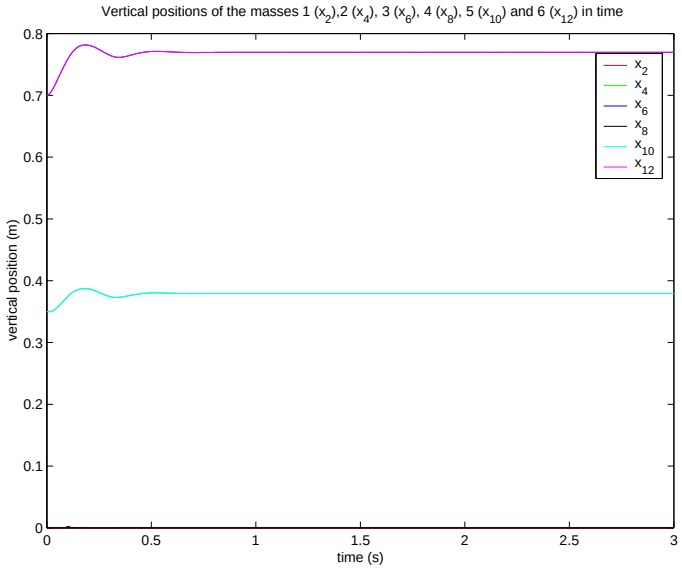

Figure 6.9: *The vertical positions of the upper four point masses in model 2 for test 1, part 2. The values for x_4 and x_{10}, and for x_6 and x_{12} coincide.*

can be concluded from this test that the model behaves realistic when the model does not start in the equilibrium state.

6.3.2 Test 2 Applying an External Vertical Force

Test objective
Test how the length constraint (2.7a) will be satisfied when applying an external vertical force to the model in a nearly stable state. The stability and behaviour properties of the model are also taken into account.

Description
From $t = 0.5$ an external vertical force of $100N$ is applied on point mass 6. The following initial conditions are taken

i	position	velocities	i	position	velocities
1	(0,0)	(0,0)	4	(1,0)	(0,0)
2	(0,3588)	(0,0)	5	(1,3588)	(0,0)
3	(0,7392)	(0,0)	6	(1,7392)	(0,0)

The parameters are as follows.

- Mass of point mass i is $m_i = 1$ kg, for $i = 1, 2, 3, 4$.
- Relaxed spring lengths $z_{rel_{v_i}} = 0.4$ for $i = 1, 2, 3, 4$.
- Spring constant k_{v_i} of spring i is $k_{v_i} = 1000$ for $i = 1, 2, 3, 4$.
- Damping constant d_{v_i} of spring i is $d_{v_i} = 50$ for $i = 1, 2, 3, 4$.
- The elasticity constants for the contact algorithm are both chosen as 0.9.
- Friction constant is chosen as $c_w = 0$.
- The time-interval is $t \in [0, 1]$.
- The step size is chosen as $\Delta t = 0.001$.

It is expected that the front of the motorcycle (the side which includes point mass 6) moves down wards while backside remains its vertical position.

Results
The figures 6.10, 6.11 and 6.12 present the results. Figure 6.10 shows the vertical positions of the point masses in the model. Figure 6.11 the horizontal positions of the point masses in the model. And Figure 6.12 shows the deviation of the length constraint in time.

Figure 6.10 shows that for $t \in [0, 0.5]$ the motorcycle moves to the equilibrium state. From $t = 0.5$ the external vertical force is applied and then it is seen that the front of the motorcycle moves down wards, while the backside of the motorcycle keeps nearly the same value. From $t = 0.8$ the model is going to a new equilibrium state. From Figure 6.11 it is seen that for $t \in [0, 0.5]$ nothing is changing and from $t = 0.5$, when the external vertical force is applied, the horizontal positions of the upper point masses move forward while the other point masses move backward. From Figure 6.12 it can be seen that due to the external vertical force the length constraint obtains a deviation of $1.5 \cdot 10^{-7}$, and in 0.4 seconds, this deviation is vanished.

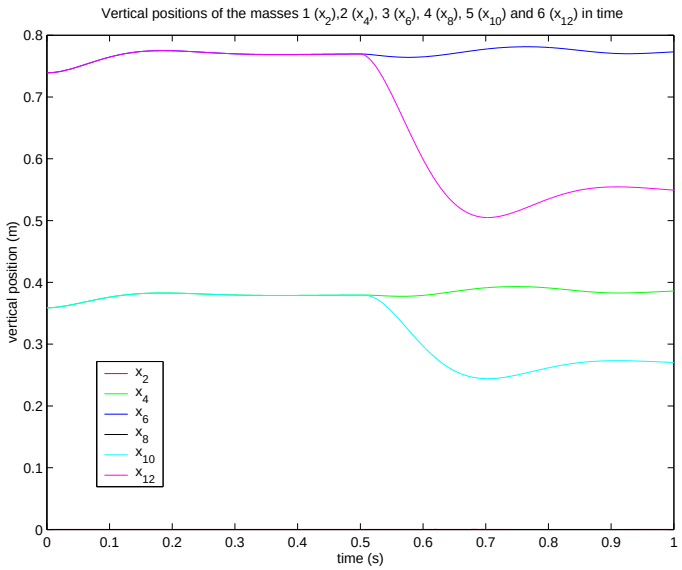

Figure 6.10: *The vertical positions of the point masses in model 2 for test 2.*

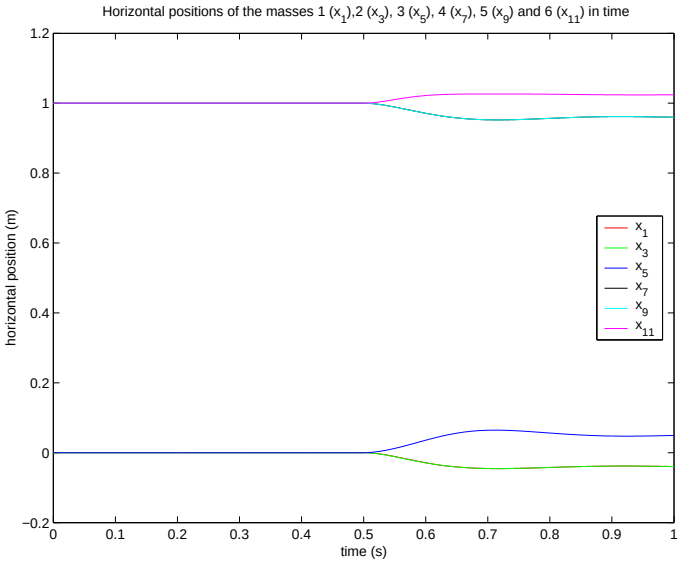

Figure 6.11: *The horizontal positions of the point masses in model 2 for test 2.*

Conclusions
Based on this test it follows that the model shows realistic behaviour and remains stable during simulation. Changing the horizontal positions of the point masses satisfies the constraint. The deviation of the length constraint (2.7a) is very low and corrected in little time. As for model 1, the DLMM shows his

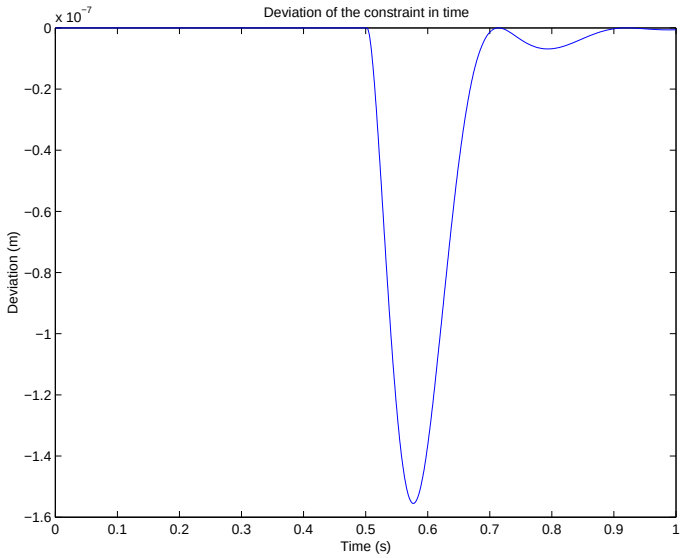

Figure 6.12: *The deviation of the length constraint (2.7a) in time for model 2 in test 2.*

power in satisfying constraints.

6.3.3 Test 3 Accelerating and Decelerating

Test objective
Test the model behaviour of model 2 when applying external horizontal forces, which represent acceleration and deceleration forces.

Description
For $t \in [0, 1]$ an external positive force of $50N$ is applied on point mass 2 and for $t \in [2, 3]$ an external negative force of $50N$ is applied on point mass 2. Note that, according to Section 4.3.4 the external horizontal force is not applied to the wheel contact point, point mass 1, but on the rim represented by point mass 2.

The following initial conditions are taken

i	position	velocities	i	position	velocities
1	(0,0)	(0,0)	4	(1,0)	(0,0)
2	(0,3588)	(0,0)	5	(1,3588)	(0,0)
3	(0,7392)	(0,0)	6	(1,7392)	(0,0)

The parameters are as follows

- Mass of point mass i is $m_i = 1$ kg, for $i = 1, 2, 3, 4$.
- Relaxed spring lengths $z_{rel_{v_i}} = 0.4$ for $i = 1, 2, 3, 4$.

- Spring constant k_{v_i} of spring i is $k_{v_i} = 1000$ for $i = 1, 2, 3, 4$.
- Damping constant d_{v_i} of spring i is $d_{v_i} = 100$ for $i = 1, 2, 3, 4$.
- The elasticity constants for the contact algorithm are both chosen as 0.9.
- Friction constant is chosen as $c_w = 0$.
- The time-interval is $t \in [0, 3]$.
- The step size is chosen as $\Delta t = 0.001$

During acceleration, it is expected that the front of the motorcycle (represented by the point masses $4, 5$ and 6) becomes higher than the backside. During deceleration the front has to become lower than the backside.

Results
The angle of the orientation of the chassis subject to the terrain is given in Figure 6.13. Figure 6.14 shows the vertical positions of the point masses in the model.

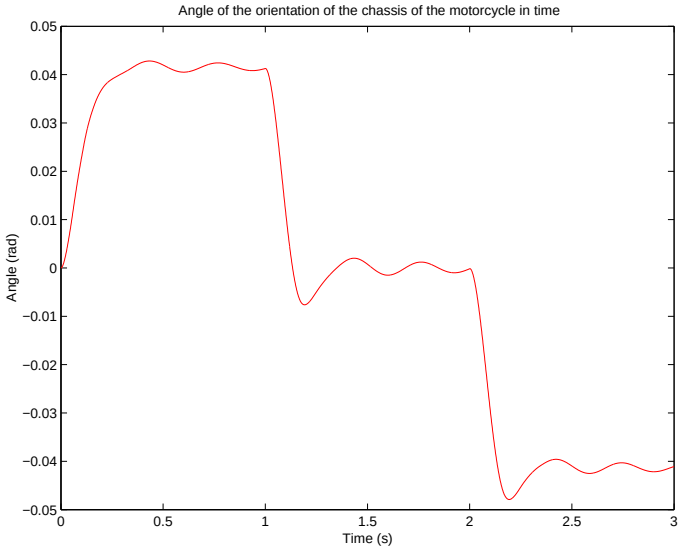

Figure 6.13: *The orientation of the chassis subject to the (flat) terrain for model 2 in test 3.*

In Figure 6.13, it can be seen that the orientation of the chassis depends on the extra horizontal force. When the acceleration force is applied, it is seen that the angle becomes positive, which means that the front side of the motorcycle moves up wards subject to the backside. From Figure 6.14 one can see that both front and backside move in opposite vertical directions. For $t \in [1, 2]$ no external force is applied, and as can be seen, the orientation of the chassis subject to the terrain moves to the equilibrium where the angle is 0. This is also seen in Figure 6.14, from $t = 1$ the vertical positions are going to the equilibrium, which means, that the vertical positions converge to each other. From $t = 2$ it

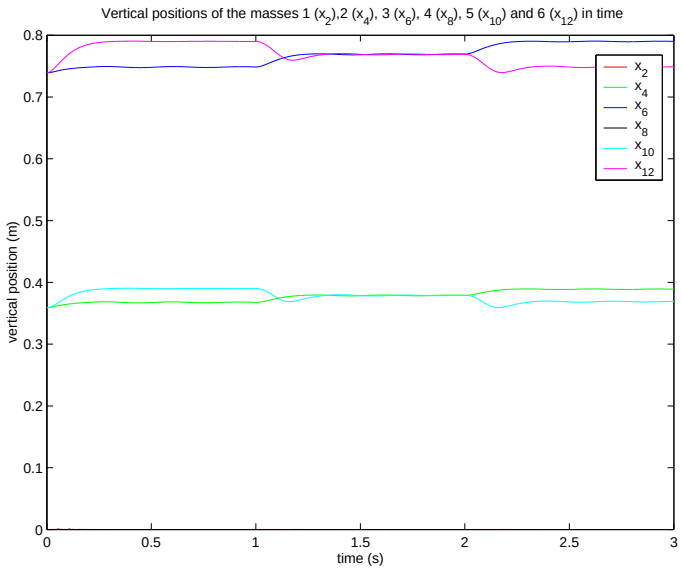

Figure 6.14: *The vertical positions of the point masses in model 2 for test 3.*

is seen that the opposite behaviour occurs, the deceleration force is applied and the angle of the orientation becomes negative. And from Figure 6.14 one can see that the front of the motorcycle moves down wards, while the backside of the motorcycle moves up wards.

Conclusions
From this test it can be concluded that this way of modeling the vehicle leads to a realistic behaviour when applying external horizontal forces representing acceleration and deceleration forces. Acceleration and deceleration cause a rotating chassis in according to reality.

6.4 Comparing Model 1 and Model 2

Both model 1 and model 2 have stable numerical processes. Both models have a good and also realistic behaviour in the case where the model does not start in an equilibrium state on a plane basis and without external horizontal forces.

The difference between the models becomes visible when looking for accelerating and decelerating behaviour as tested in test 3. It is shown that model 1 does not have the wanted rotating behaviour when an external horizontal force is applied. Because this is essential in modeling a motorcycle, model 2 is developed and tested in the remainder of this chapter.

6.5 Further Testing Model 2

In this section model 2 is tested further.

6.5.1 Test 4 Riding over a Hill

Test objective
Test for stability and behaviour properties when the motorcycle rides over a hill.

Description
The hill is defined as a $1 - \cos$ function. The amplitude b of the cos-function is chosen as $b = 0.5$ and the hill is defined as $b - b\cos(0.5(x-5)) = 0.5 - 0.5\cos(0.5(x-5))$, where x is the horizontal position. The following initial conditions are taken.

i	position	velocities	i	position	velocities
1	(0,0)	(5,0)	4	(1,0)	(5,0)
2	(0,3588)	(5,0)	5	(1,3588)	(5,0)
3	(0,7392)	(5,0)	6	(1,7392)	(5,0)

The parameters are as follows.

- Mass of point mass i is $m_i = 1$ kg for $i = 1, 2, 3, 4$.
- Relaxed spring lengths $z_{rel_{v_i}} = 0.4$ for $i = 1, 2, 3, 4$.
- Spring constant k_{v_i} of spring i is $k_{v_i} = 1000$ for $i = 1, 2, 3, 4$.
- Damping constant d_{v_i} of spring i is $d_{v_i} = 50$ for $i = 1, 2, 3, 4$.
- The elasticity constants for the contact algorithm are both chosen as 0.9.
- Friction constant is chosen as $c_w = 0$.
- The time-interval is $t \in [0, 5]$.
- The step size is chosen as $\Delta t = 0.001$.

It is expected that, when riding upward the hill, the velocity of the motorcycle decreases, due to gravity. When the top is passed the velocity will increase. After the hill is passed the vertical positions of the motorcycle are moving to an equilibrium state and the velocity becomes the same as the initial velocity.

Results Figure 6.15 shows the results for the vertical positions of the point masses. Figure 6.16 shows the horizontal velocities of the point masses.

Figure 6.15 shows that the front of the motorcycle moves up wards from $t \approx 0.8$. First the wheel contact point and then, as a result thereof, the rim and the part of the chassis. On $t \approx 1$, the same happens for the backside of the wheel. From Figure 6.15 it can be seen, that from time $t \approx 0.9$, the velocities decreases.

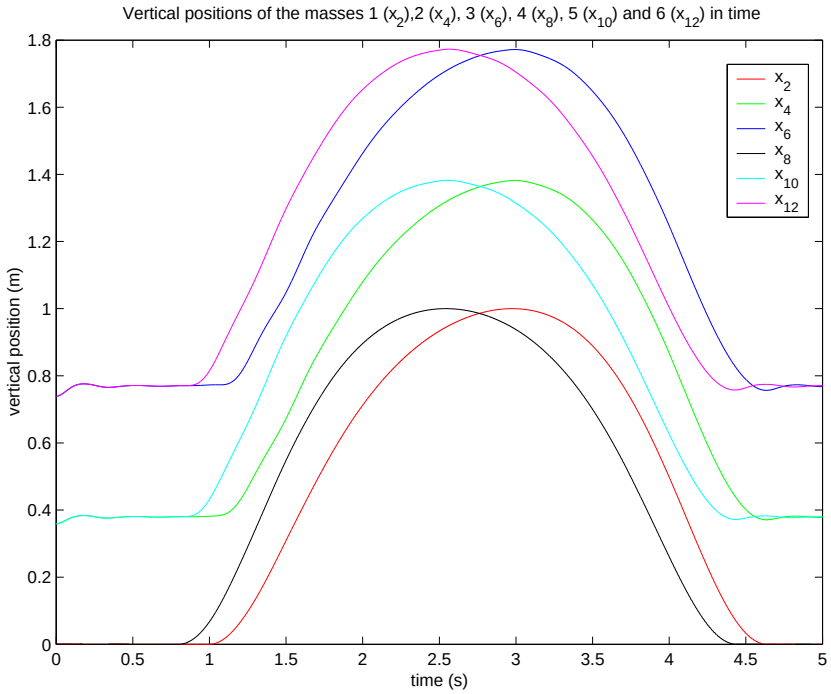

Figure 6.15: *The vertical positions of the point masses in model 2 for test 4.*

This is because the wheel on the front reaches the hill and is going upward the hill. Due to the constraint and the upward motion of the chassis part of the wheel on the front, the horizontal velocities of the rims of both wheels, first shows some increasing. A few times later the back wheel also arrives at the hill and all velocities decrease. When the top of the hill is passed on $t \approx 2.6$ the velocities increase. The minimum velocities of the rims are lower than the minimum velocities of the chassis parts this is caused by the connection of the chassis parts to their rims by springs. The initial velocity of the motorcycle is not reached at the end of the simulation. This is because of the bouncing behaviour of the wheel contact point. Though not always visible, it is still there on some times. Because the horizontal elasticity constant is 0.9 and not 1, some horizontal velocity is lost due to the bouncing behaviour.

Conclusions
From this test it can be concluded that the behaviour of the model when simulating riding over a hill is realistic. A stable numerical solution is obtained.

6.5.2 Test 5 Friction

Test objective
Test the stability and behaviour properties of the model while riding on a flat terrain with friction.

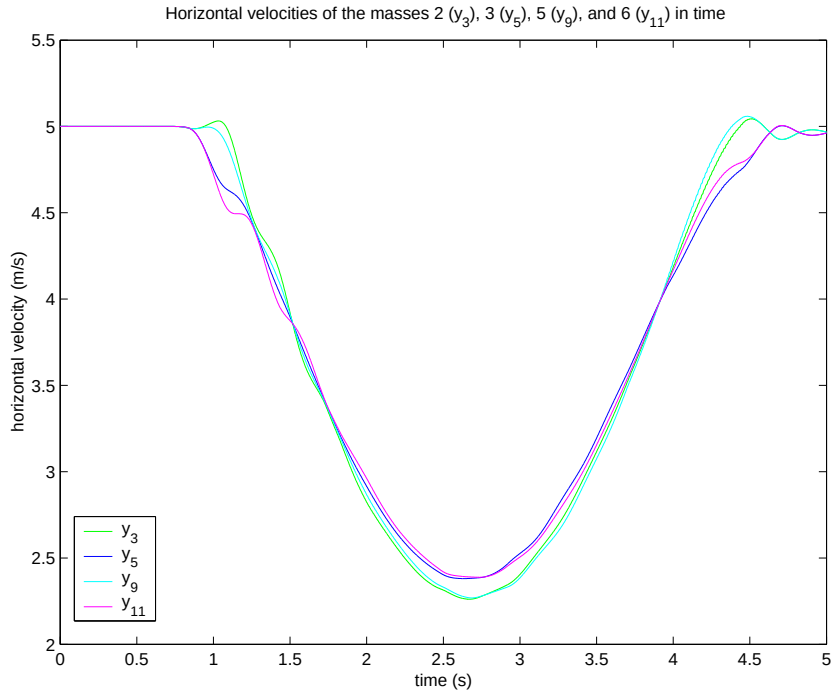

Figure 6.16: *The horizontal positions of the upper four point masses in model 2 for test 4. The values for x_4 and x_{10}, and for x_6 and x_{12} coincide.*

Description
The friction force is implemented as a force opposite to the direction of the traffic. The force F_w is defined as $F_w = c_w \cdot y$ where c_w is a friction constant and y the velocity. The following initial conditions are taken.

i	position	velocities	i	position	velocities
1	(0,0)	(20,0)	4	(1,0)	(20,0)
2	(0,3588)	(20,0)	5	(1,3588)	(20,0)
3	(0,7392)	(20,0)	6	(1,7392)	(20,0)

The parameters are as follows.

- Mass of point mass i is $m_i = 1$ kg for $i = 1, 2, 3, 4$.
- Relaxed spring lengths $z_{rel_{v_i}} = 0.4$ for $i = 1, 2, 3, 4$.
- Spring constant k_{v_i} of spring i is $k_{v_i} = 1000$ for $i = 1, 2, 3, 4$.
- Damping constant d_{v_i} of spring i is $d_{v_i} = 50$ for $i = 1, 2, 3, 4$.
- The elasticity constants for the contact algorithm are both chosen as 0.9.
- Friction constant is chosen as $c_w = 1$.
- The time-interval is $t \in [0, 10]$.

- The step size is chosen as $\Delta t = 0.001$

The behaviour that the horizontal velocities will decrease in the same way as a e^{-x} function decreases is expected.

Results
Figure 6.17 shows the horizontal velocities of the different point masses. Figure 6.18 shows the vertical positions of the different wheel contact points.

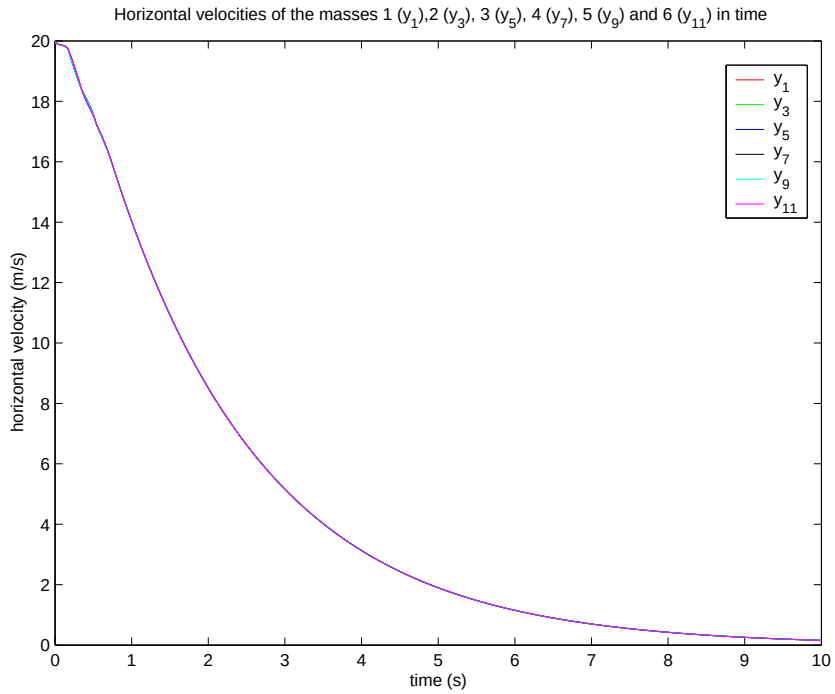

Figure 6.17: *The horizontal velocities of the point masses in model 2 for test 5.*

The general view of the figure shows the expected behaviour. However in the first second, the wanted behaviour is not fully found. To explain this, first is noticed that the friction force is only applied when the wheel rolls. This is the cause of the other behaviour in the first second. It can be seen from Figure 6.18, which represents the vertical positions of the wheel contact points for $t \in [0, 1]$, that in the first second the motorcycle is not rolling but just bouncing. This is because the springs stimulate bouncing of the wheel (on small scale). Only after two bounces have been detected in one time-step (step 7 in the contact algorithm as described in Section 4.3) the vehicle starts rolling.

Conclusions
From this test it can be concluded that the implementation of friction in the model is successfully performed. The behaviour properties are according to the behaviour wanted and expected on base of theoretical and physical knowledge. The bouncing behaviour can be an obstacle.

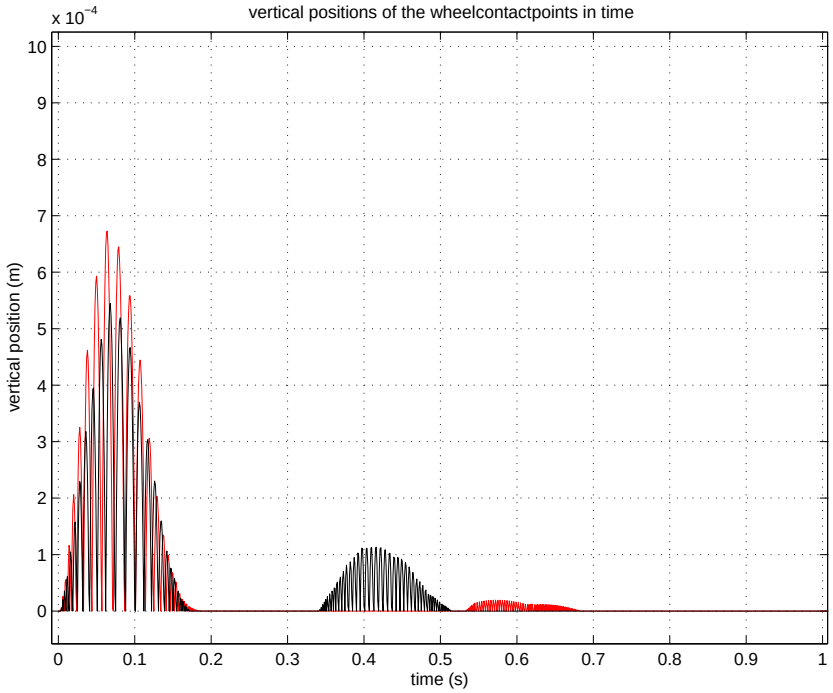

Figure 6.18: *The vertical positions of the wheel contact points in model 2 for test 5.*

6.5.3 Test 6 a Falling Motorcycle on a Flat Terrain

Test objectives
Test the stability and behaviour properties of the model when simulating a falling motorcycle. Testing the contact algorithm on a flat terrain.

Description
In this test the motorcycle will fall from a height of 0.5m above the flat terrain. The terrain height is set as -0.5m. The following initial conditions are taken:

i	position	velocities	i	position	velocities
1	(0,0)	(0,0)	4	(1,0)	(0,0)
2	(0,3588)	(0,0)	5	(1,3588)	(0,0)
3	(0,7392)	(0,0)	6	(1,7392)	(0,0)

The parameters are as follows.

- Mass of point mass i is $m_i = 1$ kg for $i = 1, 2, 3, 4$.
- Relaxed spring lengths $z_{rel_{v_i}} = 0.4$ for $i = 1, 2, 3, 4$.
- Spring constant k_{v_i} of spring i is $k_{v_i} = 1000$ for $i = 1, 2, 3, 4$.
- Damping constant d_{v_i} of spring i is $d_{v_i} = 50$ for $i = 1, 2, 3, 4$.

- The elasticity constants for the contact algorithm are both chosen as 0.9.
- Friction constant is chosen as $c_w = 0$.
- The time-interval is $t \in [0, 1]$.
- The step size is chosen as $\Delta t = 0.001$.

It is expected that the motorcycle begins with a free falling behaviour. When the wheel contact points are on the ground, there will be an impact so that the wheel contact points bounce, while the other point masses are still moving downward. Due to the impact of the wheel contact point, the upper masses will slow down their vertical negative velocity and then, due to the spring forces, again will move upward. However not so high as in the initial state because of the elasticity constant for the vertical direction and the damping forces. Finally the motorcycle will remain on the terrain and move to an equilibrium point.

Results
Fig 6.19 shows the results of this test.

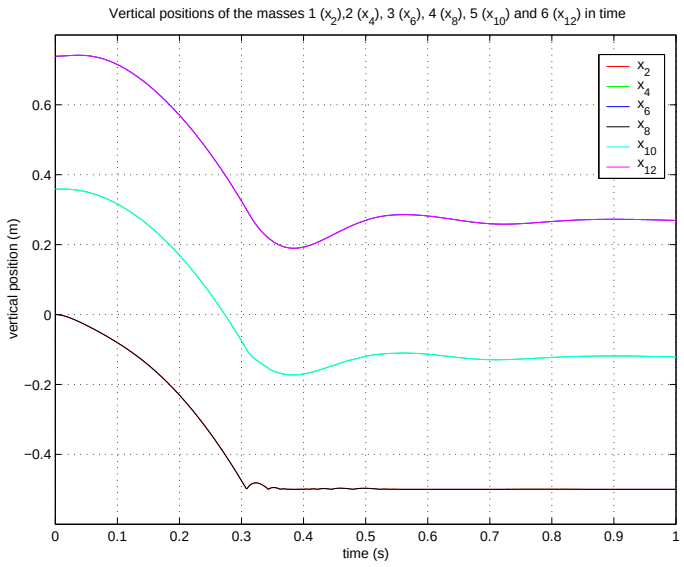

Figure 6.19: *The vertical positions of the point masses in model 2 for test 6.*

The model behaves as was expected.

Conclusions
Based on this test it can be concluded that the modeling and implementation of the contact algorithm is successfully performed. The results are according to the expectations and the numerical process keeps stable.

6.5.4 Test 7 a Falling Motorcycle on a Rising Plane

Test objective
Test the stability and behaviour properties of the model when the motorcycle is falling on a rising plane. Special attention is paid to the contact algorithm.

Description
This test is done to test the contact algorithm for rising planes. The terrain height is given by the following function $h = -1 + 0.2x$, here h is the height dependent of the horizontal location x. The following initial conditions are taken.

i	position	velocities	i	position	velocities
1	(0,0)	(0,0)	4	(1,0)	(0,0)
2	(0,3588)	(0,0)	5	(1,3588)	(0,0)
3	(0,7392)	(0,0)	6	(1,7392)	(0,0)

The parameters are as follows.

- Mass of point mass i is $m_i = 1$ kg for $i = 1, 2, 3, 4$.
- Relaxed spring lengths $z_{rel_{v_i}} = 0.4$ for $i = 1, 2, 3, 4$.
- Spring constant k_{v_i} of spring i is $k_{v_i} = 1000$ for $i = 1, 2, 3, 4$.
- Damping constant d_{v_i} of spring i is $d_{v_i} = 50$ for $i = 1, 2, 3, 4$.
- The elasticity constants for the contact algorithm are both chosen as 0.9.
- Friction constant is chosen as $c_w = 0$.
- The time-interval is $t \in [0, 1]$.
- The step size is chosen as $\Delta t = 0.001$.

It is expected that the wheel on the front of the motorcycle first reaches the terrain. The impact of this wheel contact point will cause a negative horizontal velocity due to the slope of the rising plane. A few moments after the front the backside wheel will reach the terrain and do the same. Some wheel-road contacts will follow and finally the motorcycle will go down backward, rolling on the rising plane.

Results
The following figures are given as output. First the first wheel-road contact of the front of the motorcycle. This is Figure 6.20. Then Figure 6.21, representing the vertical positions of all point masses. Finally the horizontal velocities are represented in Figure 6.22, in this last figure not all velocities are represented for clearance reasons.

In Figure 6.20, it is seen that for $t \approx 0.4$ wheel-road contact occurs. It is seen that the wheel contact point bounces, but very little, from Figure 6.22 it is seen that from this time the horizontal velocity is no longer equal to 0, but gets a negative value. This is the behaviour wanted at first. From Figure 6.21 it is

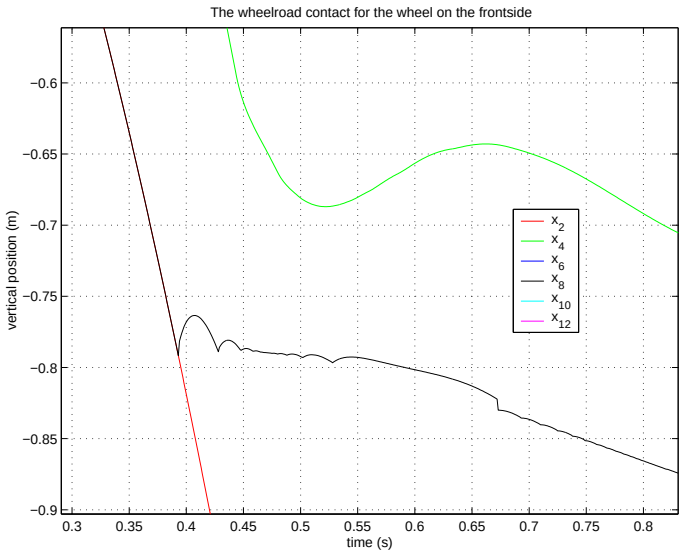

Figure 6.20: *The wheel-road contact of the front of the motorcycle for model 2 in test 7. The black line is the vertical position of the wheel contact point in time. The green line represents the appropriate rim.*

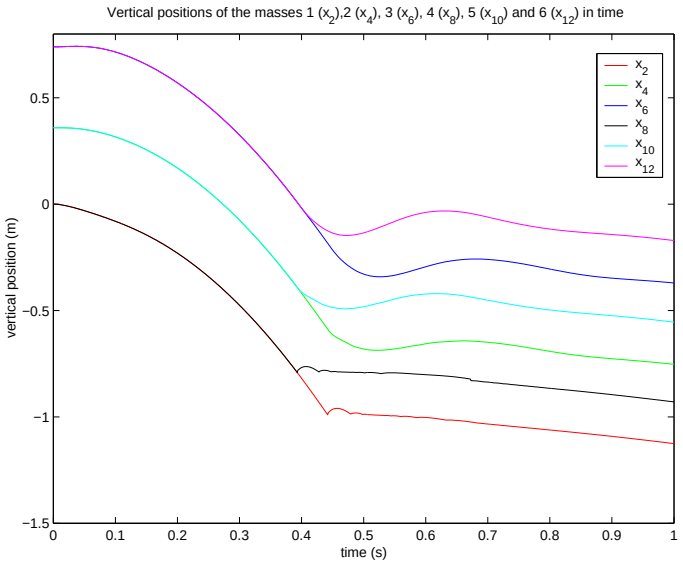

Figure 6.21: *The vertical positions of the point masses for model 2 in test 7.*

seen that for $t \approx 0.44$ the wheel on the backside has wheel-road contact. As seen in Figure 6.22 the back wheel had already a negative horizontal velocity, due to the wheel-road contact of the wheel on the front side. This is caused by the constraints (2.7). After the wheel-road contact of the backside wheel

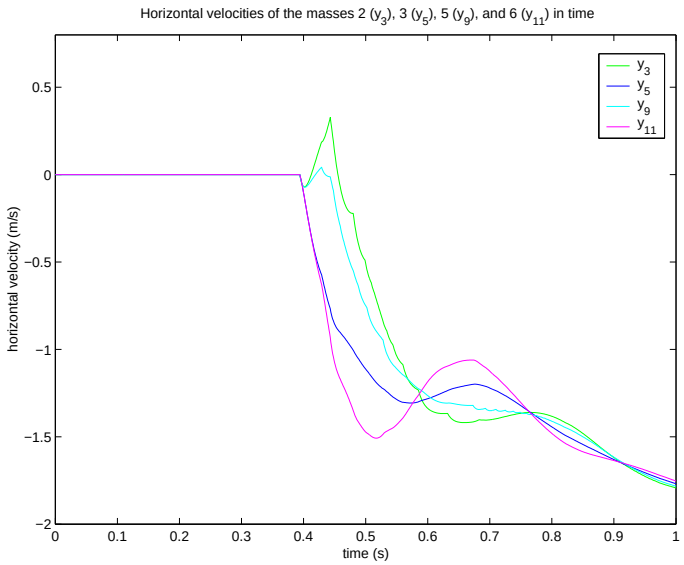

Figure 6.22: *The horizontal velocities of some point masses in model 2 for test 7. y_1 is the horizontal velocity of point mass 1, y_3 of point mass 2, y_5 of point mass 3 and y_7 of point mass 4.*

the swinging behaviour damps out and the point masses become in rest. And then as can be seen in the Figure 6.21 and 6.22 the motorcycle moves down backward, due to the gravitational forces.

Conclusions
After this test the conclusion is that the contact algorithm successfully implemented for a falling two-part vehicle on an rising plane.

6.5.5 Test 8 a Falling Motorcycle on a Sine-Wave Plane

Test objective
Test the stability and behaviour properties of model 2, when the motorcycle is falling on a terrain designed as a sine wave. Special attention is paid to the contact algorithm.

Description
This test is performed to test the contact algorithm for a sine-wave terrain. The terrain height is given by the function $h(x) = -2 + \cos(0.5x)$, here h is the terrain height dependent of the horizontal location x. The following initial conditions are taken.

i	position	velocities	i	position	velocities
1	(0,0)	(0,0)	4	(1,0)	(0,0)
2	(0,3588)	(0,0)	5	(1,3588)	(0,0)
3	(0,7392)	(0,0)	6	(1,7392)	(0,0)

The parameters are as follows.

- Mass of point mass i is $m_i = 1$ kg for $i = 1, 2, 3, 4$.
- Relaxed spring lengths $z_{rel_{v_i}} = 0.4$ for $i = 1, 2, 3, 4$.
- Spring constant k_{v_i} of spring i is $k_{v_i} = 1000$ for $i = 1, 2, 3, 4$.
- Damping constant d_{v_i} of spring i is $d_{v_i} = 50$ for $i = 1, 2, 3, 4$.
- The elasticity constants for the contact algorithm are both chosen as 0.9.
- Friction constant is chosen as $c_w = 0$.
- The time-interval is $t \in [0, 5]$.
- The step size is chosen as $\Delta t = 0.001$

The behaviour, which is expected, is that when one of the wheels reaches the terrain first, then the vertical motion of the wheel contact point will partially change to horizontal motion due to the terrain. A few moments after the first side the other wheel will reach the terrain and do the same. Some wheel-road contacts will follow and finally the motorcycle will turn between two maximums of the terrain and finally come to a stand-still.

Results
Figure 6.23 represents the vertical velocities of all point masses in time. In Figure 6.24 the vertical positions subject to the horizontal positions are given. This figure represents the exact positions of the motorcycle.

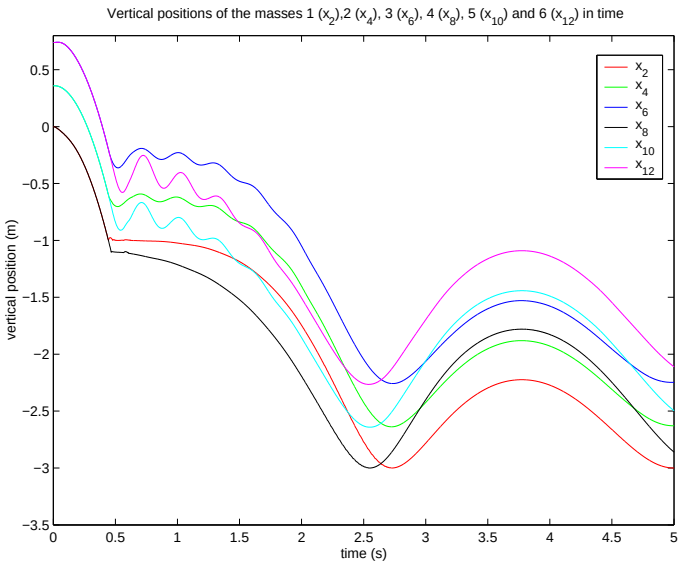

Figure 6.23: *The vertical positions of the point masses in model 2 for test 8.*

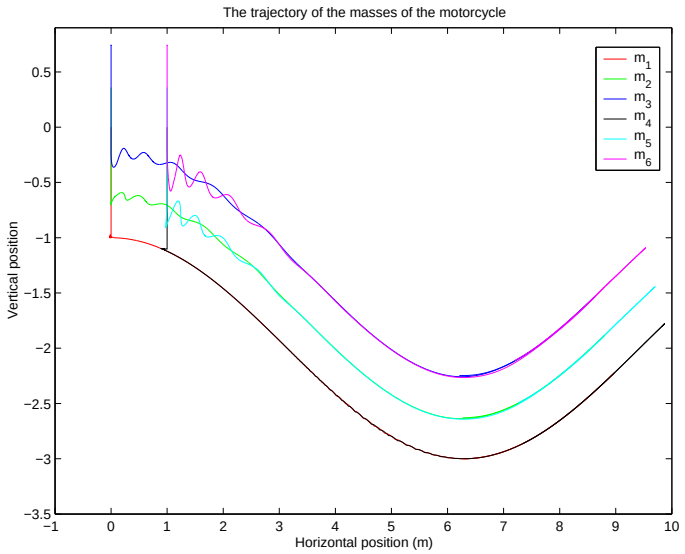

Figure 6.24: *The exact positions of the point masses in the motorcycle model. The vertical positions are set subject to their horizontal positions.*

As can be seen in Figure 6.23 the wheel-road contacts show the same behaviour as in test 7. When the contact is damped out to a rolling behaviour, the upper point masses m_3 and m_6 show just some swinging behaviour, but this also damps out. The vertical position behaves like a sine wave with decreasing amplitude. The motorcycle finally turns between the two maximums, but converges to the minimum. When simulating longer it can be seen that the motorcycle comes to a standstill in the hollow between the two maximums (not represented by a figure).

Conclusions
This test and the previous test for the contact algorithm gives reason to conclude that the contact algorithm successfully is developed and implemented for simulations of a vehicle on a second terrain. The obtained results are realistic.

The last test, that is done, is already described in Section 1.7

6.6 Final Test Conclusion

Based ont the results of the tests performed, it can be stated that the motorcycle model 2 has a realistic behaviour. The contact algorithm is successfully implemented and the results for simulating impacts are realistic and obtained in a stable way. The overall conclusion can be that the model can be used as a tool to do realistic simulations.

Chapter 7

Parameter Studies

7.1 Introduction

In this chapter parameter studies are performed. This means that simulations are done to get a feeling how the behaviour of a motorcycle will change when the different parameters change. The simulations now illustrate how this kind of models can be useful for designers. With the model, they can test the design they made for behaviour properties. The model developed and implemented in this research however is not really according to a real motorcycle, but the technique implemented in the model is the same as for a model of a real motorcycle. The parameter studies performed in this chapter are only done to show how a model as developed in this research can help designers. With some simulations in less time, it is possible to calculate the effects of multiple alterations in a design.

In section 7.2 the design objective is defined and explained. In section 7.3 the spring and damping constants are set such that the shock absorber systems of the motorcycle are as comfortable as possible. All other parameters are fixed. In section 7.4 the dimensions of the shock absorber system are also taken into account.

7.2 Design Objective

In the simulations is made use of a design objective. This is a measure for the comfortableness of the motorcycle. As design objective *sd* is chosen the second derivative of the vertical positions of the upper point masses. These point masses represent the chassis part and the driver takes his place on the chassis parts. The second derivatives of the vertical positions represent the vertical accelerations and are therefore a measure for comfortableness.

The design objective *sd* is defined as the maximum of the absolute values of the second derivatives of the vertical positions of the upper two point masses of the

two partial vehicles. The second derivatives are computed from the generated data by the following formula

$$(y_6(t_{n+1}) - y_6(t_n))/\Delta t.$$

Here $y_6(t_n)$ is the vertical velocity of point mass on $t = t_n$ and Δt is the step size. A similar formula is made for the second derivative of the other upper point mass m_{12}. The lower the *sd* the more comfortable the design.

All simulations are performed with the motorcycle riding over a hill. First 5m flat terrain and then the hill begins and is defined as a $1 - \cos$ function. The amplitude b of the cos-function is chosen as $b = 0.5$ and the hill is defined as $b - b\cos(0.5(x-5)) = 0.5 - 0.5\cos(0.5(x-5))$, where x is the horizontal position. The initial horizontal velocities are all 5m/s.

Each section starts with the fixed parameters and in a table the found values for the second derivatives are given for the different varying parameters. The simulations start with initial conditions such that the vertical positions of all point masses in the model are in equilibrium. To obtain this positions before starting the simulation first another simulation is done on a flat terrain, without horizontal velocity.

7.3 Setting Spring and Damping Constants of the Shock Absorber Systems

This section contains the simulations performed to set the spring and damping constants of the upper springs such that the appropriate *sd* is minimized. The spring and damping constants in the tyre are fixed. It is assumed that the two partial vehicles are identically.

The following parameters are fixed for the simulations in this section.

- Masses: $m_1 = m_4 = 5$kg, $m_2 = m_5 = 20$kg and $m_3 = m_6 = 125$kg;
- relaxed spring lengths $z_{rel_1} = z_{rel_3} = 0.15$m and $z_{rel_2} = z_{rel_4} = 0.50$m;
- friction constant $C_w = 0$;
- length of the motorcycle $L = 1$;
- elasticity constants $e_{vert} = ehor = 0.9$;
- spring constants $k_{v_1} = k_{v_3} = 100000$.
- damping constants $d_{v_1} = d_{v_3} = 5000$

The following table of *sd*s is obtained. dv represents the values for d_{v_2} and d_{v_4}, kv represents the values for k_{v_2} and k_{v_4}.

kv / dv	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
50000	5.8143	5.5109	5.4080	5.6840	5.8449	6.2334	6.3207	6.6422	6.8048	6.9365
10000	6.1580	6.0149	5.9146	5.8579	5.9187	6.3132	6.4180	6.6961	6.8318	6.9449
150000	6.0372	6.1081	6.0864	6.0501	6.0624	6.3688	6.5585	6.7038	6.8519	6.9763
200000	6.2496	6.2574	6.2371	6.2056	6.1852	6.3991	6.5856	6.7437	6.8835	6.9736
250000	7.5477	7.2191	6.8643	6.6523	6.5126	6.4309	6.6169	6.7932	6.8870	6.9923
300000	8.3535	7.9147	7.4610	7.1395	6.9007	6.7291	6.6550	6.7851	6.9132	7.0195
350000	8.4823	7.9094	7.5655	7.2904	7.0685	6.8922	6.7530	6.8220	6.9402	7.0521
400000	8.0738	7.4406	7.1437	7.0513	6.9268	6.8193	6.7259	6.8294	6.9712	7.0558
450000	7.5208	6.8065	6.5378	6.5674	6.5647	6.6361	6.7875	6.8631	6.9736	7.0692
500000	7.0195	6.7929	6.6681	6.6134	6.6517	6.7001	6.8215	6.9161	7.0190	7.0998

One can see that the lowest sd is obtained for $k_{v_2} = k_{v_3} = 50000N/m$ and $k_{v_2} = k_{v_3} = 3000Ns/m$. A general rule for the value of the spring constant seems to be: the lower the constant, the lower the sd. For the damping constants it seems there is not such a rule.

7.4 Setting the Shock Absorber Systems Including Dimensions

This section contains the simulations performed to design the most comfortable shock absorber system for a motorcycle with the given masses. This means that spring and damping constants of the upper springs are varied and that the relaxed spring lengths of the upper springs are varied. Again it is assumed that the two partial vehicles are identical.

From the table of sd-values in the foregoing section one can see that the best values are found in the domain $kv \in [50000, 150000]$ and $dv \in [1000, 5000]$. Therefore this is the domain taken in the simulation in this section for the spring and damping constants. The other parameters varied in this test are z_{rel_2}, z_{rel_4} and L. z_{rel_2} and z_{rel_4} are the same because the two partial vehicles are taken identically and the value is represented by z_{rel}. The domain is $z_{rel} \in [0.2, 0.7]$. The domain for the length of the motorcycle is taken as $L = \in [0.8, 1.6]$.

Five tables of results are represented. First the simulations are performed for $L = 0.8$ and in the appropriate tables the represented sd-values are the minimal sd-values in the whole domain of z_{rel}. The second table contains the same for $L = 1$, and so on. After the sd-values in the tables, the appropriate z_{rel} given.

Table for $L = 0.8$m.

kv / dv	1000	2000	3000	4000	5000
50000	5.7043 (0.2)	5.3888 (0.2)	5.3192 (0.2)	5.4451 (0.4)	5.6717 (0.5)
75000	5.7719 (0.2)	5.7173 (0.2)	5.6263 (0.2)	5.5969 (0.2)	5.7352 (0.4)
100000	6.1756 (0.2)	5.9072 (0.2)	5.8162 (0.2)	5.7700 (0.2)	5.8474 (0.3)
125000	6.2287 (0.2)	6.0500 (0.2)	5.9543 (0.2)	5.9001 (0.2)	5.9482 (0.3)
150000	6.7745 (0.2)	6.2896 (0.2)	6.1010 (0.2)	6.0077 (0.2)	5.9940 (0.2)

Table for $L = 1.0$m.

kv / dv	1000	2000	3000	4000	5000
50000	5.6763 (0.2)	5.3776 (0.2)	5.3038 (0.2)	5.5560 (0.2)	5.7552 (0.2)
75000	5.7273 (0.2)	5.6977 (0.2)	5.6009 (0.2)	5.6318 (0.2)	5.8868 (0.4)
100000	6.3276 (0.2)	5.9040 (0.2)	5.7858 (0.2)	5.7349 (0.2)	5.9287 (0.6)
125000	6.0472 (0.2)	5.9755 (0.2)	5.8979 (0.2)	5.8538 (0.2)	5.9602 (0.6)
150000	6.4222 (0.2)	6.1379 (0.2)	6.0171 (0.2)	5.9517 (0.2)	6.0412 (0.2)

Table for $L = 1.2$m.

kv / dv	1000	2000	3000	4000	5000
50000	5.6300 (0.2)	5.3466 (0.2)	5.2764 (0.2)	5.4519 (0.2)	5.7791 (0.3)
75000	5.6775 (0.2)	5.6626 (0.2)	5.5734 (0.2)	5.5813 (0.3)	5.8988 (0.4)
100000	6.4283 (0.2)	5.8987 (0.2)	5.7631 (0.2)	5.7106 (0.2)	5.9766 (0.5)
125000	5.8510 (0.2)	5.8908 (0.2)	5.8520 (0.2)	5.8188 (0.2)	5.9638 (0.3)
150000	5.9984 (0.2)	5.9573 (0.2)	5.9339 (0.2)	5.9042 (0.2)	6.0610 (0.2)

Table for $L = 1.4$m.

kv / dv	1000	2000	3000	4000	5000
50000	5.5888 (0.2)	5.3303 (0.2)	5.2627 (0.2)	5.5054 (0.2)	5.8273 (0.2)
75000	5.6804 (0.2)	5.6454 (0.2)	5.5574 (0.2)	5.5632 (0.3)	5.7687 (0.2)
100000	6.5235 (0.2)	5.9112 (0.2)	5.7469 (0.2)	5.6936 (0.2)	5.9153 (0.3)
125000	5.6907 (0.2)	5.8288 (0.2)	5.8153 (0.2)	5.7933 (0.2)	5.9647 (0.3)
150000	5.5942 (0.2)	5.8046 (0.2)	5.8642 (0.2)	5.8653 (0.2)	5.9961 (0.3)

Table for $L = 1.6$m.

kv / dv	1000	2000	3000	4000	5000
50000	5.5526 (0.2)	5.3189 (0.2)	5.2530 (0.2)	5.4562 (0.2)	5.7837 (0.2)
75000	5.7040 (0.2)	5.6329 (0.2)	5.5455 (0.2)	5.5458 (0.2)	5.8443 (0.2)
100000	6.5150 (0.2)	5.8973 (0.2)	5.7337 (0.2)	5.6808 (0.2)	5.8963 (0.2)
125000	5.5630 (0.2)	5.7793 (0.2)	5.7879 (0.2)	5.7733 (0.2)	5.9382 (0.2)
150000	5.5194 (0.2)	5.6840 (0.2)	5.8100 (0.2)	5.8361 (0.2)	5.9841 (0.2)

From the data, the following conclusions can be made:

- the springs has to be chosen as short as possible (nearly all found *sd*-values are found for $z_{rel} = 0.2$m, the lower bound of the domain);
- the longer the motorcycle, the more comfortable the motorcycle (the lowest *sd*-value is found for $L = 1.6$m);

- for the spring and damping constant, no clear conclusion can be made. There is a tendency that the lower the spring constant the lower the *sd*-value, because the lowest *sd*-value in all tables is found for $kv = 50000\text{N/m}$. From physical knowledge it is expected that the optimal kv-dv combination is strongly dependent on the terrain.

The lowest *sd*-value is found for $kv = 50000N/m, dv = 3000Ns/m, z_{rel} = 0.2m$ and $L = 1.6m$. There the found *sd*-value is 5.2530. The found trajectories of the point masses in the model for this *sd*-value is represented in Figure 7.1

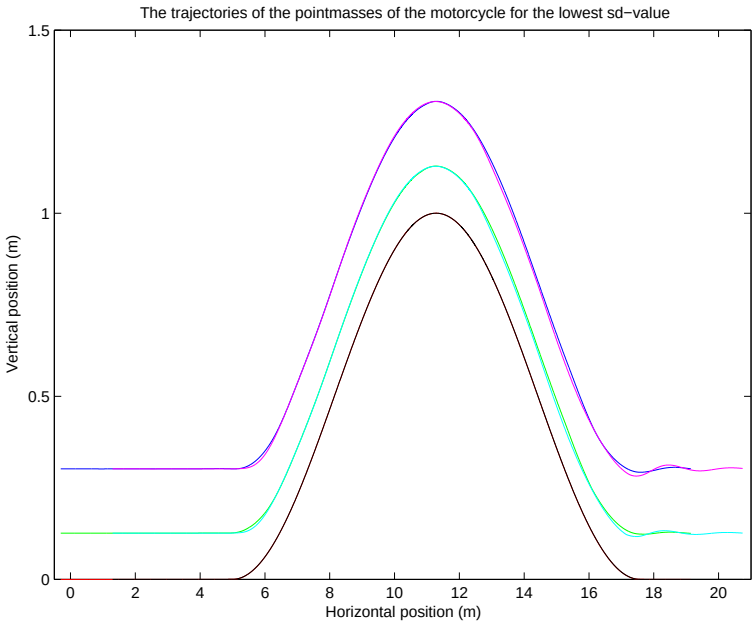

Figure 7.1: *Found trajectories for the point masses of the motorcycle model with most comfortable shock absorber system for riding over the hill as defined in section*

From all tables it is remarkable that for the combination $kv = 50000N/m$ and $dv = 3000Ns/m$ the lowest *sd*-values are obtained.

After these multiple simulations and the resulting *sd*-values thereof one can see, that a behaviour model as developed in this research can be very useful for designers to measure effects of changing design. With this kind of models, it is very easy for designers to calculate the effects of alterations in design, without building the real motorcycle. The model used for the simulations done in this chapter takes a step in the right direction, for more realistic simulating.

Chapter 8

Conclusions and Recommendations

8.1 Conclusions

Modeling a vehicle with use of partial vehicles leads to a behaviour model consisting of a system of equations of motion, equality constraints and inequality constraints. Extending a behaviour model with one or more partial vehicles can easily be done by adding the equations of motion, the equality constraints and the inequality constraints corresponding to the added partial vehicle. The equality constraints define the position of the added partial vehicles subject to the other partial vehicles in the vehicle. The inequality constraints appear from the requirement that the wheel is always above or on the terrain, and that the chassis part is always above the rim and the rim is always above the wheel contact point.

The whole of equations of motion and equality constraints can be solved by the Discrete Lagrange Multiplier Method (DLMM). This numerical method leads to stable numerical processes and handles the equality constraints such that they keep satisfied during the solving process. This method has a fixed step-size and it is an explicit method.

The simulation of wheel-road contact yields satisfying the inequality constraints that set the wheels always above or on the terrain. To simulate the wheel-road contact, a contact algorithm is developed. This contact algorithm simulates the wheel-road contact on an arbitrary terrain in a realistic way.

The behaviour model for a two-part vehicle, such as a motorcycle, in two dimensions is successfully implemented in MATLAB/Simulink. With this Simulink model, it is possible to do multiple simulations. It is easy to change parameters, initial conditions and terrain. Therefore, this kind of models could be very useful to designers to test designs of vehicles for behaviour properties.

The used packages, MATLAB, Simulink and the MATLAB Compiler are useful tools to develop and implement the behaviour model. The MATLAB Compiler makes it easy to create the needed C mex S-functions.

8.2 Recommendations

It is recommended to do further research on modeling vehicles with use of partial vehicles. The following step is to implement the behaviour model of a four-wheeled vehicle in three dimensions.

It is recommended to do further research on solving DAEs arising from mechanical systems.

Simulating the wheel-road contact is simulating a complicated process. It might be useful to model the wheel independent of the chassis and then interrelate the computed values. Then there are more models, which simulates parallel to each other. If this is possible, potential numerical instability can be easily detected and maybe prevented.

The Simulink model in the model made in this research consists mainly of S-functions. The MATLAB Compiler generated all this C mex S-functions from M-files. This tool for making Simulink blocks models is appropriate and it might be in further research on this project.

Appendix A

Using the Developed Model

A.1 List of Files

The following files are available:

- Motor.mdl; this file contains the Simulink model;
- Contact.m; this file contains the MATLAB function which is compiled to the C mex S-function Contact by the MATLAB Compiler, it contains the MATLAB code that perform the contact algorithm;
- Motereq.m; this file contains the MATLAB function which is compiled to the C mex S-function Motoreq by the MATLAB Compiler, it contains the MATLAB code that compute the equations of motion.
- Friction.m; this file contains the MATLAB function which is compiled to the C mex S-function Friction by the MATLAB Compiler, it contains the MATLAB code that determine if friction has to be applied or not.
- Terrain1.m, Terrain2.m and Terrain3.m, This are three different terrain function. All containing MATLAB code. In this files different terrain types are defined. Terrain1.m defines a flat, Terrain2.m a rising plane and Terrain 3 a hill.

In the following sections the different files are discussed.

A.1.1 Motor.mdl

The Simulink model is performed by MATLAB 6.5 Release 13. Using the model with previous versions of MATLAB may give problems. The parameters in the model are represented in the following list:

- the mass vector `mv`, containing the different weights of the point masses in the model;
- the vector `zrel`, containing the relaxed springlengths of the springs in the model;
- the vector `kv`, containing the spring constants of the springs in the model;
- the vector `dv`, containing the damping constants of the damping elements in the schock absorber systems;
- the constant `dt`, which represents the stepsize;
- the constant `cw`, representing the friction constant
- the constant `evert`, representing the vertical elasticity constant;
- the constant `ehor`, representing the horizontal elasticity constant;
- the constant `L`, representing the length of the massless rod, which interconnects the two partial vehicles;
- the vector `xf`, containing possible external forces;
- and the vector `y0`, containing the initial condition of the simulation.

It is most easy to set this parameters globally. The parameters could also be set by defining the different parameters in the masks of the subsystems or the specific Simulink blocks. Before running the model, the M-files has to be compiled by the MATLAB Compiler. How to use this MATLAB Compiler will be described in section A.2.

A.1.2 The M-files Contact.m, Motoreq.m and Friction.m

The M-file Contact.m has an input vector of 65 elements that contains all information needed for the computations in the C mex S-function. The output consists of the corrected state, a vector with 24 elements. The M-file Motoreq.m has an input vector of 60 elements that contains all information needed for the computation of the forces and the computation of the equations of motion. The output vector consist of 24 elements, consisting of the computed uncorrected state. The M-file Friction has an input vector of 36 elements, consisting of the corrected statevector and terrain information needed for the determining if friction has to be applied, or not. The output vector contains two elements.

A.1.3 The Terrain Functions

In subsystem "Terrain information" a S-function block is found. In this block the terrain can be defined by typing the name of the C mex S-function. Available are Terrain1, Terrain2 and Terrain3. These C mex S-functions are obtained by running the MATLAB Compiler on the appropriate M-files.

A.2 How to Use the MATLAB Compiler

Running the Matlab Compiler version to obtain the C mex S-functions form M-files, can be done by prompting the following command in the command line:

```
mcc -S -u <size input vector> -y <size output vector> <name M-file>
```

The flag `-S` defines the kind of S-function wanted, viz. C mex S-functions, the flag `-u` defines the size of the input vector and the flag `-y` sets the size of the ouput vector.

It is remarked that the results of running the MATLAB Compiler on a Windows machine, are not compatible with a Unix/Linux machine. For running the model on an Unix/Linux machine the Compiler has to be runned again on a Unix/Linux machine.

Appendix B

Abbreviations and Symbols

B.1 Abbreviations

In this document the following abbreviations are used

CLMM	Continue Lagrange Multiplier Method
DAE	Differential Algebraic Equation
DLMM	Discrete Lagrange Multiplier Method
IW	The department Mathematical Models and Methods of the NLR
NLR	National Aerospace Laboratory
ODE	Ordinary Differential Equation

B.2 Symbols

The main symbols and their explanations are listed below.

m_i	the mass of point mass i,
x or $\mathbf{x}$	the state vector (the positions of all point masses),
y or $\mathbf{y}$	the velocities of all point masses,
h	the ground height,
$F_c(t)$	the control input force, function of time,
F_g	gravitational force
F_{v_i}	spring force in spring i
F_{d_i}	damping force in spring i
$B(t, \mathbf{x}, \dot{\mathbf{x}})$	the Coriolis, gravitational and centrifugal force/torque vector
L	distance between the two partial vehicles of a two-part vehicle.
k_{v_i}	spring constant of spring i,
d_{v_i}	damping constant of spring i,
$\delta_{min}, \delta_{max}$	resp the minimal and maximal spring length,
$z_{rel_{vi}}$	relaxed spring length of spring i,
g	gravitational constant.
$\alpha, \beta, \gamma, \delta$	used angles, locally defined
L_1, L_2	dimensions of a four-wheeled-vehicle
$d(m_i, m_j)$	distance between point mass m_i and m_j
$P(\mathbf{x}, t)$	a vector-valued function describing the constraints
M	mass matrix of the system
C	the constraint matrix $\frac{d}{d\mathbf{x}} P(\mathbf{x}, t)$
Δt or dt	step size
μ, λ	Lagrange multipliers for the Lagrange multiplier methods
y_{bc}, y_{ac}	resp the velocity just before and just after an impact
t^*	impact time
c_w	friction constant

Bibliography

[1] Matlab product information. URL: http://www.Mathworks.com.

[2] National aerospace laboratory. URL: http://www.NLR.nl.

[3] Bjorck A.˙ *Numerical Methods for Least Sqauares Problems*. SIAM, 1996.

[4] Ralston A.˙ *A First Course in Numerical Analysis*. McGraw-Hill Book Company, 1965.

[5] Ten Dam A.A.˙ Numerical solution of dynamical systems with equality state-space constraints. NLR-TP-89419-U.

[6] Ten Dam A.A.˙ Stable numerical integration of dynamical systems subject to equality state-space constraints. *Journal of Engineering Mathematics*, 26:315–337, 1992.

[7] Petzold L.R. Ascher U.M. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. Society for Indsutrial and Applied Mathematics (SIAM), 1998.

[8] P.W. Bannenberg. Drie dimensionaal gedragsmodel van de contactsimulatie van een voertuig. NLR-Memorandum IW-2002-015.

[9] R. BenYerrou. Multibody systeem simulatie van een voertuig met simpack. NLR-Memorandum IW-2000-18.

[10] Petzold L.R. Brenan K.E., Campbell S.L. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. Elsevier Science Publishing Co., Inc., 1989.

[11] Froberg C.E.˙ *Introduction to Numerical Analysis*. Addison-Wesley Publishing Company, 1973.

[12] Greenspan D.˙ *Discrete Numerical Methods in Physics and Engineering*. Academic Press, Inc., 1974.

[13] Bjorck A. Dahlquist G. *Numerical Methods*. Prentice-Hall, Inc., 1974.

[14] de Kort N.˙ *Klassieke Mechanica*. Teleac, 1989.

[15] C. Doets. Dynamisch gedragsmodel van de contactsimulatie van een voertuig geimplementeerd in matlab/simulink. NLR-Memorandum IW-2001-11.

[16] Stiefel E.L.˙ *An Introduction to Numerical Mathematics*. Academic Press, Inc., 1963.

[17] Roche M. Hairer E., Lubich C. *The Numerical Solution of Differential-Algebraic Systems by Runge-Kutta methods*. Springer-Verlag, 1989.

[18] Wanner G. Hairer E. *Solving Ordinary Differential Equations II*. Springer-Verlag, 1991.

[19] Lieberstein H.M.˙ *A Course in Numerical Analysis*. Harper & Row, 1969.

[20] Hartog J.P.˙ *Mechanics*. Dover publications, inc., 1948.

[21] Ten Dam A.A. Lammen W.F., Nelisse A.H.W. Mosaic: Automated model transfer in simulation development. NLR-TP-2002-628.

[22] Hamming R.W.˙ *Introduction to Applied Numerical Analysis*. McGraw-Hill, Book Company, 1971.

[23] Fhrer C. Soellner E.E. *Numerical Methods in Multibody Dynamics*. B.G. Teubner Stuttgart, 1998.

[24] M.T.S. Tse. Dynamisch gedragsmodel van een voertuig geimplementeerd in matlab/simulink. NLR-Memorandum IW-2000-013.

[25] Arnold V.I.˙ *Mathematical Methods of Classical Mechanics*. Springer-Verlag, 1978.

[26] Lammen W.F.˙ Literatuuroverzicht van multibody-simulatiemethoden. NLR-Memorandum IW-2001-002.

[27] K. Willems. Weg-wiel contact en bewegingssimulatie van voertuigen. NLR-Memorandum IW-2001-027.

[28] Vankan W.J.˙Dynamic model design of wheeled vehicles: a linear multibody model implemented in matlab/simulink. NLR-Memorandum IW-98-19.