Dynamical investigations of the Cooperrider Bogie model

by Ulla Uldahl, s080051

Technical University of Denmark (DTU)
Informatics and Mathematical Modelling (IMM-B.Sc. 2012-34)
Main supervisor: Allan Engsig-Karup
24th February 2012

Preface

This thesis is a requirement for obtaining the Bachelor degree at The Technical University of Denmark (DTU). The work has been carried out at the Department of Informatics and Mathematical Modelling, DTU Informatics. The project started September 2011 and was completed January 2012.

Professor Allan Peter Engsig-Karup and emeritus associate professor Hans True have been supervising the project.

Ulla Uldahl
Kgs. Lyngby, January 2012

Contents

Introduction

This report will focus on the study of the dynamic motions of a railway bogie. To develop a fast and safe train the dynamic behaviours must be considered. In this model the bogie is running with constant velocity along a straight, horizontal and perfect track. When the velocity reaches a certain value, the bogie starts performing oscillations - hunting motions - and it is hard to gain stability again. These hunting motions causes a great wear of the railway vehicle and the tracks, and at the same time it is not comfortable for the passengers. Therefore it is necessary to know at which velocity the hunting motions occurs.

My contributions:

- The model has previously been investigated and implemented in C++ by several other authors and to our knowledge it is the first time, that the model is implemented in MATLAB.
- We have defined all forces as vectors in the same reference coordinate system (the track system). The code is divided into modules, such that e.g. the normal force of each wheel is computed using the same function.
- We have found three critical velocities for this model, which we describe in a bifurcation diagram.

In this model we describe fourteen basic motions, and we use the precomputed parameters from the RSGEO table for realistic wheel/rail contact. The wheels used in this simulation have the S1002 standard profile and the rails have the UIC60 profile. UIC is the worldwide international organisation of the railway sector, and 60 stands for 60 kilogram per meter. The rails are tilted 1/40 inwards the center of the track. For the wheel/rail contact force, the Hertz' and the Shen, Hendrick, and Elkins' theories are used. Wheel lift off is not considered in this report.

The Report

In the first chapter, the bogie model is described and all the physical parameters (constants) of this model are listed. In addition, the derivation of the equations is briefly described. This is followed by chapter 2, describing the

mathematical model based on previous work carried out at IMM-DTU [6]. The three coordinate systems and the transformation matrices between these systems are defined. Creep and normal forces are defined by combining the theories by Hertz' and Shen, Hendrick, and Elkins'. In chapter 3 we verify our MATLAB implementation by analyzing the behaviour of the dynamical variables and the contact forces, and by comparing our results against results obtained in previous work [8]. In chapter 4, different methods are described for finding the various solutions of the bifurcation diagram of this model. Chapter 5 includes the conclusion and a brief discussion about further work. Finally, we have some appendices containing a list of symbols used, the rotation matrices, data from the RSGEO table, and the source code.

Chapter 1

The bogie model

In this work we implement the Cooperrider bogie model in MATLAB. The railway vehicle considered is a four-axle bogie wagon, consisting of a car body which is resting on two bogies. Each bogie consists of three stiff elements: A bogie frame and two wheel axles denoted as the front wheel and the rear wheel. Through the primary suspension, the wheel axles are connected to the bogie frames that in turn is connected to the carbody through the secondary suspension.
The wheel axles are connected to the frame with springs and to ensure good driving properties these are relatively stiff. The frame is connected to the car body with both springs and dampers. In comparision to those of the primary suspension the springs are relativley soft. This is to prevent the vibrations, as a result of the tracks, is being transmitted to the car body. In this model all dampers and springs, longitudinal as torsional, are considered linear and they obey Hooke's law, $F_{spring} = -ky$. The values for the springs and dampers are listed in table 1.2.
The wheels used in this simulation have the S1002 profile and the rails have the UIC60 profile. The rails are tilted 1/40 inwards.

1.1 Views of the model

In figure 1.1 a top view of the cooperriders bogie can be seen and figure 1.2 shows a view of the cooperriders bogie seen from the rear. k refers to the springs and D refers to the dampers, both placed at the same position. The left and the right side of the cooperrider bogie is symmetric.

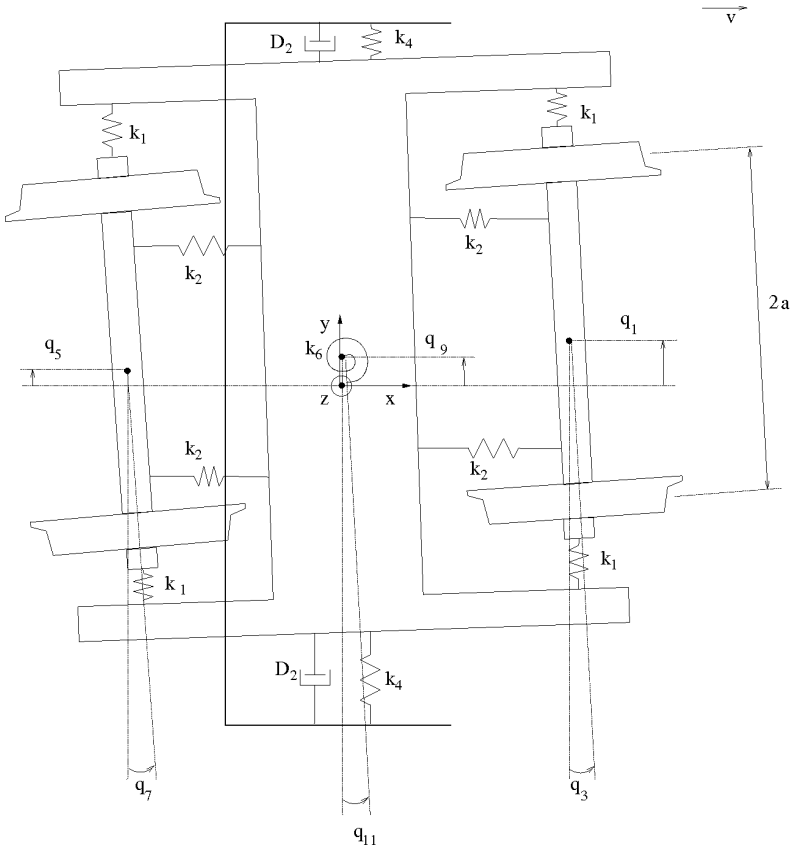

Figure 1.1: Cooperrider bogie seen from the top.

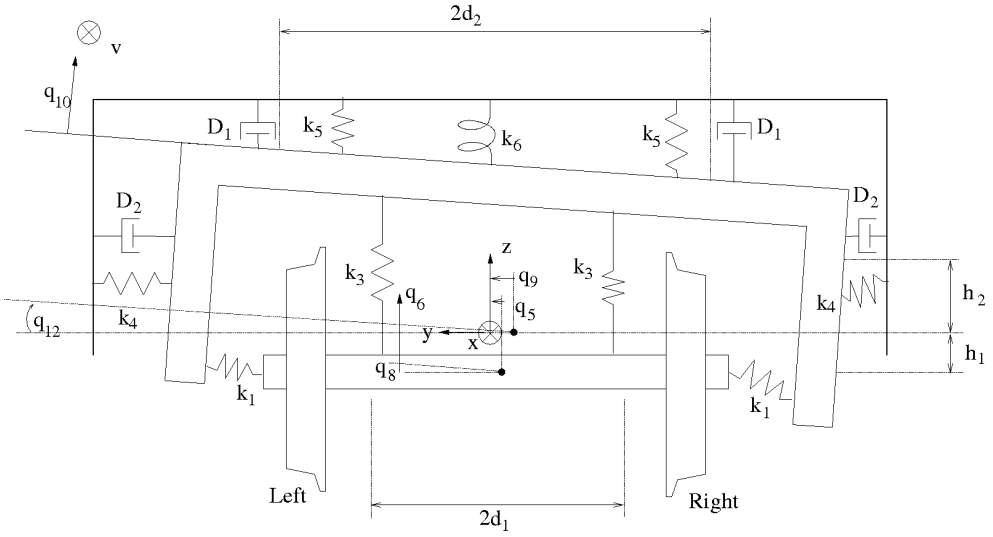

Figure 1.2: Bogie seen from the rear axle.

1.2 The constants of the model

The nominal distance between the two rails, measured $14mm$ under the top of the rails, is $1435mm$. This distance is called the track gauge, see figure 1.3.

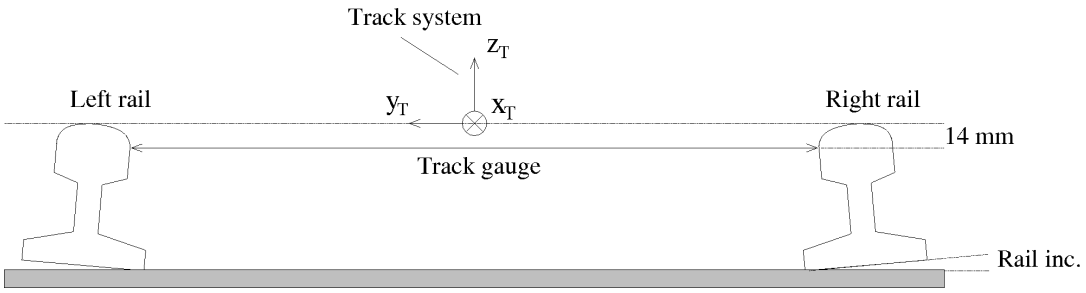

Figure 1.3: Track gauge and rail incliration.

In the following table the constants of the model are listet. The springs are considered linear and they obey Hooke's law. The dampers are also considered linear. M.o.i. is used as shorthand for moments of inertia.

Dimensions	
$a = 0.75$ m	Half the track gauge.
$b = 1.074$ m	Half the distance between the two wheel axles.
$r_0 = 0.425$ m	The nominal rolling radius of the wheel.
$d_1 = 0.620$ m	Horizontal distance from the springs k_2 and k_3 to a center of gravity.
$d_2 = 0.680$ m	Horizontal distance from the springs k_5 and the damper D_1 to a center of gravity.
$h_1 = 0.0762$ m	Vertical distance from the springs k_1 to a center of gravity.
$h_2 = 0.6584$ m	Vertical distance from the springs k_4 and the damper D_2 to a center of gravity.
The masses and moments of inertia (M.o.i.) used in this work	
$m_w = 1022$ kg	Mass of the wheel axle.
$I_{wx} = 678$ kg·m^2	M.o.i. for the roll motions of the wheel around longitudinal axis.
$I_{wy} = 80$ kg·m^2	M.o.i. for the pitch motions of the wheel around lateral axis.
$I_{wz} = 678$ kg·m^2	M.o.i. for the yaw motions of the wheel around vertical axis.
$m_f = 2918.9$ kg	Mass of the frame.
$I_{fx} = 6780$ kg·m^2	M.o.i. for the roll motions of the frame around longitudinal axis.
$I_{fz} = 6780$ kg·m^2	M.o.i. for the yaw motions of the frame around vertical axis.
$m_c = 44388$ kg	Mass of the railcar.
Primary suspensions	
$k_1 = 1823$ kN/m	Lateral horizontal spring, wheel-frame.
$k_2 = 3646$ kN/m	Longitudinal horizontal spring, wheel-frame.
$k_3 = 3646$ kN/m	Vertical spring, wheel-frame.
Secondary suspensions	
$k_4 = 182.3$ kN/m	Lateral horizontal spring, frame-carbody.
$k_5 = 333.3$ kN/m	Vertical spring, frame-carbody.
$k_6 = 2710$	kN/m Torsion spring, frame-carbody.
$D_1 = 20$ kN/m	Vertical damper, frame-carbody.
$D_2 = 29.2$ kN/m	Lateral horizontal damper, frame-carbody.
$D_3 = 500$ kN/m	Lateral horizontal damper, frame-carbody.
$D_m = 150k$ kN·s/m	Material damper, contact area. [6]

Table 1.1: Constants of the model. Unless otherwise mentioned, all paramters are from Lasse Engbo Christensen Master project. [2].

1.3 System of equations

The large weight of the car body and the relative soft secondary suspension will lead to small movements of the carbody. At least in comparision to the bogie frame. Therefore the position of the car body can be assumes fixed and the interaction between the two bogies and the railway vehicle is shut out. The driving properties of the whole railway vehicle is then done by examination one single bogie.
The basic equations are given by Kaas-Petersen [1]. But since the creep force is calculated using Hertz' and Shen, Hendrick, and Elkins theory, there are added 7 more equations to the system. This also gives a more precise and realistic model. These seven extra equations is given by the vertical movement for all three elements and by the roll for the frontwheel and rearwheel and at least the spinperturbation for each wheel axle. Thereby we allow 14 degrees of freedom in this model. Lateral, vertical, yaw and roll motions for the frame and both of the wheelaxles and a spin perturbation only for the wheelaxles.
The fourteen degrees of freedom are:

Front axle:	1.	Lateral - displacement transversal to the track.
	2.	Vertical - displacement perpendicular upward the track.
	3.	Yaw - rotation around a vertical axis.
	4.	Roll - rotation around a horizontal axis.
Rear axle:	5.	Lateral - displacement transversal to the track.
	6.	Vertical - displacement perpendicular upward the track.
	7.	Yaw - rotation around a vertical axis.
	8.	Roll - rotation around a horizontal axis.
Bogie frame:	9.	Lateral - displacement transversal to the track.
	10.	Vertical - displacement perpendicular upward the track.
	11.	Yaw - rotation around a vertical axis.
	12.	Roll - rotation around a horizontal axis.
Front axle:	13.	Angular velocity perturbation.
Rear axle:	14.	Angular velocity perturbation.

By lateral motions is meant displacement in a horizontal plane orthogonal to the tracks direction. By vertical movements is meant displacement perpendicular up relative to the track. By yaw motions is meant rotation in a horisontal plane around a vertical axis. By roll motions is meant rotation in a vertical plane around a horizontal axis parallel to the track.
The 14 state variables $q_1, ..., q_{14}$ are examinated as functions of time t. In order to do this the dynamics equations are formulated using Newton's second law. The forces of the model come from the springs and dampers. The springs are assumed to obey Hooke's low $F_{spring} = -ky$, which is valid for small displacements. For the dampers the linear velocity is used, $F_{damper} = -D\dot{y}$. The springs for the model can be stretched out in three directions. For simplicity

we will only take the contribution from the direction where the springs and dampers are mounted into account. The angles are assumed to be small and thereby an aproximation is done using:

$$\sin(\delta) \approx \delta \quad , \quad \cos(\delta) \approx 1 \tag{1.1}$$

From figure 1.1 and 1.2 we see that the pure lateral displacement for the front wheel is given by:

$$q_1 - q_9 - b\sin(q_{11}) - h_1\sin(q_{12}) \approx q_1 - q_9 - bq_{11} - h_1 q_{12} \tag{1.2}$$

Note the yaw contribution bq_{11}, as the torsion spring is mounted in the center of the bogie frame. The same is valid for the roll $h_1 q_{12}$ where the spring is mounted h_1 above the rotation axis. The pure lateral displacement for the rear wheel is given by:

$$q_5 - q_9 + b\sin(q_{11}) - h_1\sin(q_{12}) \approx q_5 - q_9 + bq_{11} - h_1 q_{12} \tag{1.3}$$

and for the bogie itself:

$$h_2\sin(q_{12}) - q_9 \approx h_2 q_{12} - q_9 \tag{1.4}$$

and the damping of the bogie is given by:

$$2D_2(h_2\dot{q}_{12} - \dot{q}_9) \tag{1.5}$$

where $\dot{q}_9$ and $\dot{q}_{12}$ is the velocity respectively of the lateral motion and the roll motion. By writing up Newton's second law, the ODE for the bogie-frames lateral movement is found:

$$m_f\ddot{q}_9 = 2k_1(q_1 + q_5 - 2q_9 - 2h_1 q_{12}) + 2k_4(h_2 q_{12} - q_9) + 2D_2(h_2\dot{q}_{12} - \dot{q}_9) \tag{1.6}$$

The definition of force moment is $\mathbf{M}_b = \mathbf{r} \times \mathbf{F}$. By use of the rotation matrices $\mathbf{A}_{bT}$ derived in section 2.2 we project the contact force into the body system.

$$\mathbf{M}_b = \mathbf{R}_w \times \mathbf{A}_{bT}(\mathbf{F} + \mathbf{N}) \tag{1.7}$$

$$\begin{aligned}
\mathbf{M}_b &= \begin{pmatrix} 0 \\ R_{wy} \\ R_{wz} \end{pmatrix} \times \begin{bmatrix} 1 & \psi & 0 \\ -\psi & 1 & \phi \\ 0 & -\phi & 1 \end{bmatrix} \left(\begin{pmatrix} F_x \\ F_y \\ F_z \end{pmatrix} + \begin{pmatrix} 0 \\ N_y \\ N_z \end{pmatrix} \right) \\
&= \begin{bmatrix} R_{wy}(-\phi(F_y + N_y) + F_z + N_z) - R_{wz}(-\psi F_x + F_y + N_y + \phi(F_z + N_z)) \\ R_{wz}(F_x + \psi(F_y + N_y)) \\ -R_{wy}(F_x + \psi(F_y + N_y)) \end{bmatrix}
\end{aligned}$$

We do this for the whole system and the governing equations are listed, where $F_{wfc} = (m_w + \frac{1}{2}m_f + \frac{1}{4}m_c)g$ is introduced for the static load for each wheelset.

In the equations N_{ijk} is the normal force and F_{ijk} is the friction force, both given in the track system. The index i defines the wheelset front or rear, the second index j defines the side, left or right and the final index k defines the direction x, y or z. $R_{wy} = a_{ij}$ is the lateral distance to the contact point and $R_{wz} = r_{ij}$ is the actual rolling radius of the wheels, both in reference to the center of mass of the wheelset, see section 2.3 . The latter two equations calculates the difference between the actual angular velocity of the wheelsets and the theoretical value.

$$m_w\ddot{q}_1 = -A_1 + F_{fly} + F_{fry} + N_{fly} + N_{fry} \tag{1.8a}$$

$$m_w\ddot{q}_2 = -A_2 + F_{flz} + F_{frz} + N_{flz} + N_{frz} - F_{wfc} \tag{1.8b}$$

$$\begin{aligned} I_{wz}\ddot{q}_3 = &-A_3 - a_{fr}(F_{frx} + (F_{fry} + N_{fry})q_3) \\ &- a_{fl}(F_{flx} + (F_{fly} + N_{fly})q_3) \end{aligned} \tag{1.8c}$$

$$\begin{aligned} I_{wx}\ddot{q}_4 = &-A_4 + a_{fl}(F_{flz} + N_{flz} - (F_{fly} + N_{fly})q_4) \\ &+ a_{fr}(F_{frz} + N_{frz} - (F_{fry} + N_{fry})q_4) \\ &- r_{fl}(-q_3F_{flx} + F_{fly} + N_{fly} + q_4(F_{flz} + N_{flz})) \\ &- r_{fr}(-q_3F_{frx} + F_{fry} + N_{fry} + q_4(F_{frz} + N_{frz})) \end{aligned} \tag{1.8d}$$

$$m_w\ddot{q}_5 = -A_5 + F_{rly} + F_{rry} + N_{rly} + N_{rry} \tag{1.8e}$$

$$m_w\ddot{q}_6 = -A_6 + F_{rlz} + F_{rrz} + N_{rlz} + N_{rrz} - F_{wfc} \tag{1.8f}$$

$$\begin{aligned} I_{wz}\ddot{q}_7 = &-A_7 - a_{rr}(F_{rrx} + (F_{rry} + N_{rry})q_7) \\ &- a_{rl}(F_{rlx} + (F_{rly} + N_{rly})q_7) \end{aligned} \tag{1.8g}$$

$$\begin{aligned} I_{wx}\ddot{q}_8 = &-A_8 + a_{rl}(F_{rlz} + N_{rlz} - (F_{rly} + N_{rly})q_8) \\ &+ a_{rr}(F_{rrz} + N_{rrz} - (F_{rry} + N_{rry})q_8) \\ &- r_{rl}(-q_7F_{rlx} + F_{rly} + N_{rly} + q_8(F_{rlz} + N_{rlz})) \\ &- r_{rr}(-q_7F_{rrx} + F_{rry} + N_{rry} + q_8(F_{rrz} + N_{rrz})) \end{aligned} \tag{1.8h}$$

$$m_f\ddot{q}_9 = A_1 + A_5 + A_9 + 2D_2(h_2\dot{q}_{12} - \dot{q}_9) \tag{1.8i}$$

$$m_f\ddot{q}_{10} = A_2 + A_6 - A_{10} - 2D_1\dot{q}_{10} \tag{1.8j}$$

$$I_{fz}\ddot{q}_{11} = bA_1 + A_3 - bA_5 + A_7 - A_{11} - D_3\dot{q}_{11} \tag{1.8k}$$

$$\begin{aligned} I_{fx}\ddot{q}_{12} = &\, h_1A_1 + A_4 + h_1A_5 + A_8 - h_2A_9 - A_{12} - 2D_1d_2^2\dot{q}_{12} \\ &- 2h_2D_2(h_2\dot{q}_{12} - \dot{q}_9) \end{aligned} \tag{1.8l}$$

$$\begin{aligned} \dot{\beta}_{fy}\dot{q}_{13} = &-A_{13} + r_{fr}(F_{frx} + (F_{fry} + N_{fry})q_3) \\ &+ r_{fl}(F_{flx} + (F_{fly} + N_{fly})q_3) \end{aligned} \tag{1.8m}$$

$$\begin{aligned} \dot{\beta}_{ry}\dot{q}_{14} = &-A_{14} + r_{rr}(F_{rrx} + (F_{rry} + N_{rry})q_7) \\ &+ r_{rl}(F_{rlx} + (F_{rly} + N_{rly})q_7) \end{aligned} \tag{1.8n}$$

Where all the spring forces are:

$$A_1 = 2k_1(q_1 - q_9 - bq_{11} - h_1q_{12}) \tag{1.9a}$$
$$A_2 = 2k_3(q_2 - q_{10}) \tag{1.9b}$$
$$A_3 = 2k_2d_1^2(q_3 - q_{11}) \tag{1.9c}$$
$$A_4 = 2k_3d_1^2(q_4 - q_{12}) \tag{1.9d}$$
$$A_5 = 2k_1(q_5 - q_9 + bq_{11} - h_1q_{12}) \tag{1.9e}$$
$$A_6 = 2k_3(q_6 - q_{10}) \tag{1.9f}$$
$$A_7 = 2k_2d_1^2(q_7 - q_{11}) \tag{1.9g}$$
$$A_8 = 2k_3d_1^2(q_8 - q_{12}) \tag{1.9h}$$
$$A_9 = 2k_4(h_2q_{12} - q_9) \tag{1.9i}$$
$$A_{10} = 2k_5q_{10} \tag{1.9j}$$
$$A_{11} = k_6q_{11} \tag{1.9k}$$
$$A_{12} = 2k_5d_2^2q_{12} \tag{1.9l}$$
$$A_{13} = 2k_3d_1^2q_3q_{12} \tag{1.9m}$$
$$A_{14} = 2k_3d_1^2q_7q_{12} \tag{1.9n}$$

and products of small quantities have been neglected i.e. small angles see (1.1) and the simplified contributions from the springs.

1.4 Differential equations

The full set of equations are given by twelve second order and two first order differential equations. In order to solve the system, it is therefore necessary to rewrite them into a system of 26 ordinary differential equations of first order (ODEs). This makes it possible to express the system of equations in the general form.
This is done very easily by introducing ODEs for the velocity. The velocity is used as a variable $v = \dot{x}$, and then the first derivative of this is used to express the acceleration.
The substitutions:

$$x_1 = q_1, x_2 = \dot{q}_1, x_3 = q_2, x_4 = \dot{q}_2, ..., x_{25} = q_{13}, x_{26} = \dot{q}_{14}$$

And the rewritten form:

$$\begin{aligned}
\frac{dx_1}{dt} &= \frac{dq_1}{dt} = \dot{q}_1 = x_2 \Rightarrow \ddot{q}_1 = \frac{d}{dt}\frac{dq_1}{dt} = \frac{d\dot{q}_1}{dt} \\
\frac{dx_2}{dt} &= \frac{d\dot{q}_1}{dt} = \ddot{q}_1 = \frac{\sum F_{y_{frontwheel}}}{m_w} \\
&\vdots \\
\frac{dx_{23}}{dt} &= \frac{dq_{12}}{dt} = \dot{q}_{12} = x_{24} \\
\frac{dx_{24}}{dt} &= \frac{d\dot{q}_{12}}{dt} = \ddot{q}_{12}
\end{aligned}$$

Chapter 2

Mathematical model

In this chapter we describe how the wheel/rail interaction is modelled. We introduced three coordinate systems to describe the orientation of the bogie. The derivations are made for the left wheel and all coordinates are right hand systems. It has been shown that the same equations apply to the right wheel, but with opposite sign for the contact angle see appendix B. Furthermore we derive the equations used calculating the creep and the normal forces. Finally we give a short description of the data in the RSGEO data file.

2.1 Coordinate system

The coordinate systems are found appropriated for the track analysis, especially for calculating the creep and normalforces.

	System	Base	Description
R_T :	$\{O_T; x_T, y_T, z_T\}$	$\mathbf{i}_T, \mathbf{j}_T, \mathbf{k}_T$	Track system
R_b :	$\{O_b; x_b, y_b, z_b\}$	$\mathbf{i}_b, \mathbf{j}_b, \mathbf{k}_b$	2 body systems
R_c :	$\{O_c; x_c, y_c, z_c\}$	$\mathbf{i}_c, \mathbf{j}_c, \mathbf{k}_c$	4 wheel-rail contact systems

Table 2.1: Coordinate systems.

The origin of the track system O_T is in the track center line. x_T is a horizontal axis points in the derection of travel. y_T is a horizontal axis pointing towards the left rail w.r.t to the direction we travel. z_T is pointing upwards from the track center, see figure 1.3.
The origin of the body system O_b is located in the center of mass of each wheel axle. This axis is pointing the same way as for the track system. The wheel-rail contact system R_c is an auxiliary coordinate system with an origo in the contact point between the wheel and rail. x_c is a horizontal axis pointing in the direction of travel and the y_c follows the conicity of the wheel. z_c is perpendicular to y_c and pointing upwards. See figure 2.1.

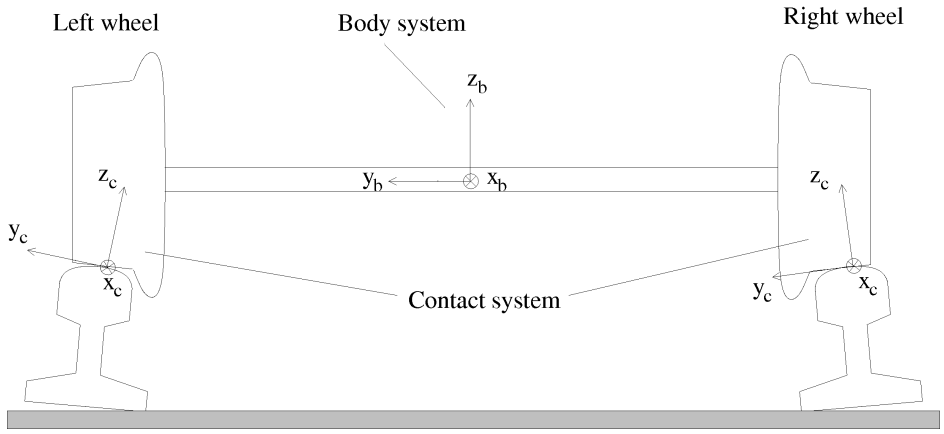

Figure 2.1: Contact system.

2.2 Rotation matrices

By use of rotation matrices the relation between the coordinate system and the orientation of the axes are defined. Then it is easy to pass from one system to another. The body system does not follow the rotation around the spinning axis, which means that there is no pitch and the rotation matrix around the y-axis is therefore unnecessary.
An important property of the rotation matrices is that the determinant is equal to 1. This means that the matrices are orthogonal and thereby their inverse are equal to their transpose, $\mathbf{A}^{-1} = \mathbf{A}^T$. At the same time we see that when $\alpha = 0$ we get the identity matrix for all the rotation matrices.

$$\mathbf{A}_x^{(\alpha)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix} \mathbf{A}_z^{(\alpha)} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Body system to track system - track to body

1. Rotation around z by ψ (yaw)
2. Rotation around x by ϕ (roll)

$$\mathbf{A}_{Tb} = \mathbf{A}_z^{(\psi)} \mathbf{A}_x^{(\phi)} \quad , \quad \mathbf{A}_{bT} = \mathbf{A}_{Tb}^T$$

Wheel/rail system to body system - body to wheel/rail

1. Rotation around x_b by δ - the contact angle is given from RSGEO-table.

$$\mathbf{A}_{bc} = \mathbf{A}_x^{(\delta)} \quad , \quad \mathbf{A}_{cb} = \mathbf{A}_{bc}^T$$

Equations of motion

The velocity $\mathbf{v}$ of the center of mass is expressed with reference to the track base.

$$\mathbf{v} = \begin{bmatrix} V \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} V \\ \dot{y} \\ \dot{z} \end{bmatrix} \tag{2.1}$$

and the angular velocity $\mathbf{\Omega}$ of the wheel-axle is expressed in reference to the body system:

$$\mathbf{\Omega}_b = \begin{bmatrix} 1 & \psi & 0 \\ -\psi & 1 & \phi \\ 0 & -\phi & 1 \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{V}{r_0} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ \frac{V}{r_0} - \dot{\beta} \\ \dot{\psi} \end{bmatrix} \tag{2.2}$$

where $\dot{\beta} = \psi\dot{\phi}$ is a spin perturbation that measures the difference between the actual and the theoretical value of the spin of the wheelset around the y' axis. The nominal spin is $\frac{V}{r_0}$, where r_0 is the nominal rolling radius.

2.3 Wheel/rail interaction

The wheels and rails are in contact and create contact forces. This chapter describes how these forces are determined. In order to solve the normal force contact problem, this contact is considered elastic. Apart from that the wheels and rails are assumed to be rigid. Both wheels and rails are made of steel with following material properties:

Young's modulus	E	$= 2.1 \cdot 10^{11}\mathrm{N/m^2}$
Poissions ratio	ν	$= 0.27$
Shear modulus	G	$= \mathrm{E}/(2(1+\nu)) = 8.2677 \cdot 10^{10}\mathrm{N/m^2}$
Friction coefficient	μ	$= 0.3$

In order to use the theory available, the relative motions between the bodies is here to be found. In order to find the shape of the contact patch the Hertz' theory is used. The elastic deformation is determined by the penetration of the wheels into the rail.

Penetration

The data from RSGEO contains a static penetration and the additional penetration is here to be found. $\mathbf{R}_C$ is a vector defining the position of the center of the mass of the wheelset and $\mathbf{R}_R$ defines the position of the contact point on the rail. Both vectors are with reference to the base of the track system. $\mathbf{R}_w$ is the position of the contact point on the wheel in reference to the base

of the body system. See figure 2.2

$$\mathbf{R}_C = (\bar{y} + y)\mathbf{j}_T + (\bar{z} + z)\mathbf{k}_T$$
$$\mathbf{R}_R = R_{Ry}\mathbf{j}_T + R_{Rz}\mathbf{k}_T$$
$$\mathbf{R}_w = R_{wy}\mathbf{j}_b + R_{wz}\mathbf{k}_b$$

Here the vector $[0, \bar{y}, \bar{z}]^T$ defines the equilibrium position. The vector in the wheel/rail contact system, pointing from the contact point on the wheel to the contact point on the rail, is given by :

$$\mathbf{R}_{pen} = \mathbf{A}_{cb}(\mathbf{A}_{bT}(\mathbf{R}_R - \mathbf{R}_C) - \mathbf{R}_w)$$

where the penetration depth is the z-component of $\mathbf{R}_{pen}$. Note, that this vector is positive.

$$\begin{aligned} q_{pen} \approx \sin(\delta)(-R_{Ry} + R_{Cy} - \phi(R_{Rz} - R_{Cz}) + R_{wy}) \\ + \cos(\delta)(R_{Rz} - R_{Cz} - \phi(R_{Ry} - R_{Cy}) - R_{wz}) \end{aligned} \tag{2.3}$$

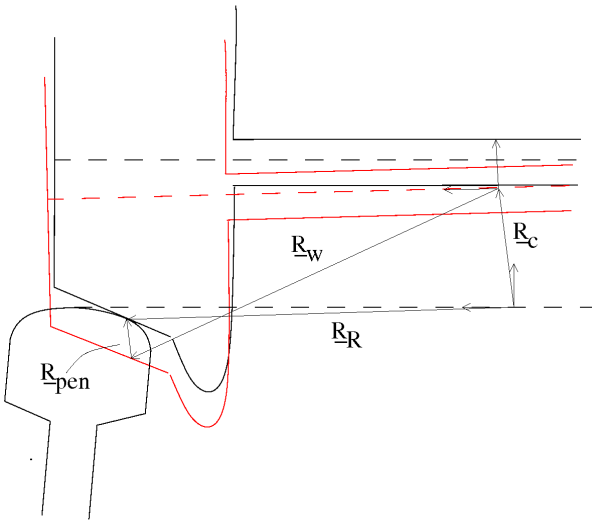

Figure 2.2: Vectors defining the penetration for the left wheel.

Normal force

The normal force is a function of the penetration of the wheel into the rail and is calculated using Hertz' theory. According to Hertz' static theory, the contact region is elliptical, with the major axis a and minor axis b, see figure 2.3. It is seen that the semi axes of the contact point scales with the normal force raised to the power of on third. [5] page 35. From the theory one get

$$a \propto N^{\frac{1}{3}} \quad , \quad b \propto N^{\frac{1}{3}} \tag{2.4}$$

The penetration is required in order to calculate the normal force and the relation is given by:

$$N \propto {q_{pen}}^{\frac{3}{2}} \tag{2.5}$$

The resulting normal force is computed using the actual geometry of the bodies. By pre-calculating the reference value N_0 and q_{pen0} it is possible to compute the normal force during the simulation.

$$N_z = N_{spring} + N_{damper} \tag{2.6}$$

where

$$N_{spring} = N_0 \left(\frac{q_{pen}}{q_{pen0}} \right)^{\frac{3}{2}} \quad , \quad N_{damper} = v_{pen} D_m$$

The linear damper is added for numerical reasons to represent the material damping from the real model. The material damping coefficient is set to $1.5 \cdot 10^5$ Ns/m see [6] page 28, and the velocity of the penetration is calculated by projecting velocity onto the z-direction

$$v_{pen} = (\dot{y} - \dot{\phi}(R_{Rz} - R_{Cz}))\sin(\delta) - (\dot{z} + \dot{\phi}(R_{Ry} - R_{Cy}))\cos(\delta) \tag{2.7}$$

the contact ellipse is also dynamically adjusted using the pre-calculated values of a_0 and b_0.

$$\frac{a}{b} = \frac{a_0}{b_0} \tag{2.8}$$

$$ab = a_0 b_0 \left(\frac{N}{N_0} \right)^{\frac{2}{3}} = a_0 b_0 \left(\frac{N_0 \left(\frac{q_{pen}}{q_{pen0}} \right)^{\frac{3}{2}}}{N_0} \right)^{\frac{2}{3}} = a_0 b_0 \frac{q_{pen}}{q_{pen0}} \tag{2.9}$$

Creep forces

Once the bogie is not in equilibrium position the rolling radius of the right and the left wheel will be different. This will also effect the rotational speed of the two wheels. But the angular velocity of the two wheels have to be the same due to the connection through the axle they are mounted on. Longitudinal and lateral forces will then begin acting on the wheels and these forces are called the creep forces. The creep forces are really important for the dynamic stability. The most recognized theory on the contact region is the Hertz theory stating that the contact region is an ellipse, [7] page 8:15.

In order to use the theory by Kalker, the relative velocity between the wheel and the rail, has to be found. This is given by a contribution from the wheelset

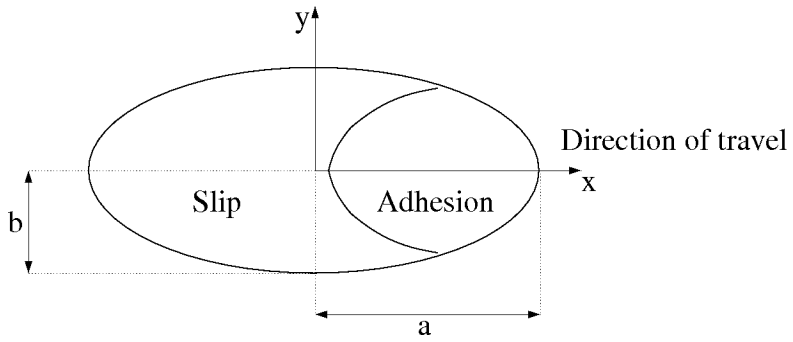

Figure 2.3: Contact patch according to Hertz.

translational motion and a contribution from the wheelset angular velocity. [5] page 151.

$$\begin{aligned}
\mathbf{v}_{\text{con}} &= \mathbf{A}_{cb}(\mathbf{A}_{bT}\mathbf{v} + \boldsymbol{\Omega}_b \times \mathbf{R}_\omega) \\
&\approx \mathbf{A}_{cb}\left(\begin{bmatrix} 1 & \psi & 0 \\ -\psi & 1 & \phi \\ 0 & -\phi & 1 \end{bmatrix}\begin{bmatrix} V \\ \dot{y} \\ \dot{z} \end{bmatrix} + \begin{bmatrix} \left(\frac{V}{r_0} - \dot{\beta}\right) R_{wz} - \dot{\psi} R_{wy} \\ -\dot{\phi} R_{wz} \\ \dot{\phi} R_{wy} \end{bmatrix}\right) \\
&= \mathbf{A}_{cb}\begin{bmatrix} V + \psi\dot{y} + \left(\frac{V}{r_0} - \dot{\beta}\right) R_{wz} - \dot{\psi} R_{wy} \\ \dot{y} - \psi V + \phi\dot{z} - \dot{\phi} R_{wz} \\ -\phi\dot{y} + \dot{z} + \dot{\phi} R_{wy} \end{bmatrix} \\
&= \begin{bmatrix} V + \psi\dot{y} + \left(\frac{V}{r_0} - \dot{\beta}\right) R_{wz} - \dot{\psi} R_{wy} \\ (\dot{y} - \psi V + \phi\dot{z} - \dot{\phi} R_{wz})\cos(\delta) + (\phi\psi V - \phi\dot{y} + \dot{z} + \dot{\phi} R_{wy})\sin(\delta) \\ -\sin(\delta)(-\psi V + \dot{y} + \phi\dot{z} - \dot{\phi} R_{wz}) + \cos(\delta)(\psi\phi V - \phi\dot{y} + \dot{z} + \dot{\phi} R_{wy}) \end{bmatrix}
\end{aligned} \tag{2.10}$$

The equations can be simplified by using the assumption that the wheels will not lift off the rails. It means the contact point projected onto the normal of the contact plane should be zero [5] page 154. And since the bodies in contact are assumed to be rigid, the velocity in the normal direction is zero. ($\mathbf{v}_{\text{con}}|_z = 0$). Then we get the following.

$$\mathbf{v}_{\text{con}} \approx \begin{bmatrix} V + \psi\dot{y} + \left(\frac{V}{r_0} - \dot{\beta}\right) R_{\omega z} - \dot{\psi} R_{wy} \\ (\dot{y} - \psi V + \phi\dot{z} - \dot{\phi} R_{wz})/\cos(\delta) \\ 0 \end{bmatrix}$$

Finally, the spin creepage is defined as the rotation around the normal to the contact plane, where the spin creep is the z component.

$$\mathbf{A}_{cb}\boldsymbol{\Omega}_b = \begin{bmatrix} \dot{\phi} \\ \cos(\delta)\left(\frac{V}{r_0} - \dot{\beta}\right) + \sin(\delta)\dot{\psi} \\ -\sin(\delta)\left(\frac{V}{r_0} - \dot{\beta}\right) + \cos(\delta)\dot{\psi} \end{bmatrix}$$

The creepage is split into three parts - longitudinal, lateral and spin creepage. All creepages are given in the contact plane normalized by the longitudinal velocity.

$$\begin{aligned}\xi_x &= \frac{\mathbf{v}_{\text{con}}|_x}{V} = 1 + \frac{\psi\dot{y} + \left(\frac{V}{r_0} - \dot{\beta}\right)R_{wz} - \dot{\psi}R_{wy}}{V} \\ \xi_y &= \frac{\mathbf{v}_{\text{con}}|_y}{V} = \frac{\dot{y} - \psi V + \phi\dot{z} - \dot{\phi}R_{wz}}{V\cos(\delta)} \\ \xi_{sp} &= \frac{(\mathbf{A}_{cb}\boldsymbol{\Omega}_b)|_z}{V} = \frac{-\sin(\delta)\left(\frac{V}{r_0} - \dot{\beta}\right) + \cos(\delta)\dot{\psi}}{V}\end{aligned}$$

The creep forces have been formulated by Shen, Hedrick and Elkins. The nonlinearity existing between the creep and the creep forces is taken into account in this model. The longitudinal and lateral creep force and a spin creep moment is given by:

$$\mathbf{F} = \begin{Bmatrix} F_x \\ F_y \\ M_z \end{Bmatrix} \tag{2.11}$$

Due to the simulation scenario with straight tracks the spin moment existing around the vertical axis is neglected. Thereby the nonlinear creep force is given by [7] page 8:21.

$$\begin{bmatrix} \tilde{F}_x \\ \tilde{F}_y \end{bmatrix} = -Gab \begin{bmatrix} C_{11} & 0 & 0 \\ 0 & C_{22} & \sqrt{ab}C_{23} \end{bmatrix} \begin{bmatrix} \xi_x \\ \xi_y \\ \xi_{sp} \end{bmatrix} \tag{2.12}$$

where the coefficiens C_{ij} are known as Kalker's creepage coefficients depending on Poisson's ratio and the relation between a/b. For large creepages the vector sum becomes:

$$|\tilde{\mathbf{F}}| = \sqrt{\tilde{F_x}^2 + \tilde{F_y}^2} > \mu N \tag{2.13}$$

Kalker's theory does not take into account that the creep force can not exceed Columb's law, $F = \mu N$, where μ is the friction coefficient. Therefore following relations are made: Let $u = \frac{|\tilde{\mathbf{F}}|}{\mu N}$, then the creep force can approximately be determined with

$$|\mathbf{F}| = \begin{cases} \mu N(u - \frac{1}{3}u^2 + \frac{1}{27}u^3) & \text{if } u \leq 3 \\ \mu N, & \text{if } u > 3 \end{cases} \tag{2.14}$$

Here defining the adjustment factor from Shen, Hedrick and Elkins' model.

$$\epsilon = \frac{|\mathbf{F}|}{|\tilde{\mathbf{F}}|} \tag{2.15}$$

And finaly, the longitudinal creep force F_x and the lateral creep force F_y are given by

$$F_x = \epsilon\tilde{F}_x \quad , \quad F_y = \epsilon\tilde{F}_y \tag{2.16}$$

These forces are given in the contact coordinate system.

2.4 RSGEO - contact table

The RSGEO profile used in this work was provided from Lasse Engbo Christensen [2]. The rail profile is the standard UIC60 profile with an inclination of 1/40 toward the center of the track.
For efficiency reasons it is less time consuming to use the RSGEO table. This contact table contains 3402 pre-calculated points for the lateral displacement between ±17 mm. Any value between two points is given by linear interpolation. The data refer to the left wheels and changing the sign of the lateral position of the wheels we get the corresponding data for the right wheels. The contact table is generated from a static consideration.

Column	MATLAB	Decription
1	`rsg_lat`	Lateral displacement of the center of the mass af the wheelset measured from the center of the track [m]
2	`rsg_n0`	Static normal force in contact coordinate system [N]
3	`rsg_angle`	Angle δ_l between the wheelaxle and the contact plane [rad]
4	`rsg_a0`	Biggest semi-axis of the contact patch (static) [m]
5	`rsg_b0`	Smallest semi-axis of the contact patch (static) [m]
6	`rsg_rwy`	Lateral distance to the contact point on the wheel measured from the center of mass of the wheelset [m]
7	`rsg_rwz`	Actual rolling radius (positive) [m]
8	`rsg_c11`	Kalker's creepage coefficient C_{11} [-]
9	`rsg_c22`	Kalker's creepage coefficient C_{22} [-]
10	`rsg_c23`	Kalker's creepage coefficient C_{23} [-]
11	`rsg_rrz`	Vertical distance of the contact point on the rail measured from the center of the track [m]
12	`rsg_q0`	Static penetration depth [m]
13	`rsg_rry`	Lateral distance to the contact point on the rail measured from the center of the track [m]

Table 2.2: Parameters in the RSGEO table file.

In figure 2.4 the static normalforce is shown. When we for exampel plot the size of the semi axes of the contact ellipse we notice the very fast change in the axes for certain displacements. This shows that the parameters are discontinuous functions of the lateral displacement.

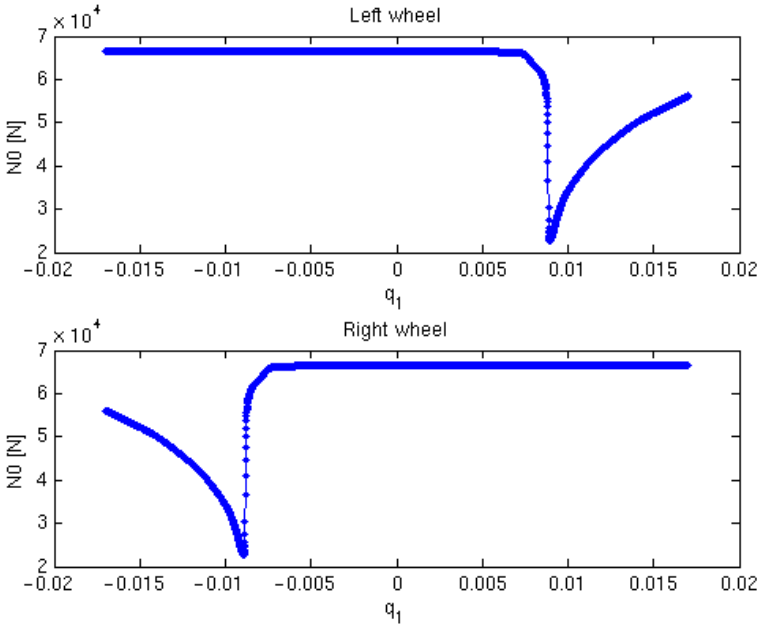

Figure 2.4: The static normal force in the contact coordinate system.

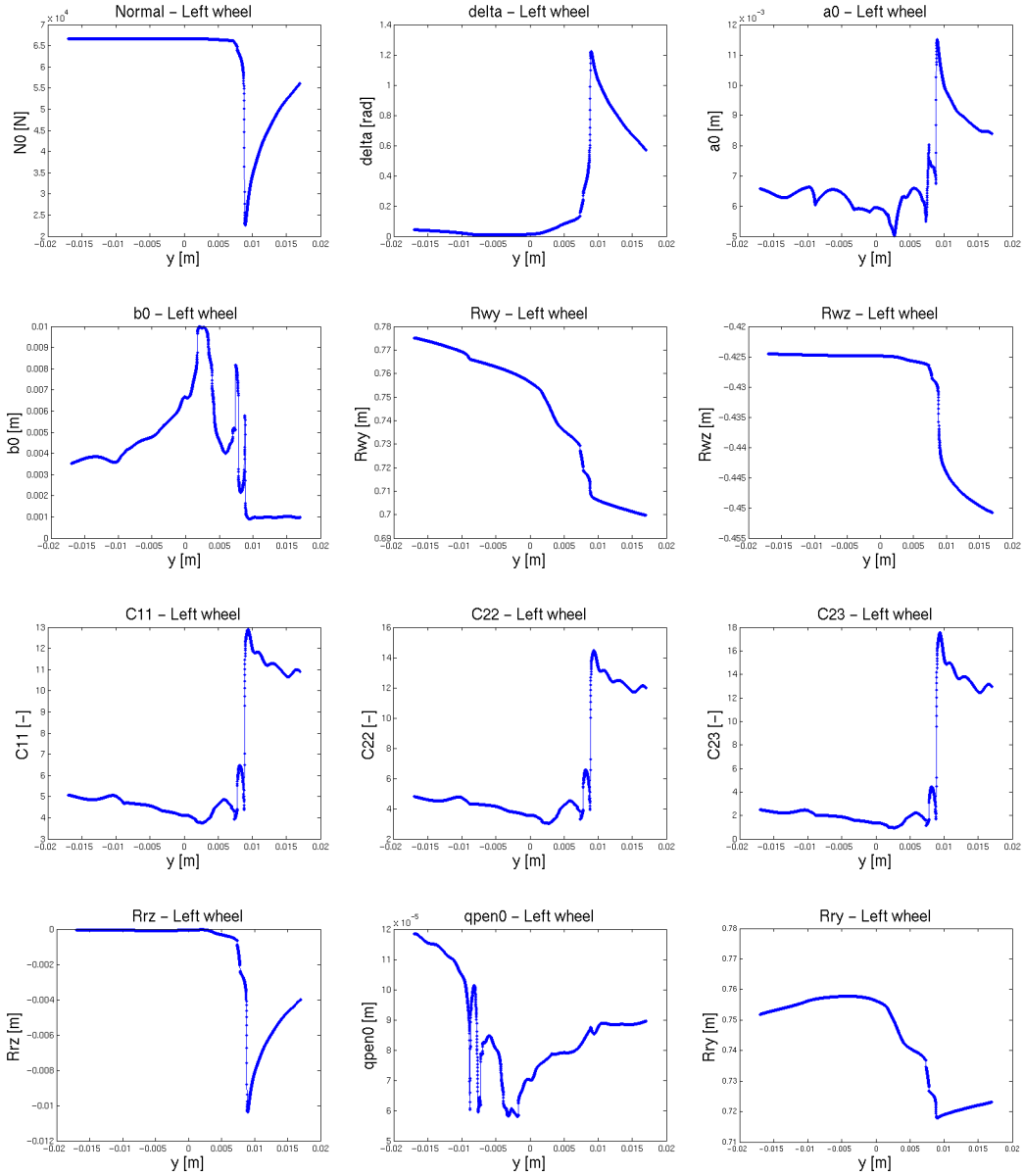

Figure 2.5: RSGEO values for the left wheel as function of the lateral displacement of the wheelset.

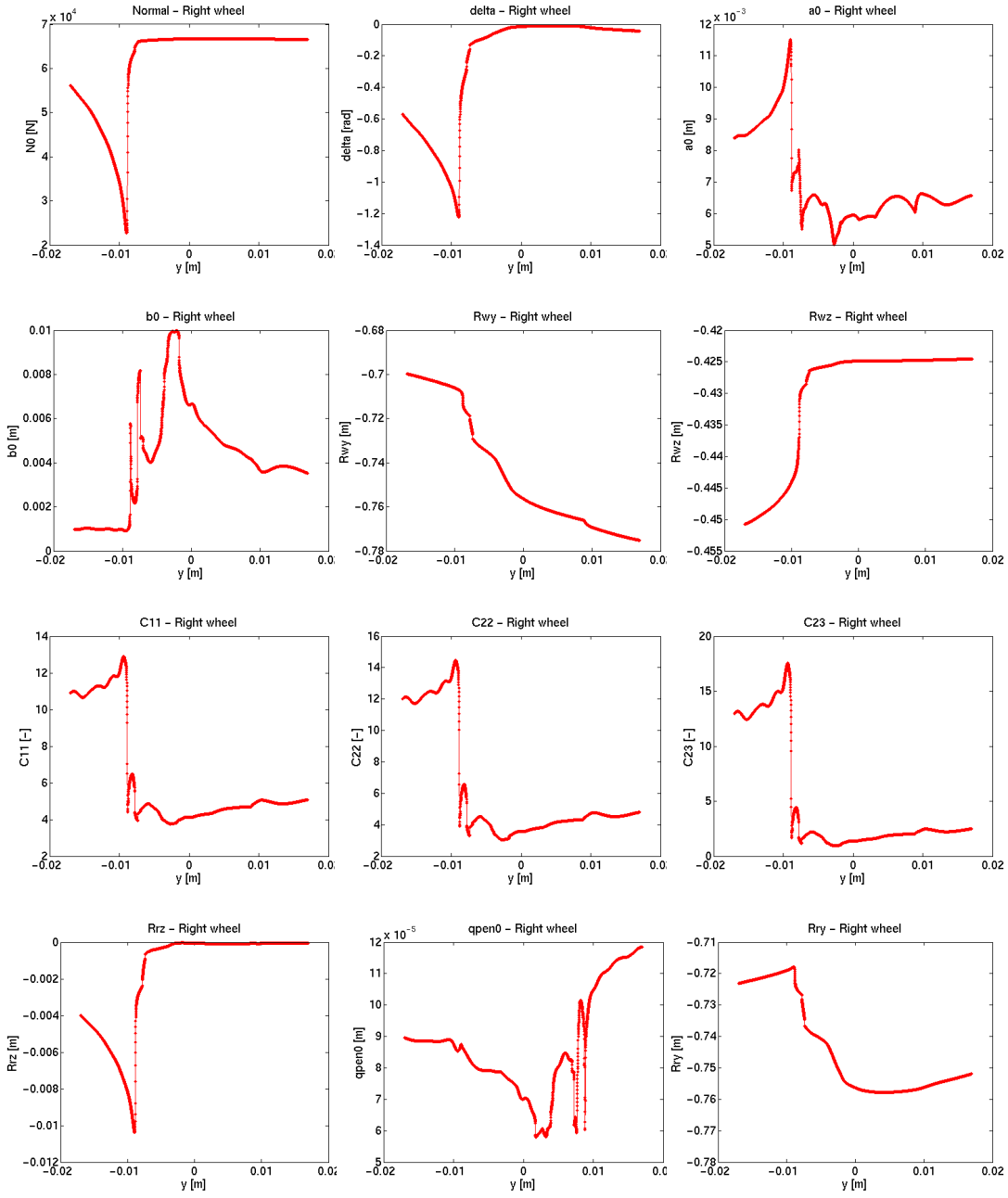

Figure 2.6: RSGEO values for the right wheel as function of the lateral displacement of the wheelset.

Chapter 3

Numerical implementation

In this work we implement the Cooperrider bogie model in MATLAB. The model has previously been investigated and implemented in C++ by several other authors [2, 6, 8]. To our knowledge it is the first time, that the model is implemented in MATLAB. In this chapter we describe the solution of known numerical issues related to the original Cooperrider model, we comment on the verification strategy/process that we have performed during the implementation, we describe how we speed up the computation of the model and our choice of numerical integrator, and we verify our MATLAB implementation. The verification is done by analyzing the behaviour of the dynamical variables and the contact forces, and by comparing our results against the results obtained in [8].

3.1 Known numerical issues of the Cooperrider model

During simulations of the model we have experienced some numerical problems due to the contact forces. In the following we describe these problems and how we solve them in practice .

- **Material damper:** The hard steel to steel contact between wheels and rails exerts large normal forces on the bogie. The normal forces are essentially modelled as spring forces in (2.6). The strong normal forces may cause the wheels to lift from the rails. This problem is solved by adding a linear material damper [6] to the normal forces (2.6).

- **Yaw damper:** A yaw damper is added to the dynamic equation of the bogie frame (1.8k). This is done to stabilize the bogie and to prevent large rotational movements around the z-axis, which may cause the bogie to derail.

3.2 Verification strategy

In the following we list the steps that we have performed in order to verify our MATLAB implementation of the Cooperrider model.

- Simplifying the code, so it runs faster during the verification process - and in general.
- Evaluation of the model by comparing solutions from different numerical time integration methods.
- Testing expected system behavior:
  - Including normal and creep forces.
  - Without normal and creep forces.
- Plotting each direction of the normal and the creep forces.
- Testing the system at different velocities.
- Comparing different models.

A further description of each individual step is given in the sections below.

3.3 Simplifying the code

The data points in the RSGEO table are distributed equidistantly. They are given for the lateral movement of the left wheel for each 10^{-5} m. We compute values between the data points using a linear interpolation. Because of the small distance between each data point (i.e. high density of data points), a linear interpolation will not induce any significant error in the simulation/solution. Therefore, we do not use any high order interpolation method, such as cubic splines. Linear interpolation is the most simple and the computationally fastest method for interpolation. We use our own implementation for linear interpolation, in order to avoid the overhead calculations in the built in Matlab interpolator `interp1`. This further speeds up the computations.
We are careful not to calculate any values more than once, e.g. if $\cos(\delta)$ is needed in different lines of code, then we save the value in a constant variable the first time it is computed.

3.4 Test and evaluation of different numerical time integration methods

In this section we motivate our choice of numerical time integration method. We try different methods of different order and we test how step size and error tolerance influence the convergence of the methods.

The choice of solver

We have considered two different Rung-Kutta methods for numerical integration: explicit Runge-Kutta (ERK) methods and implicit Runge-Kutta (IRK) methods. The stiffness of the system determines whether it is best to use an explicit or an implicit method. Our system of equations is stiff due to the contact forces. This is why, it has been solved using an IRK method in [2, 8, 10]. When solving stiff equation systems, an implicit solver normally uses bigger time steps in comparison to an explicit solver. Such that fewer steps may be needed to compute a solution using an IRK method. However, each implicit step involves the solution of a set of nonlinear equations, and the solution of the nonlinear equation system is computationally time consuming: it is done iteratively by e.g. Newton's method, and each Newton iteration may involve an evaluation of the Jacobian of the model. Although ERK methods may need more time steps in order to perform the integration, then each step is computationally faster than for IRK methods. ERK methods only involves function evaluations, which are not as time consuming as doing both function and Jacobian evaluations, as is done in IRK methods. So, the choice between ERK and IRK methods highly depend on the system we have to solve. Keeping this in mind and based on the work by [8], we have chosen to use an explicit solver.
The first solver we tried was `ode45`, which is a built in solver in MATLAB. It is explicit using the Dormand-Prince scheme of 4th order accuracy. This method works well for most problems, and was therefore applied as our first try for simulating the bogie. For comparison we also tried `ode15s` (built in MATLAB solver), which in general is a good method if the problem is stiff. We did not find any notable difference in the solution, but the CPU time increased when using `ode15s`. Inspired by [2] we also tried ode solvers with lower order than `ode45`: Bogacki-Shampine (2nd order accuracy) and Heun-Euler (1st order accuracy). In order to try the different methods mentioned above, we have used a solver `erk.m` which we have got from [10] personally. Again, we did not find any notable difference in the solution. However, it is worth mentioning that the computation time increases substantially using a solver of low order. Which is because of the smaller step sizes used in low order methods, which results in more function evaluations. We have not included the results obtained by the different methods mentioned above. This is because we have

not found any notable difference in the solutions, and this is also the reason that we decided to use `ode45` for simulating the bogie in Chapter 4.

Convergence test

From [2] we know that the step size and the tolerance affects the performance of the solution. Therefore, we decided to perform a convergence test, where we compare the absolute error (3.1) of the displacement of the rear wheels

$$\text{error} = |q_9^{\text{ref}} - q_9| \tag{3.1}$$

In the test we compute solutions with different tolerances, i.e. the solver uses variable step size. We have done the test with `erk.m` using the Dormand-Prince scheme. The reason for using `erk.m` was because we wanted to be sure to use the infinity norm to measure if a step is accepted or rejected. The reference solution q_9^{ref} in (3.1) is used, because we do not have the exact solution of the system. It is also computed using `erk.m`. The test is performed as follows: we simulate the system in the time span $[0; 2]$ s, the bogie travels at a velocity of $v = 135$ m/s, and we start the simulation with a lateral disturbance of 10^{-3} m on the front wheels. We then compare q_9 with the reference solution q_9^{ref} in 3.1, at $t = 2$ s.
The test is shown in figure 3.1. As expected the error becomes smaller as we decrease the tolerance. We also see that the CPU time increases as we decrease the tolerance. At a tolerance of around 10^{-6} we see that the error is approximately 10^{-10}, and we also see that the CPU time starts to increase significantly. Therefore we use a tolerance of 10^{-6}, both the relative and the absolute, for simulating the bogie in Chapter 4. It is worth mentioning

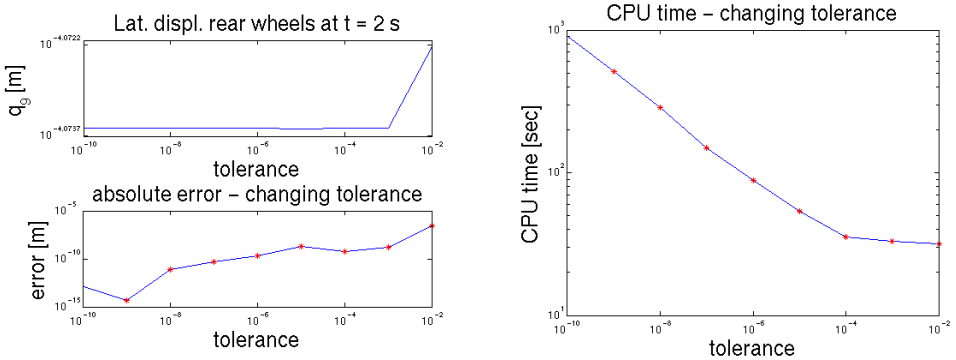

Figure 3.1: Convergence test with different tolerances using `erk.m`.

that Dormand-Prince is a 4th order method, meaning that the accumulated global error per step is off the order $\mathcal{O}(h^4)$. However, this only applies to smooth functions that are differentiable any number of times. Our model is not

differentiable any number of times, because of the discontinuous parameters in RSGEO, see section 2.4.

3.5 Verification of the implementation of the bogie model

The entire system is connected through several springs and dampers. A disturbance in one element will therefore affect the entire model. In the following, we test each body (front wheels, rear wheels and bogie frame) separately. In order to test e.g. the behaviour of the front wheels, we fix the position of the rear wheels and the bogie frame. In this way it is easy to see, if the body to be tested behaves as expected.

Test with normal and creep forces

In the following, we will perform a test to verify if the front wheels are guided back into the center of the track after a disturbance. We let the bogie travel at a velocity of 135 m/s, and we give the wheels a lateral disturbance of 10^{-4} m (figure 3.2), a disturbance of 10^{-4} rad of the yaw angle (figure 3.4) and a disturbance of 10^{-4} rad of the roll angle (figure 3.5). Because the wheels are constrained to be in contact with the rails through the wheel/rail contact point, we do not give any vertical disturbance to the wheels (figure 3.3).

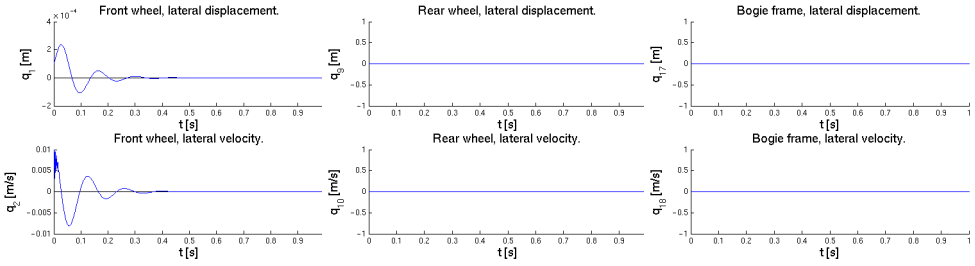

Figure 3.2: Lateral disturbance of the front wheels of 10^{-4} m with fixed rear wheels and bogie frame.

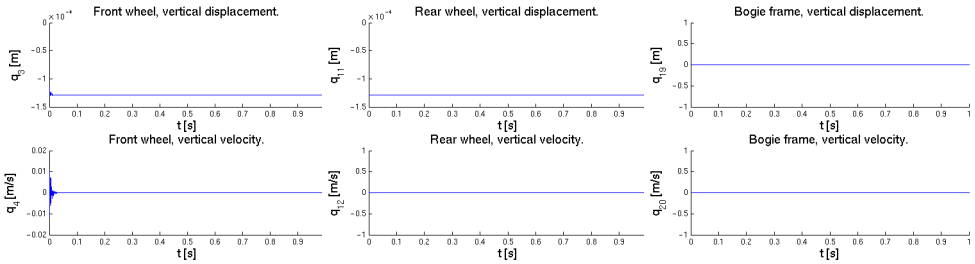

Figure 3.3: No vertical disturbance of the front wheels and fixed rear wheels and bogie frame.

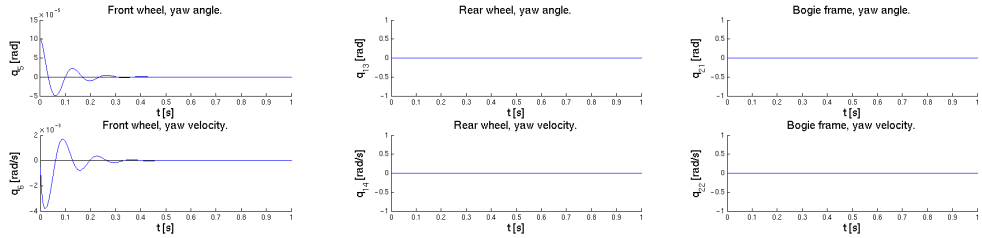

Figure 3.4: Disturbance of the yaw angle of 10^{-4} rad with fixed rear wheels and bogie frame.

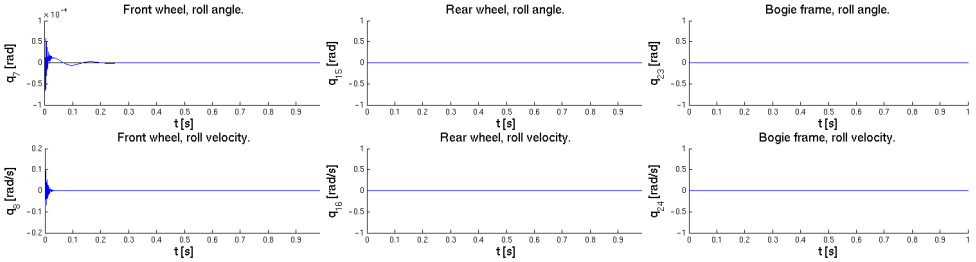

Figure 3.5: Disturbance of the roll angle of 10^{-4} rad with fixed rear wheels and bogie frame.

In the figures 3.2, 3.4 and 3.5 we see that the front wheels are guided to back to center of the track within a short time, and we therefore conclude that the front wheels behaves as expected. However, we see that the front wheels are damped more rapidly than in [8]. This may be because of the fixed car body in our model. The tests for the rear wheels and the bogie frame are performed in the same manner and can be found in Appendix D.

Test without normal and creep forces

In the following, we will perform a test to verify if the front wheels oscillates around the center of the track. To perform this test we neglect the contributions from the normal and the creep forces, otherwise the system will be damped. Once again we let the bogie travel at a velocity of 135 m/s, and we give the wheels a lateral disturbance of 10^{-4} m (figure 3.6), a disturbance of 10^{-4} rad of the yaw angle (figure 3.8) and a disturbance of 10^{-4} rad of the roll angle (figure 3.9). Like in the previous test, we do not give any vertical disturbance to the wheels (figure 3.7).

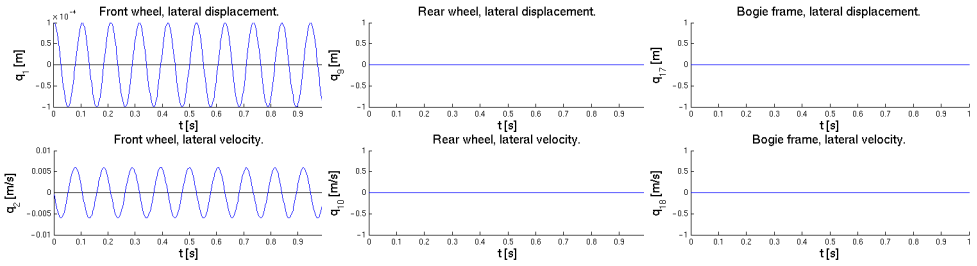

Figure 3.6: Lateral disturbance of the front wheels of 10^{-4} m with fixed rear wheels and bogie frame and without normal and creep forces.

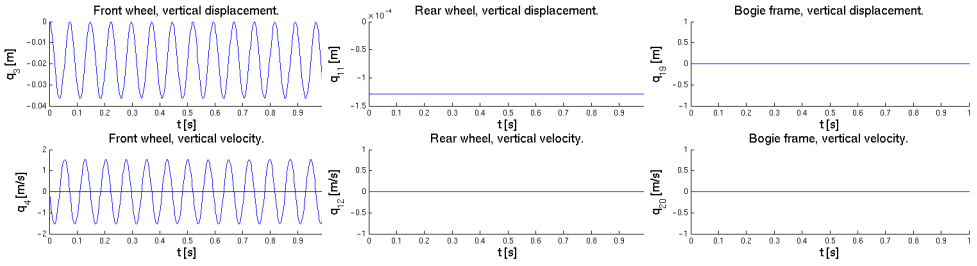

Figure 3.7: No vertical disturbance of the front wheels and with fixed rear wheels and bogie frame and without normal and creep forces.

In the figures 3.6, 3.8 and 3.9 we see that the front wheels oscillates around the center of the track. Due to the missing normal and creep forces, the system has no damping, and as expected, the system does not loose energy and the oscillations continue with constant amplitude - assuming that there is no numerical diffusion in the integration method. The tests for the rear wheels and the bogie frame are performed in the same manner and can be found in Appendix D.

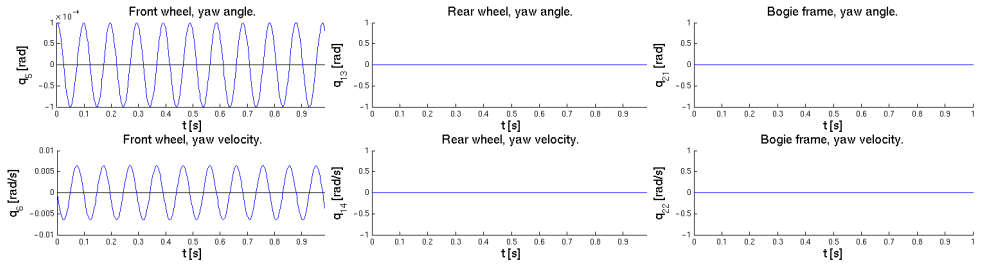

Figure 3.8: Disturbance of the yaw angle of 10^{-4} rad with fixed rear wheels and bogie frame and without normal and creep forces.

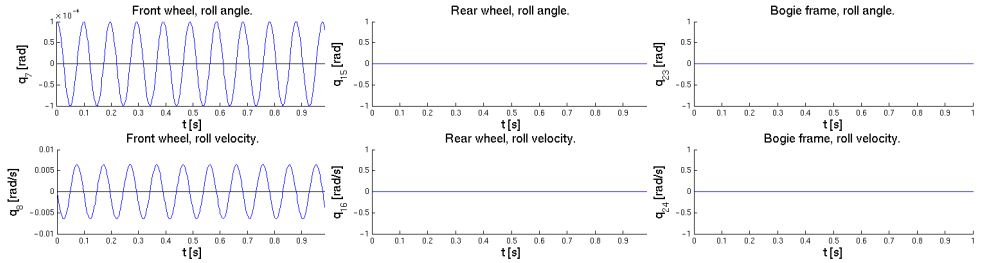

Figure 3.9: Disturbance of the roll angle of 10^{-4} rad with fixed rear wheels and bogie frame and without normal and creep forces.

3.6 Verification of the implementation of the normal and the creep forces

In this section we verify our implementation of the normal and the creep forces. We plot each principal direction (x-, y-, z-direction) of the forces (given in the track system) for the left and the right front wheels, and we compare our results with the results in [8]. We perform two tests: in the first test we let the bogie travel at 40 m/s, in the second test we let the bogie travel at 132 m/s. In each test we give both the front and the rear wheels a disturbance of 10^{-4} rad of the roll angle.

Testing the normal forces

In figure 3.10 the bogie travels at 40 m/s, i.e. below the critical velocity. In figure 3.11 the bogie travels at 132 m/s, i.e. above the critical velocity. In both figures we see that: N_x oscillates around zero for both the left and the right wheel. N_y points towards the center of the track, i.e. it is negative for the left wheel and positive for the right wheel. N_z is positive and points upwards for both the left and the right wheel and corresponds to 1/8 of the total weight of one railway wagon including two bogies. Furthermore, we see

that N_x oscillates in antiphase when comparing the left and the right wheels, and N_y oscillates in phase when comparing the left and the right wheels. The penetration depth *qpen* is positive in the range 10^{-4} m, which is consistent with the RSGEO table values. From these results, and from comparison with the results in [8], we conclude that the normal forces behave like expected.

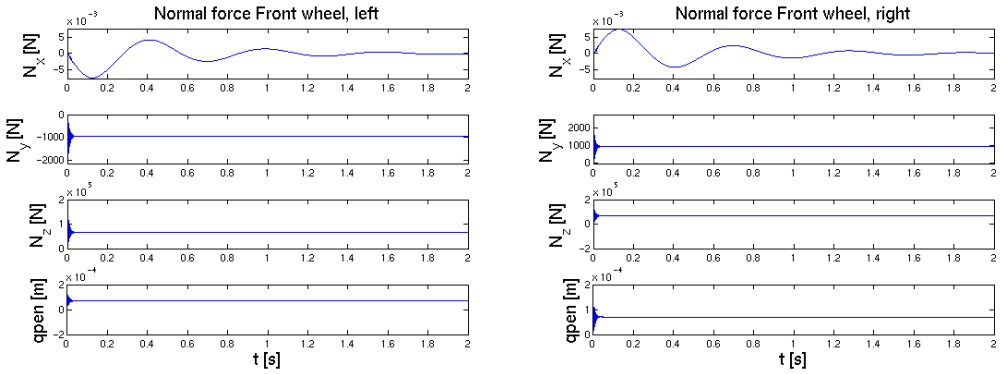

Figure 3.10: Normal forces acting on the left and right front wheels. The bogie travels at 40 m/s, i.e. below the critical velocity.

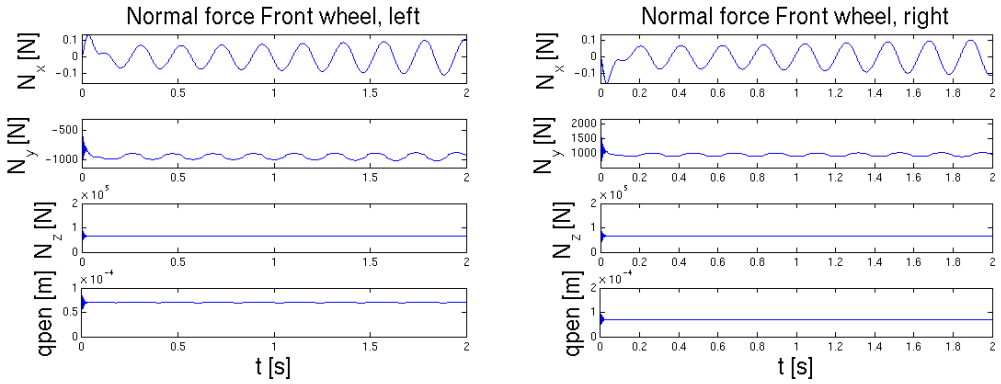

Figure 3.11: Normal forces acting on the left and right front wheels. The bogie travels at 132 m/s, i.e. above the critical velocity.

Testing the creep forces

In figure 3.12 the bogie travels at 40 m/s, i.e. below the critical velocity. In figure 3.13 the bogie travels at 132 m/s, i.e. above the critical velocity. In both figures we see that F_x oscillates around zero for both the left and the right wheel. F_y points away from the center of the track, i.e. it is positive for

the left wheel and negative for the right wheel. F_z is positive and pointing upwards for both the left and the right wheel. Furthermore, we see that F_x and F_z oscillate in antiphase when comparing the left and the right wheels, F_y oscillates in phase when comparing the left and the right wheels. The reason that F_z does not oscillate around zero, is because of the contribution from the spin creep. From these results, and from comparison with the results in [8], we conclude that the creep forces behave like expected.

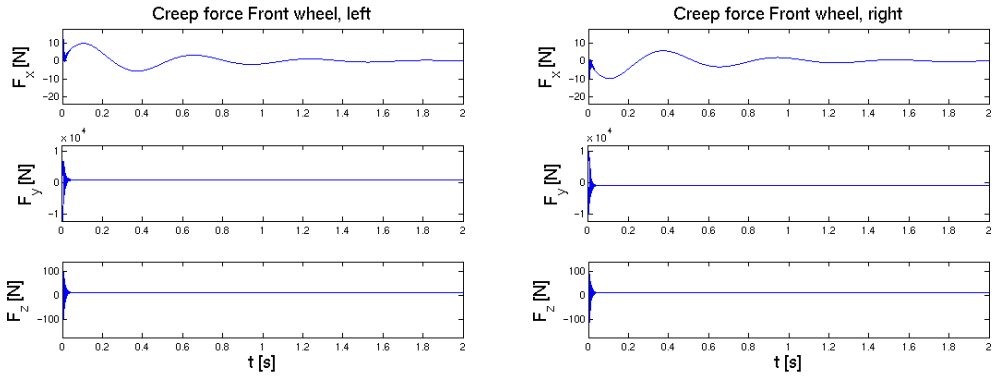

Figure 3.12: Creep forces acting on the left and right front wheels. The bogie travels at 40 m/s, i.e. below the critical velocity.

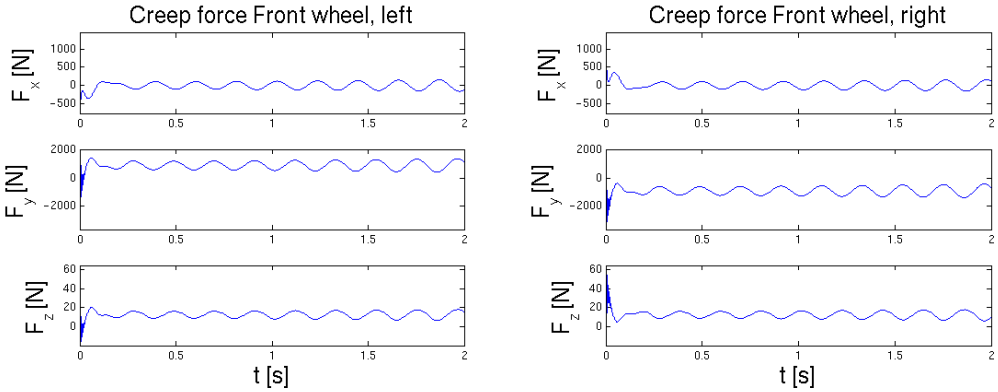

Figure 3.13: Creep forces acting on the left and right front wheels. The bogie travels at 132 m/s, i.e. above the critical velocity.

3.7 Testing the system at different velocities

In this section we test the system at three different velocities. All three tests are started with a lateral disturbance of 10^{-3} m on the front wheels, and we simulate in the time span $[0; 25]$ s.
In figure 3.14 the bogie travels at 128 m/s, and we see that the rear wheels starts performing hunting motions that oscillates around the track center. In figure 3.15 the bogie travels at 132 m/s, and still the rear wheels perform hunting motions that oscillates around the track center. However, at 132 m/s the amplitude is bigger than for 128 m/s. At both situations the bogie travels at velocities above the supercritical hopf bifurcation with an asymptotical stable periodic symmetric solution.

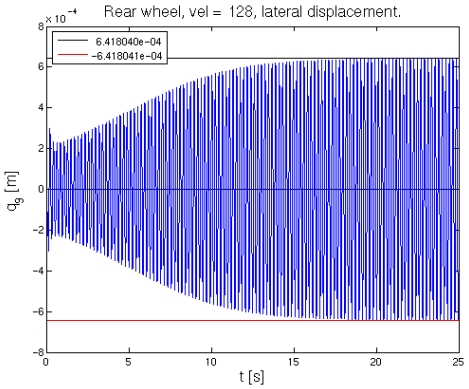

Figure 3.14: Lateral displacement of the rear wheels. The bogie travels at a velocity of 128m/s.

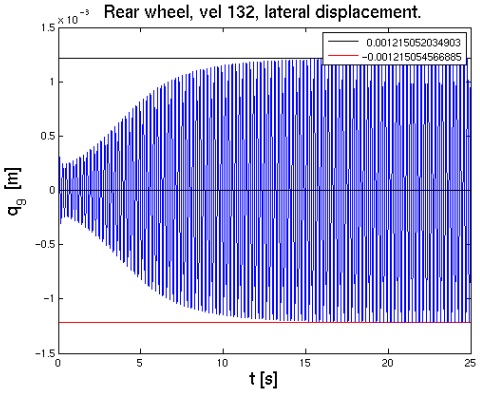

Figure 3.15: Lateral displacement of the rear wheels. The bogie travels at a velocity of 132m/s.

In figure 3.16 and 3.17 the bogie travels at 135 m/s, and first of all we see that the amplitude becomes larger, secondly we see that the hunting motion of the rear wheels does not oscillate around the track center. In addition, we see, that if we change the sign of the disturbance, then the offset will be to the opposite side. In this situation the bogie travels at a velocity above the supercritical symmetry breaking bifurcation with an asymptotical stable periodic asymmetric solution - we comment further on this in Chapter 4

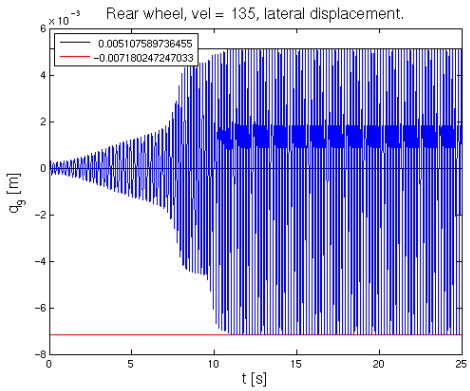

Figure 3.16: Lateral displacement of the rear wheels. The bogie travels at a velocity of 135m/s. The lateral disturbance is 10^{-3} m.

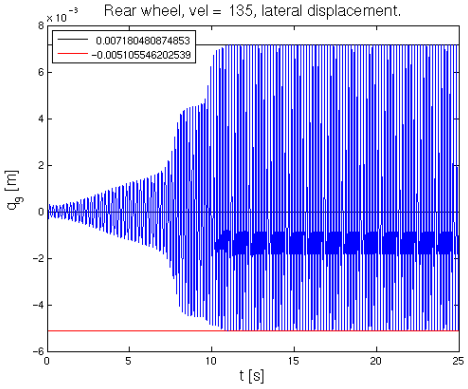

Figure 3.17: Lateral displacement of the rear wheels. The bogie travels at a velocity of 135m/s. The lateral disturbance is -10^{-3} m.

3.8 Comparing two different models

In this section we compare our results from our model with results obtained from a model implemented by Daniele Bigoni. The two models both consist of one bogie attached to a fixed carbody. However, the two model differ from each other with regard to the contact forces, in that our model also includes a material damper in the normal force (2.6).

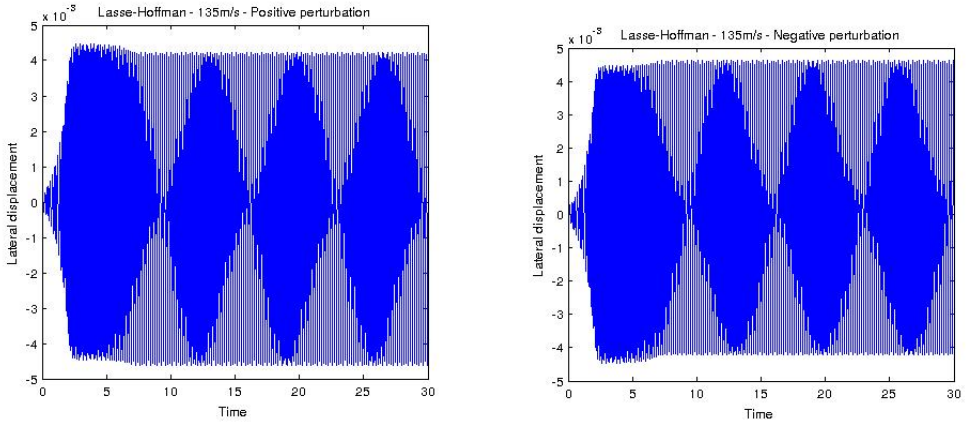

Figure 3.18: Danieles lateral diplacement - velocity $135m/s$.

Figure 3.18 shows the results obtained by Daniele Bigoni with a model as described above, and where the bogie travels at a velocity of 135 m/s. As expected we see that the hunting motion does not oscillate around the track center. Like in our model there is an offset that changes side, as the sign of

the disturbance changes. However, we see a difference in the amplitude in comparison to the behaviour of our model, see figure 3.16 and 3.17.

Chapter 4

Finding the critical velocities

In this chapter we describe the bifurcation diagram for this dynamic system and systematically find the three critical velocities:

- v_C, the critical velocity.
- v_H, the velocity at which the supercritical Hopf bifurcation occurs.
- v_S, the velocity at which the subcritical symmetri breaking bifurcation occurs.

The three velocities are related such that

$$v_C < v_H < v_S \tag{4.1}$$

This is illustrated in the bifurcation diagram in figure 4.1, in which the *green* line represents the asymptotically globally stable stationary solutions, the *red* line represents the asymptotically stable periodic symmetric solutions, and the *blue* line represents the asymptotically stable periodic asymmetric solutions. For the rest of this chapter, when we use the terms: green line, red line or blue line, then we refer to the colored lines in figure 4.1.

4.1 The bifurcation diagram

In this section we will describe what happens with the system if we increase the velocity from below v_C to above v_S, and subsequently decrease the velocity until it is below v_C again. For this we use the illustration in figure 4.1:

- **Increasing the velocity:**
  - **The green line:** Assume we let the bogie travel at a velocity below v_C, i.e. we start the system in a stable stationary solution, and hereafter we slowly increase the velocity. In this case, the system is stable and will not oscillate. If we disturb the system

in this area, it will remain stable and will be guided back into the track center. The system will stay in this condition until we reach v_H.

- **The red line:** The bogie now travels at a velocity above v_H. In this case the system is in a stable periodic symmetric cycle, i.e. it is stable and oscillates around the track center, see figure 3.14 and 3.15. The system will stay in this condition until we reach v_S.
- **The blue line:** The bogie now travels at a velocity above v_S. Also in this case the system is in a stable limit cycle, i.e. it is stable and oscillates. However, it does not oscillate around the track center, see figure 3.16 and 3.17. If we increase the velocity further, the system will stay in this condition and the amplitude will increase.

- **Decreasing the velocity:**
  - **The blue line:** The bogie now travels at a velocity above v_S, and the system oscillates as mentioned above. We then decrease the velocity until we reach v_C. At this point, the system will jump back into a stable stationary solution, i.e. the system will be guided back into the track center and stop to oscillate (the system is back on the green line).

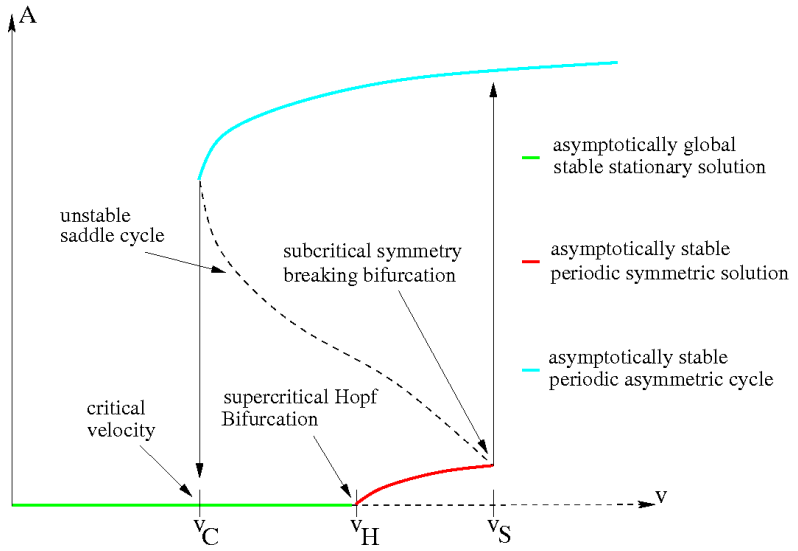

Figure 4.1: The bifurcation diagram with varying velocity.

From the description above, we see that different velocities generates different solutions, which we will list below:

- **For** $0 < v < v_C$**:**
  - 1 asymptotically globally stable stationary solution.
- **For** $v_C < v < v_H$**:**
  - 1 asymptotically stable stationary solution.
  - 2 unstable saddle cycles.
  - 2 asymptotically stable periodic asymmetric cycles.
- **For** $v_H < v < v_S$**:**
  - 1 unstable stationary solution.
  - 1 asymptotically stable periodic symmetric cycle.
  - 2 unstable saddle cycles.
  - 2 asymptotically stable periodic asymmetric cycles.
- **For** $v > v_S$**:**
  - 1 unstable stationary solution.
  - 2 asymptotically stable periodic asymmetric cycles.

4.2 Critical Velocity

In this section we find the critical velocity v_C by a method called ramping, i.e. we want to find the velocity at which the system jumps from an asymptotically stable periodic asymmetric solution (the blue line) to an asymptotically globally stable stationary solution (the green line) in figure 4.1. In this method the velocity of the bogie is decreased until the hunting motions (periodic oscillations) disappear. This is illustrated in figure 4.2, in which we decrease

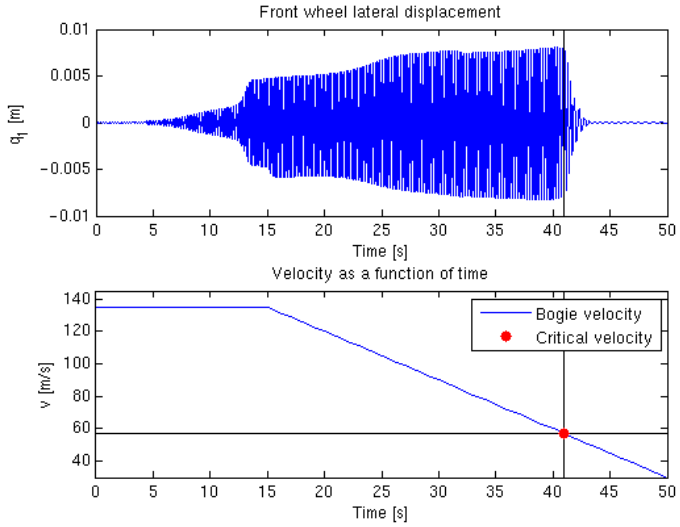

Figure 4.2: Finding the critical velocity by ramping (4.2).

the velocity by 3 m/s per integrated second, i.e. we decelerate the bogie by 3 m/s^2. We let the bogie travel at constant velocity 135 m/s in 15 s before we start to decelerate. In this way, we make sure that the system is in a stable periodic asymmetric cycle before the deceleration starts. The ramping is done as follows

$$v = v_0 + a \max(0, t - t_{ramp}), v_0 = 135 \text{ m/s}, a = -3 \text{ m/s}^2, t_{ramp} = 15 \text{ s} \quad (4.2)$$

In figure 4.2 we see that the oscillations start to decrease at $t = 41$ s, and we use (4.2) to compute the critical velocity to be

$$135 \text{ m/s} - 3 \text{ m/s}^2(41 \text{ s} - 15 \text{ s}) = 57 \text{ m/s} \quad (4.3)$$

The test performed in figure 4.2 only gives a rough approximation of v_C. This is because of the inertia in the system, i.e. it takes a certain amount of time until the hunting motions are completely vanished from the system. Therefore we cannot determine v_C by this test alone. It is simply not possible to see in the figure exactly when the oscillations starts to decrease, and v_C is probably

greater than 57 m/s.
Therefore we do a second ramping test using $a = -0.1$ m/s^2, searching for v_C between 57 m/s and 60 m/s. We use the system state at $t = 40$ s from the first test as initial state in the second test. The ramping in the second test is done as follows

$$v = v_0 + a \max(0, t - t_{ramp}), v_0 = 60 \text{ m/s}, a = -0.1 \text{ m/s}^2, t_{ramp} = 0 \text{ s} \quad (4.4)$$

The results from the second test is shown in figure 4.3 and 4.4. We see

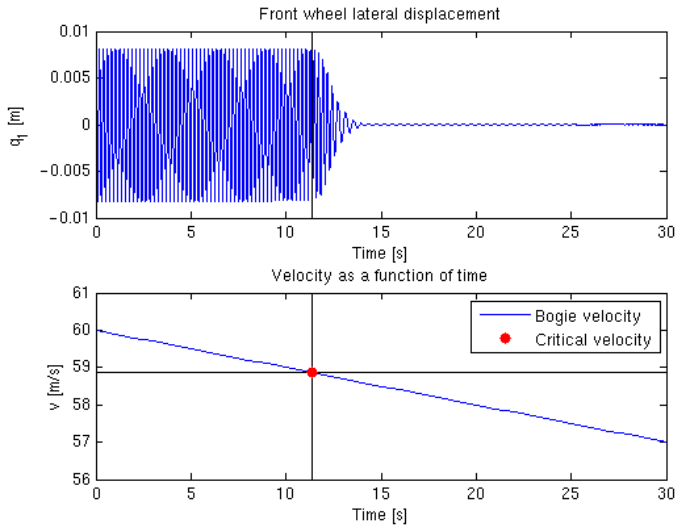

Figure 4.3: Finding the critical velocity by ramping (4.4).

that the hunting motions start to decrease at $t = 11.5$ s, and we use (4.4) to compute the critical velocity to be

$$60 \text{ m/s} - 0.1 \text{ m/s}^2(11.5 \text{ s} - 0 \text{ s}) = 58.85 \text{ m/s} \quad (4.5)$$

For the same reason as mentioned above, the second test will also just give us an approximate value of v_C.
Therefore we perform a third test, such that we can determine v_C more accurately. This is done as follows: we use the same initial values as in the first and the second tests, but now we let the bogie travel at constant velocity. Then we simulate the system until $t = 5$ s and check if the system stays in the stable periodic asymmetric cycle, or if the oscillations collapse. In figure 4.5 we run the bogie at three different velocities: 59.5 m/s, 59.7 m/s and 59.9 m/s. The figure shows that the hunting motions disappear, and that the stationary solution occurs, between 59.5 m/s and 59.7 m/s. Therefore, we conclude that $59.5 < v_C < 59.7$ m/s.

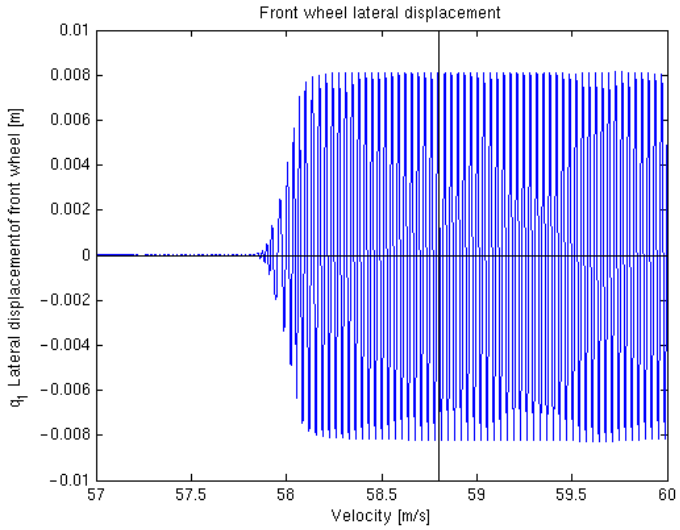

Figure 4.4: Finding the critical velocity by ramping (4.4).

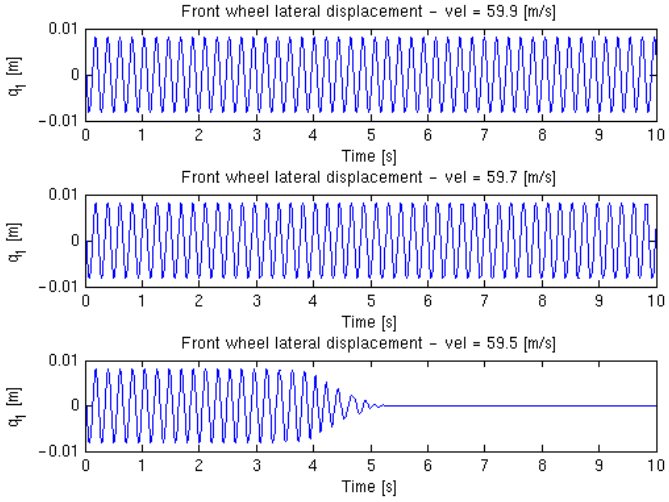

Figure 4.5: Lateral displacement at the velocities 59.5 m/s, 59.7 m/s and 59.9 m/s.

4.3 Supercritical Hopf bifurcation

In this section we find the supercritical Hopf bifurcation v_H, i.e. we want to find the velocity at which the system may shift between asymptotically globally stable stationary solutions (the green line) and asymptotically stable

periodic symmetric solutions (the red line) in figure 4.1. One way of finding the supercritical Hopf bifurcation is to look at the Jacobian matrix of the model. If the real part of the eigenvalues are in the left half-plane, i.e. Re $\lambda < 0$, then the system is in a stable stationary solution (the green line). If the real part of one of the complex conjugated pairs of the eigenvalues are in the right half-plane, i.e. Re $\lambda > 0$, then the system is in an asymptotically stable periodic symmetric solution (the red line). So, assuming the system is in a solution on the green line, and we increase the velocity. Then a change of the eigenvalues, as described above, indicates when the system shifts into a solution on the red line, and we have found v_H.
However, we have not implemented the Jacobian, because we use and explicit method for numerical integration. Instead we use bisection to find the supercritical Hopf bifurcation. From section 4.2 we know that $v_C = 59$ m/s, and from figure 3.14 we know that the system is in an asymptotically stable periodic symmetric solution (the red line) at a velocity of 128 m/s. Using this information, (4.1) tells us that $59 < v_H < 128$ m/s, which we may use to initialize the bisection. The criteria for choosing a new end-point in the bisection, is to check whether the amplitude of the hunting motions is either increasing or decreasing.

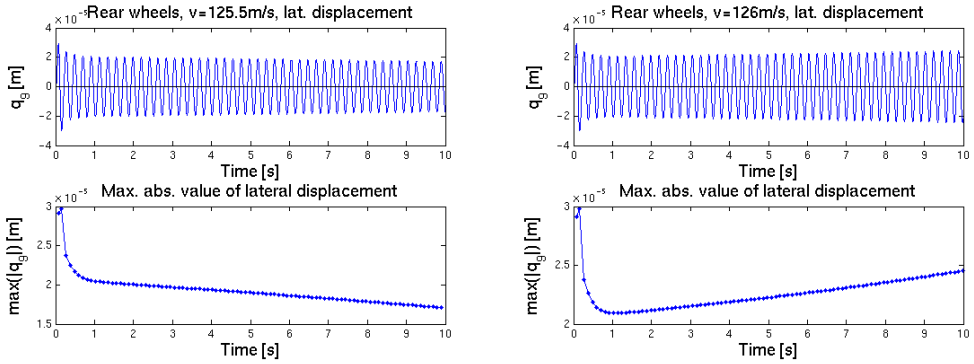

Figure 4.6: Finding the supercritical Hopf bifurcation.

Figure 4.6 shows the result of the bisection. In the left figure the bogie travels at a velocity of 125.5 m/s, and in the right figure the bogie travels at a velocity of 126 m/s. In the left figure we see that the amplitude is decreasing, so at $v = 125.5$ m/s the system is in a stable stationary solution (the green line). In the right figure we see that the amplitude is increasing, so at $v = 126$ m/s the system is in an asymptotically stable periodic symmetric solution (the red line). Therefore, we conclude that $125.5 < v_H < 126.0$ m/s.

4.4 Finding the subcritical symmetry breaking bifurcation

In this section we find the subcritical symmetry breaking bifurcation v_S, i.e. we want to find the velocity at which the system jumps from an asymptotically stable periodic symmetric solution (the red line) to an asymptotically stable periodic asymmetric solution (the blue line) in figure 4.1. Once again we use the bisection to find the subcritical symmetry breaking bifurcation. From section 4.3 we know that $v_H = 126$ m/s, and from figure 3.16 and 3.17 we know that the system is in an asymptotically stable periodic asymmetric solutions (the blue line) at a velocity of 135 m/s. Using this information, (4.1) tells us that $126 < v_S < 135$ m/s, which we may use to initialize the bisection method.

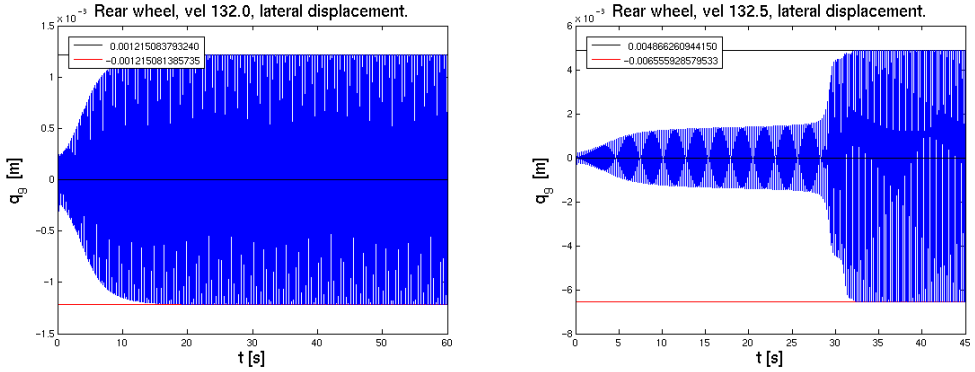

Figure 4.7: Finding the subcritical symmetry breaking bifurcation.

Figure 4.8 shows the result of the bisection. In the left figure the bogie travels at a velocity of 132 m/s, and in the right figure the bogie travels at a velocity of 132.5 m/s. In the left figure we see that the bogie oscillates around the track center, so at $v = 132$ m/s the system is in an asymptotically stable periodic symmetric solution (the red line). In the right figure we see that the hunting motion of the bogie does not oscillate around the track center, so at $v = 132.5$ m/s the system is in an asymptotically stable periodic asymmetric solutions (the blue line).

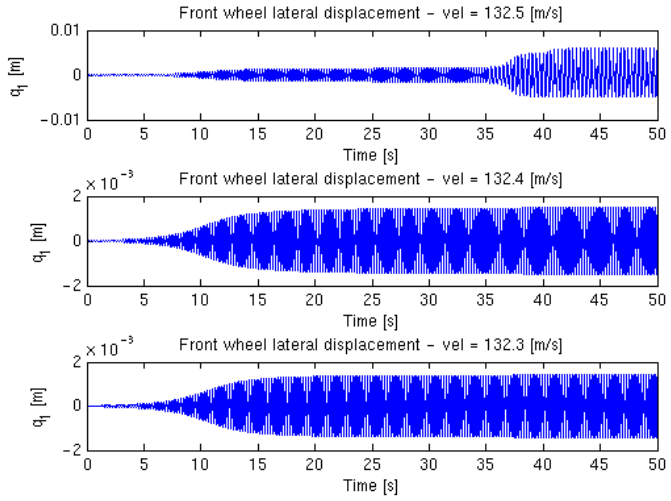

Figure 4.8: Lateral displacement at the velocities 132.3 m/s, 132.4 m/s and 132.5 m/s.

The test performed in figure 4.8 only gives a rough approximation of v_S. Therefore we do a second test. In figure 4.8 we run the bogie at three different velocities: 132.3 m/s, 132.4 m/s and 132.5 m/s. The figure shows that the bogie does not oscillate around the track center and the system is in an asymptotically stable periodic asymmetric solutions, between 132.4 m/s and 132.5 m/s. Therefore, we conclude that $132.4 < v_S < 132.5$ m/s.

Chapter 5

Conclusion

This project is a study in the nonlinear dynamics of a railway bogie using realistic wheel/rail contact. When modelling a dynamical system it is desirable to use the simplest model possible and on the same time be able to produce reliable results.

- In this model the driving properties of the whole railway vehicle is done by examination one single bogie. Therefore we only need fourteen equations describing the movements of the the railway vehicle. The basic equations is given by Kaas-Petersen [1]. When writing up the equations for the system, we have neglected all of the very small contributions from springs and dampers. In spite of the simplified model we do get reliable results, at least compared with the results obtained in [2, 8].

- A mathematical model of the system is presented and it has been shown that the same equations for the creep and normal forces apply to both the right and the left side of the wheels. The only difference is in the sign of the contact angle. The interaction forces of the system are derived as vectors and in this way we only need one function to compute the contact forces of all four wheels.

- To our knowledge it is the first time, that the model is implemented in MATLAB. When solving the ODE system we choose an explicit solver instead of an implicit solver. Regarding the CPU time used for simulations, we have no previous work to compare with. However, it is possible to solve this model in a reasonable time frame using MATLAB.

- We verify our implementation by analyzing the behaviour of the dynamical variables. This is done by several plots e.g. a plot of each principal direction (x-, y-, z-direction) of the creep and normal forces for the left and the right front wheels and by comparing our results against the results obtained in [8].

- We describe the bifurcation diagram for this dynamic system and systematically find the three critical velocities. The three velocities are related such that $v_C < v_H < v_S$. This is illustrated in the bifurcation diagram in figure 4.1.

Further work

In general, the dampers are of great influence for the driving properties of a railway vehicle. A previous study by [10] has shown that the dampers of a railway vehicle are not independent, but has great influence on each other. [10] modelled realistic nonlinear dampers, and conical wheel profiles running on circular tracks. It would be interesting to further investigate the behaviour of the nonlinear dampers using the model and the realistic wheel/rail contact implemented in this work.

Appendix A

List of symbols

$\mathbf{A}_{Tb}$	Rotation matrix changing body coordinates to track coordinates
$\mathbf{A}_{bT}$	Rotation matrix changing track coordinates to body coordinates
$\mathbf{A}_{bc}$	Rotation matrix changing contact coordinates to body coordinates
$\mathbf{A}_{cb}$	Rotation matrix changing body coordinates to contact coordinates
$\mathbf{A}_{x}^{(\alpha)}$	Rotation matrix. Around axis x with the angle α
$\mathbf{A}_{z}^{(\alpha)}$	Rotation matrix. Around axis z with the angle α
$\mathbf{a}$	Acceleration vector
a	Longitudinal semi-axis in the contact ellipse
b	Lateral semi-axis in the contact ellipse
C_{11}, C_{22}, C_{23}	Kalkers creepage coefficients
D	Dampers
E	Young's modules
$\tilde{F}_x, \tilde{F}_y$	Creep forces according til Klaker's linear theory
$\lvert\tilde{F}\rvert$	Magnitude of the creep force according to Kalker'slinear theory
$\lvert F\rvert$	Magnitude of the creep force according to Shen-Hedrick-Elkins' model
g	Contribution of gravity
G	Shear modelus
I_{wx}	M.o.i. for the roll motions of the wheel around x-axis.
I_{wy}	M.o.i. for the pitch motions of the wheel around y-axis.
I_{wz}	M.o.i. for the yaw motions of the wheel around z-axis.
I_{fx}	M.o.i for the roll motions of the frame around longitudinal axis.
I_{fz}	M.o.i. for the yaw motions of the frame around vertical axis.

Table A.1: Symbols (a-i)

$\mathbf{i}_T, \mathbf{j}_T, \mathbf{k}_T$	Base of the track coordinate system
$\mathbf{i}_b, \mathbf{j}_b, \mathbf{k}_b$	Base of the body coordinate system
$\mathbf{i}_c, \mathbf{j}_c, \mathbf{k}_c$	Base of the contact coordinate system
k	Spring stiffnesses
m_c	Mass of the carbody
m_f	Mass of the frame
m_w	Mass of the wheelset
N	Normal force between the wheel and rail
O_T	Origin of the track coordinate system
O_b	Origin of the body coordinate system
O_c	Origin of the contact coordinate system
q_{pen}	Penetration depth of the wheel into the rail
q_0	Static penetration
R_T	Track coordinate system
R_b	Body coordinate system
R_c	Contact coordinate system
$\mathbf{R}_{pen}$	Vector from the contact point on the wheel to the contact point on the rail
$\mathbf{R}_c$	Vector defining the position of the center of the mass of the wheelset
$\mathbf{R}_R$	Vector defining the contact point on the rail
$\mathbf{R}_\omega$	Vector defining the contact point on the wheel
r_0	Nominal rolling radius of the wheel
$\mathbf{v}$	Velocity vector
$\mathbf{v}_{con}$	Relative velocity between the wheel and the rail

Table A.2: Symbols (i-v)

$\dot{\beta}$	Spin perturbation
δ	Contact angle
ϵ	Adjustment factor in Shen-Hedrick-Elkins' model
μ	Friction coefficient
ν	Poisson's ratio
ξ_x	Longitudinal creep
ξ_y	Lateral creep
ξ_z	Spin creep
ϕ	Roll angle
ψ	Yaw angle
ω	The nominal spin
$\mathbf{\Omega}_b$	Angular velocity of the wheelset

Table A.3: Greek symbols

Appendix B

Rotation matrices

In this chapter I derive the transformation matrices.

Rotation around x

Following coordinate systems are considered

$$R_1 : (O_1; x_1, y_1, z_1) \quad , \quad R_2 : (O_2; x_2, y_2, z_2)$$

where R_2 is obtained by a counter-clockwise rotation of R_1 around x_1 with the angle α. The bases are related by:

$$\begin{aligned}
\mathbf{x}_1 &= (\mathbf{x}_1 \cdot \mathbf{x}_2)\mathbf{x}_2 + (\mathbf{x}_1 \cdot \mathbf{y}_2)\mathbf{y}_2 + (\mathbf{x}_1 \cdot \mathbf{z}_2)\mathbf{z}_2 = \mathbf{x}_2 \\
\mathbf{y}_1 &= (\mathbf{y}_1 \cdot \mathbf{x}_2)\mathbf{x}_2 + (\mathbf{y}_1 \cdot \mathbf{y}_2)\mathbf{y}_2 + (\mathbf{y}_1 \cdot \mathbf{z}_2)\mathbf{z}_2 = \cos\alpha\mathbf{y}_2 - \sin\alpha\mathbf{z}_2 \\
\mathbf{z}_1 &= (\mathbf{z}_1 \cdot \mathbf{x}_2)\mathbf{x}_2 + (\mathbf{z}_1 \cdot \mathbf{y}_2)\mathbf{y}_2 + (\mathbf{z}_1 \cdot \mathbf{z}_2)\mathbf{z}_2 = \sin\alpha\mathbf{y}_2 + \cos\alpha\mathbf{z}_2
\end{aligned}$$

$$\mathbf{A}_x^{(\alpha)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}$$

For the right wheels the two bases are related by:

$$\begin{aligned}
\mathbf{x}_1 &= (\mathbf{x}_1 \cdot \mathbf{x}_2)\mathbf{x}_2 + (\mathbf{x}_1 \cdot \mathbf{y}_2)\mathbf{y}_2 + (\mathbf{x}_1 \cdot \mathbf{z}_2)\mathbf{z}_2 = \mathbf{x}_2 \\
\mathbf{y}_1 &= (\mathbf{y}_1 \cdot \mathbf{x}_2)\mathbf{x}_2 + (\mathbf{y}_1 \cdot \mathbf{y}_2)\mathbf{y}_2 + (\mathbf{y}_1 \cdot \mathbf{z}_2)\mathbf{z}_2 = \cos\alpha\mathbf{y}_2 + \sin\alpha\mathbf{z}_2 \\
\mathbf{z}_1 &= (\mathbf{z}_1 \cdot \mathbf{x}_2)\mathbf{x}_2 + (\mathbf{z}_1 \cdot \mathbf{y}_2)\mathbf{y}_2 + (\mathbf{z}_1 \cdot \mathbf{z}_2)\mathbf{z}_2 = -\sin\alpha\mathbf{y}_2 + \cos\alpha\mathbf{z}_2
\end{aligned}$$

$$\mathbf{A}_x^{(\alpha)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{bmatrix}$$

Rotation around z

Following coordinate systems are considered

$$R_1 : (O_1; x_1, y_1, z_1) \quad , \quad R_2 : (O_2; x_2, y_2, z_2)$$

where R_2 is obtained by a counter-clockwise rotation of R_1 around x_1 with the angle α.

$$\begin{aligned}
\mathbf{x}_1 &= (\mathbf{x}_1 \cdot \mathbf{x}_2)\mathbf{x}_2 + (\mathbf{x}_1 \cdot \mathbf{y}_2)\mathbf{y}_2 + (\mathbf{x}_1 \cdot \mathbf{z}_2)\mathbf{z}_2 = \cos\alpha\mathbf{x}_2 - \sin\alpha\mathbf{y}_2 \\
\mathbf{y}_1 &= (\mathbf{y}_1 \cdot \mathbf{x}_2)\mathbf{x}_2 + (\mathbf{y}_1 \cdot \mathbf{y}_2)\mathbf{y}_2 + (\mathbf{y}_1 \cdot \mathbf{z}_2)\mathbf{z}_2 = \sin\alpha\mathbf{x}_2 + \cos\alpha\mathbf{y}_2 \\
\mathbf{z}_1 &= (\mathbf{z}_1 \cdot \mathbf{x}_2)\mathbf{x}_2 + (\mathbf{z}_1 \cdot \mathbf{y}_2)\mathbf{y}_2 + (\mathbf{z}_1 \cdot \mathbf{z}_2)\mathbf{z}_2 = \mathbf{z}_2
\end{aligned}$$

$$\mathbf{A}_z^{(\alpha)} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Wheel-rail contact system to body system

1. Rotation around x_b by δ

$$\mathbf{A}_{bc} = \mathbf{A}_x^{(\delta)} \quad , \quad \mathbf{A}_{cb} = \mathbf{A}_{bc}^T$$

Body system to track system

1. Rotation around z_T by ψ (yaw)
2. Rotation around x_T by ϕ (roll)

$$\mathbf{A}_{Tb} = \mathbf{A}_z^{(\psi)}\mathbf{A}_x^{(\phi)} \quad , \quad \mathbf{A}_{bT} = \mathbf{A}_{Tb}^T$$

Rotationmatrices

The following matrices is simplified using trigonometry.

Right/Left wheel	$\mathbf{A}_{Tb}$	$=$	$\mathbf{A}_z^{(\psi)}\mathbf{A}_x^{(\phi)}$
		$=$	$\begin{bmatrix} c\psi & -s\psi c\phi & s\psi s\phi \\ s\psi & c\psi c\phi & -c\psi s\phi \\ 0 & s\phi & c\phi \end{bmatrix}$
Left wheel	$\mathbf{A}_{bc}$	$=$	$\mathbf{A}_x^{(\delta)}$
		$=$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & c\delta & -s\delta \\ 0 & s\delta & c\delta \end{bmatrix}$
	$\mathbf{A}_{Tc}$	$=$	$\mathbf{A}_{Tb}\mathbf{A}_{bc}$
		$=$	$\begin{bmatrix} c\psi & -s\psi c(\phi+\delta) & s\psi s(\phi+\delta) \\ s\psi & c\psi c(\phi+\delta) & -c\psi s(\phi+\delta) \\ 0 & s(\phi+\delta) & c(\phi+\delta) \end{bmatrix}$
Right wheel	$\mathbf{A}_{bc}$	$=$	$\mathbf{A}_x^{(\delta)}$
		$=$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & c\delta & s\delta \\ 0 & -s\delta & c\delta \end{bmatrix}$
	$\mathbf{A}_{Tc}$	$=$	$\mathbf{A}_{Tb}\mathbf{A}_{bc}$
		$=$	$\begin{bmatrix} c\psi & -s\psi c(\phi-\delta) & s\psi s(\phi-\delta) \\ s\psi & c\psi c(\phi-\delta) & -c\psi s(\phi-\delta) \\ 0 & s(\phi-\delta) & c(\phi-\delta) \end{bmatrix}$

Table B.1: Rotationmatrices. Here, c = cos and s = sin

Right/Left wheel	$\mathbf{A}_{Tb}$	$=$	$\mathbf{A}_z^{(\psi)}\mathbf{A}_x^{(\phi)}$
			$\begin{bmatrix} 1 & -\psi & 0 \\ \psi & 1 & -\phi \\ 0 & \phi & 1 \end{bmatrix}$
		$=$	$\begin{bmatrix} 1 & -\psi & 0 \\ \psi & 1 & -\phi \\ 0 & \phi & 1 \end{bmatrix}$
Left wheel	$\mathbf{A}_{bc}$	$=$	$\mathbf{A}_x^{(\delta)}$
			$\begin{bmatrix} 1 & 0 & 0 \\ 0 & c\delta & -s\delta \\ 0 & s\delta & c\delta \end{bmatrix}$
		$=$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & c\delta & -s\delta \\ 0 & s\delta & c\delta \end{bmatrix}$
	$\mathbf{A}_{Tc}$	$=$	$\mathbf{A}_{Tb}\mathbf{A}_{bc}$
			$\begin{bmatrix} 1 & -\psi c(\phi+\delta) & \psi s(\phi+\delta) \\ \psi & c(\phi+\delta) & -s(\phi+\delta) \\ 0 & s(\phi+\delta) & c(\phi+\delta) \end{bmatrix}$
		$=$	$\begin{bmatrix} 1 & -\psi c(\phi+\delta) & \psi s(\phi+\delta) \\ \psi & c(\phi+\delta) & -s(\phi+\delta) \\ 0 & s(\phi+\delta) & c(\phi+\delta) \end{bmatrix}$
Right wheel	$\mathbf{A}_{bc}$	$=$	$\mathbf{A}_x^{(\delta)}$
			$\begin{bmatrix} 1 & 0 & 0 \\ 0 & c\delta & s\delta \\ 0 & -s\delta & c\delta \end{bmatrix}$
		$=$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & c\delta & s\delta \\ 0 & -s\delta & c\delta \end{bmatrix}$
	$\mathbf{A}_{Tc}$	$=$	$\mathbf{A}_{Tb}\mathbf{A}_{bc}$
			$\begin{bmatrix} 1 & -\psi c(\phi-\delta) & \psi s(\phi-\delta) \\ \psi & c(\phi-\delta) & -s(\phi-\delta) \\ 0 & s(\phi-\delta) & c(\phi-\delta) \end{bmatrix}$
		$=$	$\begin{bmatrix} 1 & -\psi c(\phi-\delta) & \psi s(\phi-\delta) \\ \psi & c(\phi-\delta) & -s(\phi-\delta) \\ 0 & s(\phi-\delta) & c(\phi-\delta) \end{bmatrix}$

Table B.2: Approximate rotationmatrices. Here, c = cos and s = sin

Appendix C

Data from RSGEO table

To make it easier to understand the code, we give a brief description of the data from RSGEO and how the table is used, respectively for the left wheels and the right wheels. Subsequently, the first five lines of the RSGEO table are listed. Furthermore we plot all RSGEO values as a function of the lateral displacement for both the left and the right wheel.

Description	Left wheel	Right wheel
y	Given directly from RSGEO	Flipped around the z-axis
N0	Given directly from RSGEO	Flipped around the z-axis
δ	Given directly from RSGEO	Flipped around the z-axis and the y-axis
a0	Given directly from RSGEO	Flipped around the z-axis
b0	Given directly from RSGEO	Flipped around the z-axis
Rwy	Given directly from RSGEO	Flipped around the z-axis and the y-axis
Rwz	Flipped around the y-axis	Flipped around the z-axis and the y-axis
C11	Given directly from RSGEO	Flipped around the z-axis
C22	Given directly from RSGEO	Flipped around the z-axis
C23	Given directly from RSGEO	Flipped around the z-axis
Rrz	Given directly from RSGEO	Flipped around the z-axis
q0	Given directly from RSGEO	Flipped around the z-axis
Rry	Given directly from RSGEO	Flipped around the z-axis and the y-axis

y [m]	N0 [N]	δ [rad]	a0 [m]
-0.01700	66604.4707017493	0.043652537	0.00656931
-0.01699	66604.4385295224	0.043663594	0.00656874
-0.01698	66604.4174969331	0.043670821	0.00656819
-0.01697	66604.4075862261	0.043674226	0.00656766
-0.01696	66604.4087883589	0.043673813	0.00656717
	b0 [m]	Rwy [m]	Rwz [m]
	0.00352271	0.775173707	0.42454303
	0.00352389	0.775167656	0.42454328
	0.00352509	0.775161599	0.42454354
	0.00352628	0.775155535	0.42454379
	0.00352748	0.775149464	0.42454404
	C11 [-]	C22 [-]	C23 [-]
	5.07979826325327	4.80394451402366	2.49187187388858
	5.07904342362409	4.80296585757342	2.49091799021527
	5.07829381464116	4.80199370040575	2.48998163323036
	5.07754943371750	4.80102804220109	2.48904108418390
	5.07680484571060	4.80005937879926	2.48811338946733
	Rrz [m]	qpen0 [m]	Rry [m]
	-0.000052853178	0.000118493	0.7518796300
	-0.000052816988	0.000118492	0.7518861589
	-0.000052780971	0.000118491	0.7518926839
	-0.000052744996	0.000118489	0.7518992057
	-0.000052708995	0.000118488	0.7519057244

Table C.1: The first five lines of date from the RSGEO table.

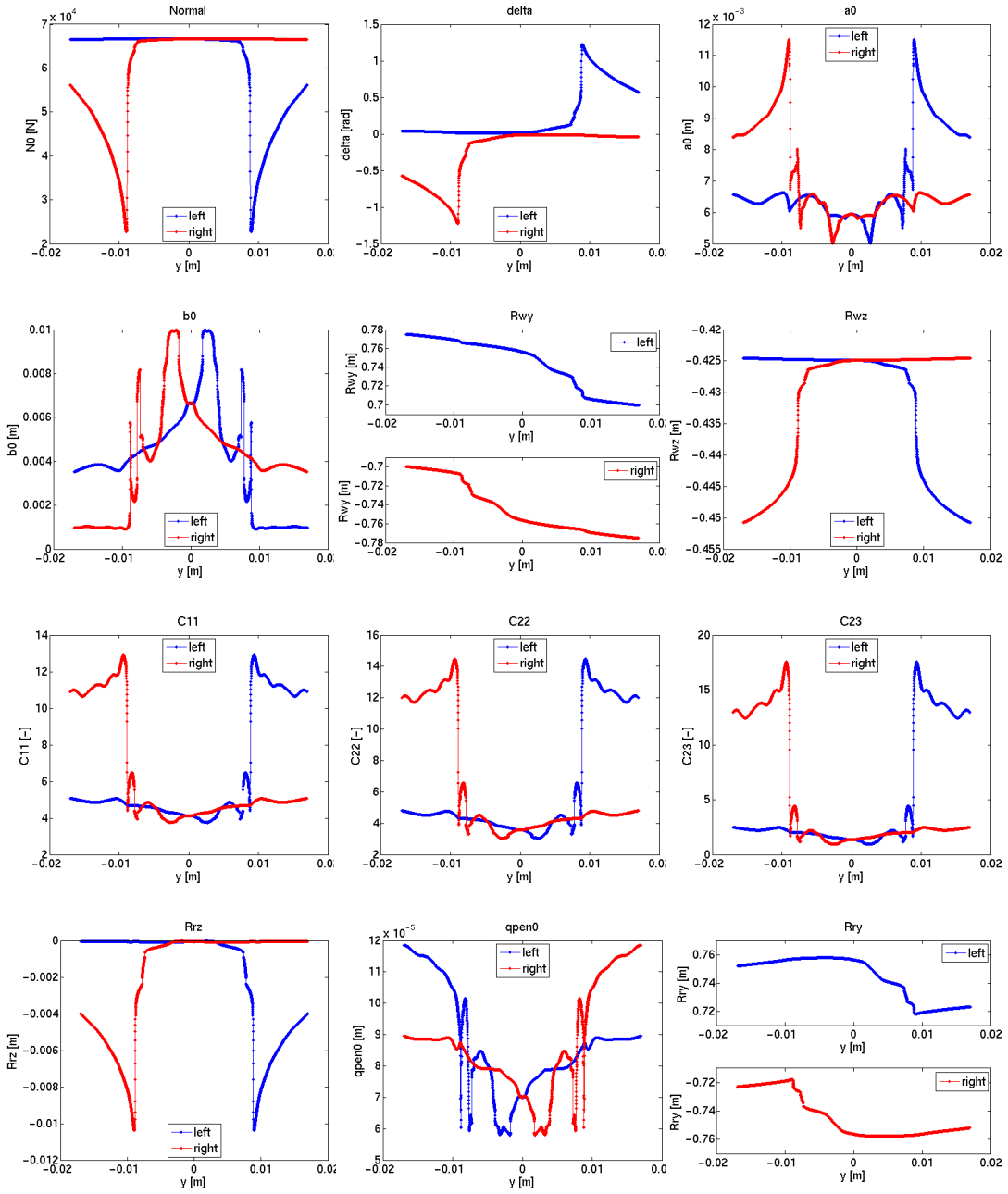

Figure C.1: RSGEO values for the left and the right wheel as a function of the lateral displacement of the wheelset.

Appendix D

Test of components separately

In the following, we test, if the rear wheels are guided into the center of the track. We let the bogie travel at a velocity of 135 m/s, and simultaneously we give a lateral disturbance of 10^{-4} m (figure D.1), a disturbance of the yaw angle of 10^{-4} rad (figure D.3) and a disturbance of the roll angle of 10^{-4} rad (figure D.4). Because the wheels are constrained to be in contact with the rails through the wheel/rail contact point, we do not give any vertical disturbance to the wheels (figure D.2).

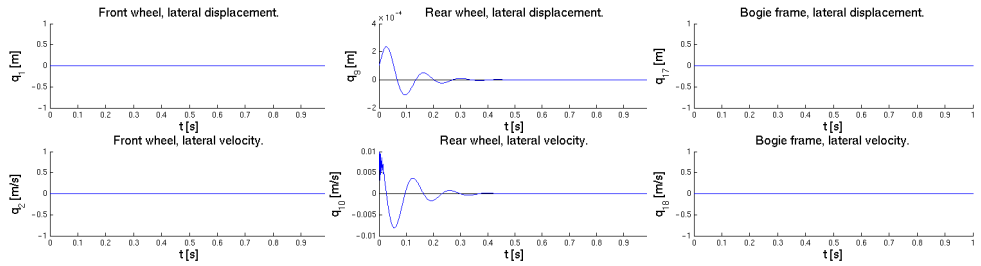

Figure D.1: Lateral disturbance of the rear wheels of 10^{-4} m with fixed front wheels and bogie frame.

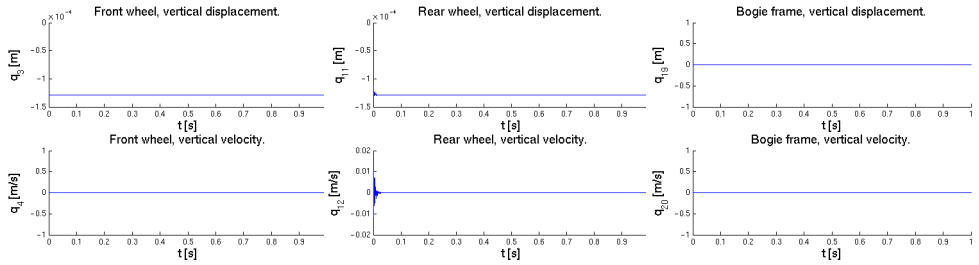

Figure D.2: No vertical disturbance of the rear wheels and fixed front wheels and bogie frame.

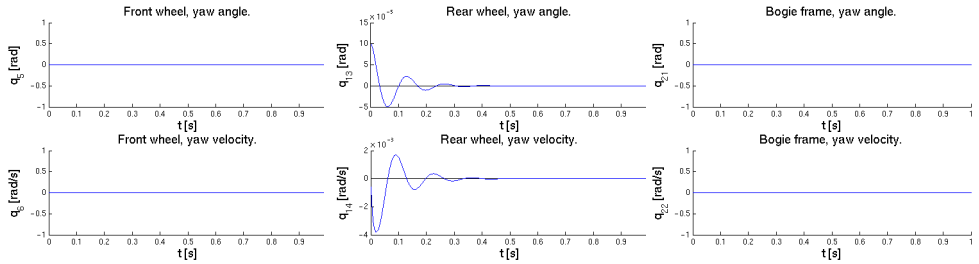

Figure D.3: Disturbance of the yaw angle of 10^{-4} rad with fixed front wheels and bogie frame.

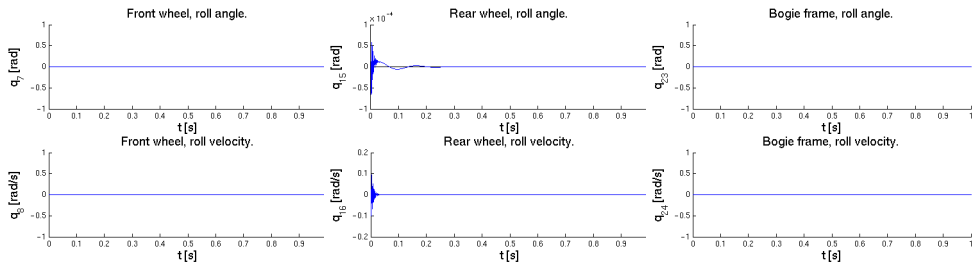

Figure D.4: Disturbance of the roll angle of 10^{-4} rad with fixed front wheels and bogie frame.

In the following, we test, if the bogie are guided into the center of the track. We let the bogie travel at a velocity of 135 m/s, and simultaneously we give a lateral disturbance of 10^{-4} m (figure D.5), a disturbance of the yaw angle of 10^{-4} rad (figure D.7) and a disturbance of the roll angle of 10^{-4} rad (figure D.8). Because the wheels are constrained to be in contact with the rails through the wheel/rail contact point, we do not give any vertical disturbance to the wheels (figure D.6).

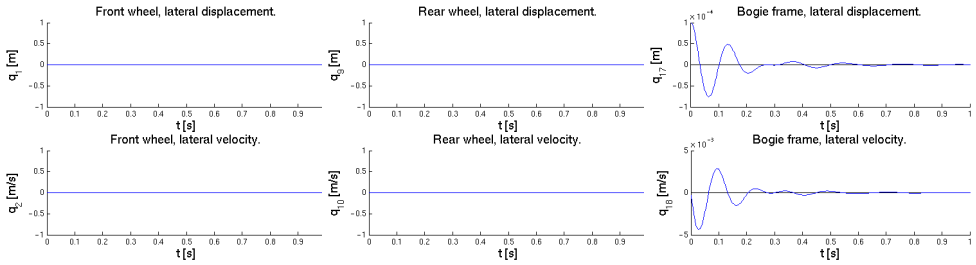

Figure D.5: Lateral disturbance of the bogie wheels of 10^{-4} m with fixed.

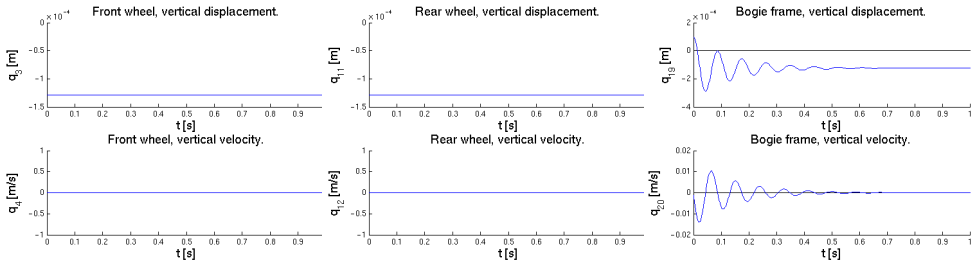

Figure D.6: No vertical disturbance of the bogie and with fixed wheels.

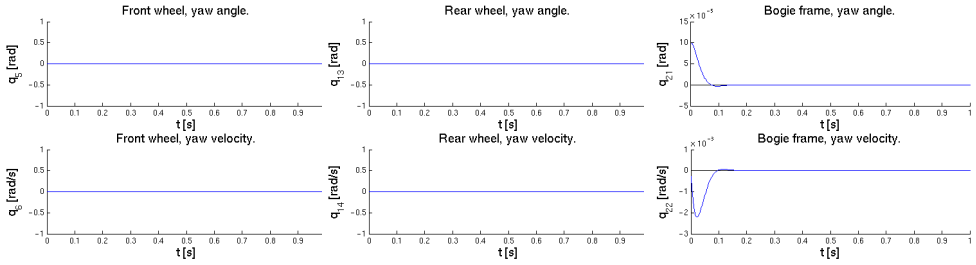

Figure D.7: Disturbance of the yaw angle of 10^{-4} rad with fixed wheels.

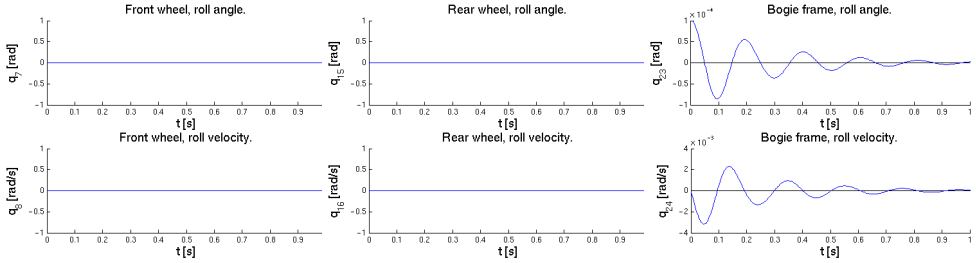

Figure D.8: Disturbance of the roll angle of 10^{-4} rad with fixed wheels.

In the following, we test, if the rear wheels are in the center of the track but this time without the contribution from normal and creep forces. Once again we let the bogie travel at a velocity of 135 m/s, and simultaneously we give a lateral disturbance of 10^{-4} m (figure D.9), a disturbance of the yaw angle of 10^{-4} rad (figure D.11) and a disturbance of the roll angle of 10^{-4} rad (figure D.12). Once again we do not give any vertical disturbance to the wheels (figure D.10).

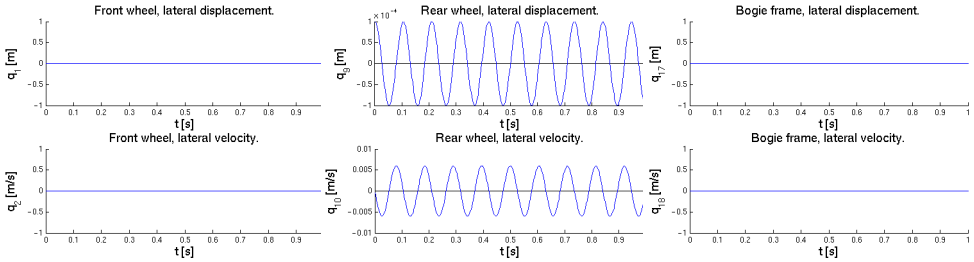

Figure D.9: Lateral disturbance of the rear wheels of 10^{-4} m with fixed front wheels and bogie frame and without normal and creep forces.

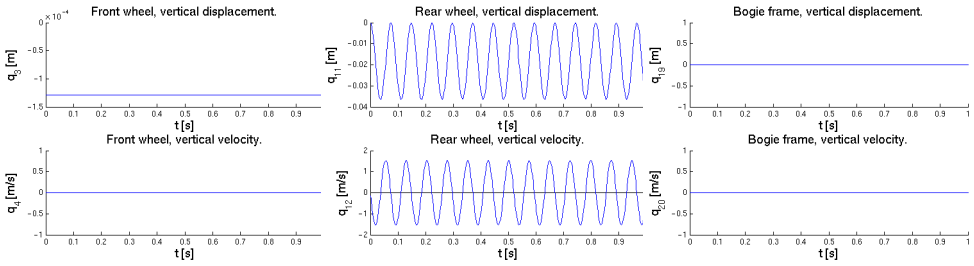

Figure D.10: No vertical disturbance of the rear wheels and with fixed front wheels and bogie frame and without normal and creep forces.

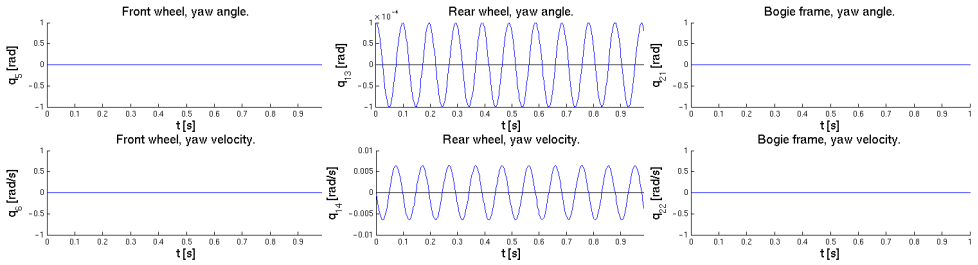

Figure D.11: Disturbance of the yaw angle of 10^{-4} rad with fixed front wheels and bogie frame and without normal and creep forces.

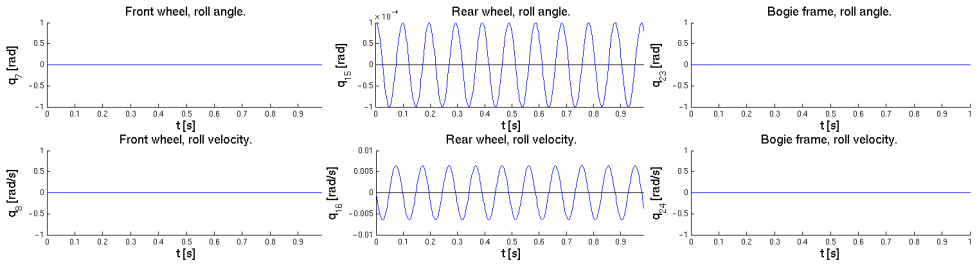

Figure D.12: Disturbance of the roll angle of 10^{-4} rad with fixed front wheels and bogie frame and without normal and creep forces.

In the following, we test, if the bogie are in the center of the track but this time without the contribution from normal and creep forces. Once again we let the bogie travel at a velocity of 135 m/s, and simultaneously we give a lateral disturbance of 10^{-4} m (figure D.13), a disturbance of the yaw angle of 10^{-4} rad (figure D.15) and a disturbance of the roll angle of 10^{-4} rad (figure D.16). Once again we do not give any vertical disturbance to the wheels (figure D.14).

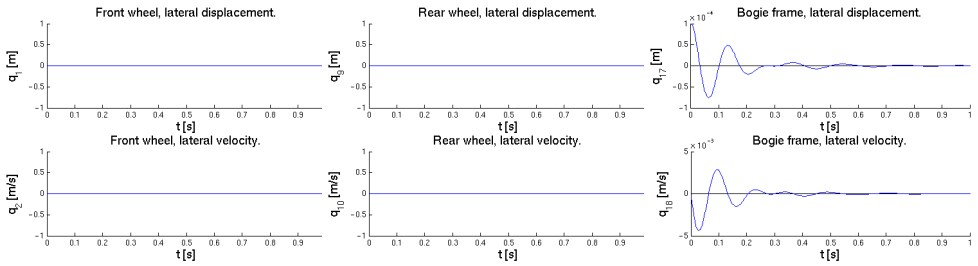

Figure D.13: Lateral disturbance of the rear wheels of 10^{-4} m with fixed wheels and without normal and creep forces.

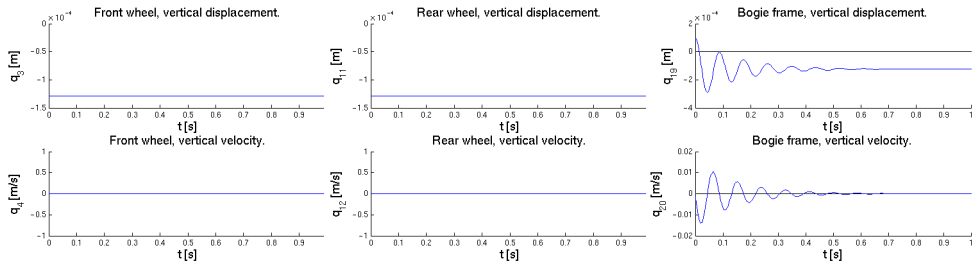

Figure D.14: No vertical disturbance of the rear wheels and with fixed wheels and without normal and creep forces.

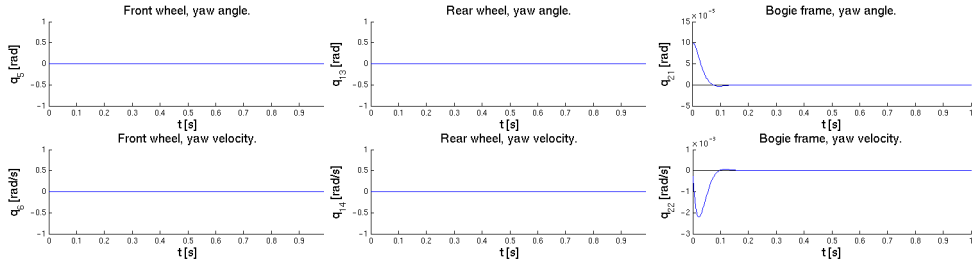

Figure D.15: Disturbance of the yaw angle of 10^{-4} rad with fixed wheels and without normal and creep forces.

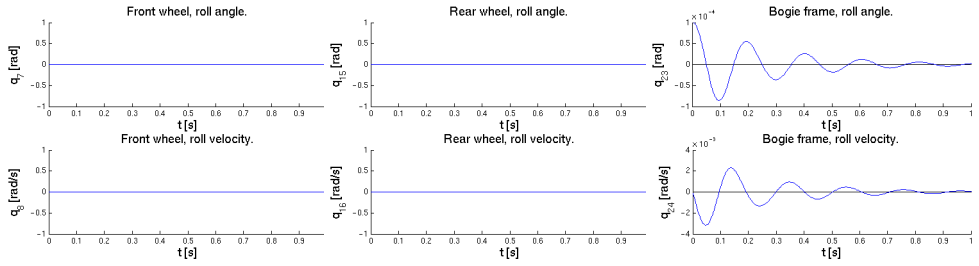

Figure D.16: Disturbance of the roll angle of 10^{-4} rad with fixed wheels and without normal and creep forces.

Appendix E

Matlab code

E.1 sol.m

```
clear all
close all
clc

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% UI                                                                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% time
tspan = [0,20];
% velocity
vel = 132;

% opts = [];
% opts = odeset('RelTol',1e-3,'AbsTol',1e-4);
% opts = odeset('RelTol',1e-4,'AbsTol',1e-5);
opts = odeset('RelTol',1e-6,'AbsTol',1e-6);
% opts = odeset('RelTol',1e-6,'AbsTol',1e-8);
% opts = odeset('RelTol',1e-8,'AbsTol',1e-10);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Data from RSGEO                                                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
load rsgeo.dat
z_start = -1.2916e-04;          % static penetration
p = 1e-3;                       % perturbation

% initial values:
qstart = [p,0,z_start,0,0,0,0,0,... % front wheel
          0,0,z_start,0,0,0,0,0,... % rear wheel
          0,0,0,0,0,0,0,0,...       % bogie frame
          0,0];                     % spin perturbation
% Split the bogie in parts...
% qstart = [p,0,z_start,0,p,0,p,0,... % front wheel
%           0,0,z_start,0,0,0,0,0,... % rear wheel
%           0,0,0,0,0,0,0,0,...       % bogie frame
```

```
%           0,0];                       % spin perturbation
% qstart = [0,0,z_start,0,0,0,0,0,... % front wheel
%           p,0,z_start,0,p,0,p,0,... % rear wheel
%           0,0,0,0,0,0,0,0,...       % bogie frame
%           0,0];                     % spin perturbation
% qstart = [0,0,z_start,0,0,0,0,0,... % front wheel
%           0,0,z_start,0,0,0,0,0,... % rear wheel
%           p,0,p,0,p,0,p,0,...       % bogie frame
%           0,0];                     % spin perturbation

% q(1)  : [m]     Front wheel lateral position
% q(2)  : [m/s]   Front wheel lateral velocity
% q(3)  : [m]     Front wheel vertical position
% q(4)  : [m/s]   Front wheel vertical velocity
% q(5)  : [rad]   Front wheel yaw angle
% q(6)  : [rad/s] Front wheel yaw angle velocity
% q(7)  : [rad]   Front wheel roll angle
% q(8)  : [rad/s] Front wheel roll angle velocity
% q(9)  : [m]     Rear wheel lateral position
% q(10) : [m/s]   Rear wheel lateral velocity
% q(11) : [m]     Rear wheel vertical position
% q(12) : [m/s]   Rear wheel vertical velocity
% q(13) : [rad]   Rear wheel yaw angle
% q(14) : [rad/s] Rear wheel yaw angle velocity
% q(15) : [rad]   Rear wheel roll angle
% q(16) : [rad/s] Rear wheel roll angle velocity
% q(17) : [m]     Bogie frame lateral position
% q(18) : [m/s]   Bogie frame lateral velocity
% q(19) : [m]     Bogie frame vertical position
% q(20) : [m/s]   Bogie frame vertical velocity
% q(21) : [rad]   Bogie frame yaw angle
% q(22) : [rad/s] Bogie frame yaw angle velocity
% q(23) : [rad]   Bogie frame roll angle
% q(24) : [rad/s] Bogie frame roll angle velocity
% q(25) : [rad/s] Spin perturbation front axel
% q(26) : [rad/s] Spin perturbation rear axel

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Bogie and wheel characteristics                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
parm.a    = 0.7563006418; % [m] Half the track gauge
parm.b    = 1.074;        % [m] Half the distance between the two
    axel
parm.d1   = 0.620;        % [m] Horizontal distance from spring to
     gravity
parm.d2   = 0.680;        % [m] Horizontal distance from spring to
     gravity
parm.h1   = 0.0762;       % [m] Vertiacl distance from spring to
    gravity
parm.h2   = 0.6584;       % [m] Vertiacl distance from spring to
    gravity
parm.mw   = 1022;         % [kg] Mass of the wheel axle
parm.Iwx  = 678;          % [kg*m^2] M.o.i for the roll motions of
     the wheel around the x-axis
```

```
parm.Iwy   = 80;            % [kg*m^2] M.o.i for the pitch motions of the wheel around the y-axis
parm.Iwz   = 678;           % [kg*m^2] M.o.i for the yaw motions of the wheel around the z-axis
parm.mf    = 2918;          % [kg] Mass of the frame
parm.mc    = 44388;         % [kg] Mass of the car body
parm.Ifx   = 6780;          % [kg*m^2] M.o.i for the roll motions of the frame around the x-axis
parm.Ifz   = 6780;          % [kg*m^2] M.o.i for the yaw motions of the frame around the z-axis
parm.r0    = 0.4248829;     % [m] Nominal roling radius of the wheel
parm.k1    = 1823e03;       % [N/m] Horizontal spring, wheel-frame, lateral
parm.k2    = 3646e03;       % [N/m] Horizontal spring, wheel-frame, longitudinal
parm.k3    = 3646e03;       % [N/m] Vertical spring, wheel-frame
parm.k4    = 182.3e03;      % [N/m] Horizontal spring, frame-carbody, lateral
parm.k5    = 333.3e03;      % [N/m] Vertical spring, frame-carbody
parm.k6    = 2710e03;       % [Nm] Torsion spring, frame-carbody
parm.D1    = 20e03;         % [Ns/m] Vertical damper, frame-carbody
parm.D2    = 29.2e03;       % [Ns/m] Horizontal damper, frame-carbody, lateral
parm.D3    = 500e03;        % [Ns/m] Horizontal damper, frame-carbody, lateral
parm.g     = 9.82;          % [m/s^2] Gravitational acceleration
parm.v     = vel;           % [m/s] Velocity in the direction of travel
parm.Fwfc = (parm.mw+1/2*parm.mf+1/4*parm.mc)*parm.g; % [N] Gravitational load for one wheel axel
parm.Dm    = 1.5e05;        % [Ns/m]   Material damping force of wheel-rail contact
% Linear decrease/increas off velocity in creep_force.m:
parm.v0    = parm.v;        %
parm.a0    = 0;%-0.05;      %
parm.t0    = 0;             %

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Material properties                                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
parm.E  = 2.1e11;                 % [N/m^2]   Youngs modulus
parm.nu = 0.27;                   % [-]       Poissions ratio
parm.G  = parm.E/(2*(1+parm.nu)); % [N/m^2]   Shear modulus
parm.mu = 0.15;                   % [-]       Adhesions coefficient

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Solving the system of equations                                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% calling function system
tic
% [T,q,info,perf,err] = erk(@bogie,tspan,qstart,2^-11.5,0,[ ],parm,rsgeo); % from Carsten, where you can define the stepsize and tol
```

```
[T,q] = ode45(@bogie,tspan,qstart,opts,parm,rsgeo); %  This should be the first solver you try ~ erk with buttab = '45DP' Dormand Prince
% [T,q] = ode15s(@bogie,tspan,qstart,opts,parm,rsgeo); % If ode45 is slow because the problem is stiff.
toc

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot q                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
plot_result(T,q,'front')
plot_result(T,q,'rear')
plot_result(T,q,'frame')
plot_hyste
plot_normal
plot_creep
envelope
compare_maxval
```

E.2 bogie.m

```
function dq = bogie(t,q,parm,rsgeo)

% BOGIE
%
% Call: bogie(t,q,parm,rsgeo)
%
% Input Parameters: t, is the time.
%                   q, array containg the position end the
%                       velocity of the wheels and the bogie.
%                   parm, array containing the position and the
%                       velocity of both wheels and the bogie.
%                   rsgeo, table containing precomputed parameters
%
% Output Parameter: dq, the solution

% Authors: % Ulla Uldahl
% Date:    % September 22, 2011

if t > ceil(t)-0.001
    disp(['Time: ',num2str(t)])
end

% Read parameters:
% mw      = parm.mw;   % [kg]       Mass of the wheel axle
% Iwx     = parm.Iwx;  % [kg*m^2] M.o.i. for the roll motions of the wheel around the x-axis
% Iwy     = parm.Iwy;  % [kg*m^2] M.o.i. for the pitch motions of the wheel around the x-axis
% Iwz     = parm.Iwz;  % [kg*m^2] M.o.i. for the yaw motions of the wheel around the z-axis
```

```
28 % mf    = parm.mf;  % [kg]      Mass of the frame
29 % Ifx   = parm.Ifx; % [kg*m^2] M.o.i. for the yaw motions of the
       frame around the x-axis
30 % Ifz   = parm.Ifz; % [kg*m^2] M.o.i. for the roll motions of the
       frame around the z-axis
31 % a     = parm.a;   % [m]       Half the track gauge
32 % b     = parm.b;   % [m]       Half the axle distance
33 % h1    = parm.h1;  % [m]       Vertiacl distance fro spring to
       gravety
34 % h2    = parm.h2;  % [m]       Vertiacl distance fro spring to
       gravety
35 % r0    = parm.r0;  % [m]       Nominal roling radius of the wheel
36 % Fwfc  = parm.Fwfc;% [N]       Gravitational load for one wheel
       axel, Fwfc = (mw+1/2*mf+1/4*mc)*g
37
38 % Allocate space:
39 dq = zeros(26,1);
40
41 % Compute spring forces:
42 A = spring_force(t,q,parm);
43
44 % Compute damper forces:
45 D = damper_force(t,q,parm);
46
47 % Compute normal forces:
48 if 1 % used debugging
49     [Nfl,Nzfl,qpenfl,Rwyfl,Rwzfl] = normal_force(t,q,parm,rsgeo,'
           front','left');
50     [Nfr,Nzfr,qpenfr,Rwyfr,Rwzfr] = normal_force(t,q,parm,rsgeo,'
           front','right');
51     [Nrl,Nzrl,qpenrl,Rwyrl,Rwzrl] = normal_force(t,q,parm,rsgeo,'
           rear','left');
52     [Nrr,Nzrr,qpenrr,Rwyrr,Rwzrr] = normal_force(t,q,parm,rsgeo,'
           rear','right');
53 else
54     Nfl = zeros(3,1);
55     Nfr = zeros(3,1);
56     Nrl = zeros(3,1);
57     Nrr = zeros(3,1);
58     Rwyfl = parm.a;
59     Rwyfr = -parm.a;
60     Rwyrl = parm.a;
61     Rwyrr = -parm.a;
62     Rwzfl = -parm.r0;
63     Rwzfr = -parm.r0;
64     Rwzrl = -parm.r0;
65     Rwzrr = -parm.r0;
66 end
67
68 % Compute creep forces:
69 if 1 % used debugging
70     FCfl = creep_force(t,q,parm,Nzfl,qpenfl,rsgeo,'front','left');
71     FCfr = creep_force(t,q,parm,Nzfr,qpenfr,rsgeo,'front','right')
           ;
```

```
72      FCrl = creep_force(t,q,parm,Nzrl,qpenrl,rsgeo,'rear','left');
73      FCrr = creep_force(t,q,parm,Nzrr,qpenrr,rsgeo,'rear','right');
74  else
75      FCfl = zeros(3,1);
76      FCfr = zeros(3,1);
77      FCrl = zeros(3,1);
78      FCrr = zeros(3,1);
79  end
80
81  if 1 % used debugging
82      % Compute dynamics of front wheel:
83      dq(1)  = q(2); % [m] lateral position (q1)
84      dq(2)  = (-A(1)+FCfl(2)+FCfr(2)+Nfl(2)+Nfr(2))/parm.mw; % [m/s
            ] lateral velocity (q1dot) -(A1)/mw
85      dq(3)  = q(4); % [m] vertical position (q2)
86      dq(4)  = (-A(2)+FCfl(3)+FCfr(3)+Nfl(3)+Nfr(3)-parm.Fwfc)/parm.
            mw; % [m/s] vertical velocity (q2dot) -(A2)/mw
87      dq(5)  = q(6); % [rad] yaw angle (q3)
88      dq(6)  = (-A(3)-Rwyfl*(FCfl(1)+q(5)*(FCfl(2)+Nfl(2)))-Rwyfr*(
            FCfr(1)+q(5)*(FCfr(2)+Nfr(2))))/parm.Iwz; % [rad/s] yaw
            angle velocity (q3dot) -(A3)/Iwz
89      dq(7)  = q(8); % [rad] roll angle (q4)
90      dq(8)  = (-A(4)+Rwyfl*(FCfl(3)+Nfl(3)-q(7)*(FCfl(2)+Nfl(2)))+
            Rwyfr*(FCfr(3)+Nfr(3)-q(7)*(FCfr(2)+Nfr(2))) ...
91              -Rwzfl*(-q(5)*FCfl(1)+FCfl(2)+Nfl(2)+q(7)*(FCfl(3)+Nfl
                    (3)))-Rwzfr*(-q(5)*FCfr(1)+FCfr(2)+Nfr(2)+q(7)*(
                    FCfr(3)+Nfr(3))))/parm.Iwx; % [rad/s] roll angle
                    velocity (q4dot) -(A4)/Iwx
92      dq(25) = (Rwzfr*(FCfr(1)+(FCfr(2)+Nfr(2))*q(5))+Rwzfl*(FCfl(1)
            +(FCfl(2)+Nfl(2))*q(5))-A(13))/parm.Iwy; % [rad] Spin
            pertubation front wheel
93  end
94  if 1 % used debugging
95      % Compute dynamics of rear wheel:
96      dq(9)  = q(10); % [m] lateral position (q5)
97      dq(10) = (-A(5)+FCrl(2)+FCrr(2)+Nrl(2)+Nrr(2))/parm.mw; % [m/s
            ] lateral velocity (q5dot) -(A5)/mw
98      dq(11) = q(12); % [m] vertical position (q6)
99      dq(12) = (-A(6)+FCrl(3)+FCrr(3)+Nrl(3)+Nrr(3)-parm.Fwfc)/parm.
            mw; % [m/s] vertical velocity (q6dot) -(A6)/mw
100     dq(13) = q(14); % [rad] yaw angle (q7)
101     dq(14) = (-A(7)-Rwyrl*(FCrl(1)+q(13)*(FCrl(2)+Nrl(2)))-Rwyrr*(
            FCrr(1)+q(13)*(FCrr(2)+Nrr(2))))/parm.Iwz; % [rad/s] yaw
            angle velocity (q3dot) -(A3)/Iwz
102     dq(15) = q(16); % [rad] roll angle (q8)
103     dq(16) = (-A(8)+Rwyrl*(FCrl(3)+Nrl(3)-q(15)*(FCrl(2)+Nrl(2))*q
            (15))+Rwyrr*(FCrr(3)+Nrr(3)-q(15)*(FCrr(2)+Nrr(2))) ...
104             -Rwzrl*(-q(13)*FCrl(1)+FCrl(2)+Nrl(2)+q(15)*(FCrl(3)+
                    Nrl(3)))-Rwzrr*(-q(13)*FCrr(1)+FCrr(2)+Nrr(2)+q
                    (15)*(FCrr(3)+Nrr(3))))/parm.Iwx; % [rad/s] roll
                    angle velocity (q4dot) -(A4)/Iwx
105     dq(26) = (Rwzrr*(FCrr(1)+(FCrr(2)+Nrr(2))*q(13))+Rwzrl*(FCrl
            (1)+(FCrl(2)+Nrl(2))*q(13))-A(14))/parm.Iwy; % [rad] Spin
            pertubation rear wheel
```

```
106 end
107 if 1 % used debugging
108     % Compute dynamics of bogie frame
109     dq(17) = q(18); % [m] lateral position (q9)
110     dq(18) = (A(1)+A(5)+A(9)+D(1))/parm.mf; % [m/s] lateral
            velocity (q9dot) (A1+A5+A9+D1)/mf
111     dq(19) = q(20); % [m] vertical position (q10)
112     dq(20) = (A(2)+A(6)-A(10)-D(2))/parm.mf; % [m/s] vertical
            velocity (q10dot) (A2+A6-A10+D2))/mf
113     dq(21) = q(22); % [rad] yaw angle (q11)
114     dq(22) = (parm.b*A(1)+A(3)-parm.b*A(5)+A(7)-A(11)-D(3))/parm.
            Ifz; % [rad/s] yaw angle velocity (q11dot) (bA1+A3-bA5+A7
            -A11)/Ifz
115     dq(23) = q(24); % [rad] roll angle (q12)
116     dq(24) = (parm.h1*A(1)+A(4)+parm.h1*A(5)+A(8)-parm.h2*A(9)-A
            (12)-D(4)-D(5))/parm.Ifx; % [rad/s] roll angle velocity (
            q12dot) (h1A1+h1A5-h2A9-A12-D3+D4)/Ifx
117 end
118 end
```

E.3 spring_force.m

```
1  function A = spring_force(t,q,parm)
2  % SPRING_FORCE used for calculating the current spring force
3  %
4  % Call: spring_force(t,q,parm)
5  %
6  % Input Parameters: t, is the time
7  %                   q, array containg the position end the
8  %                       velocity of the wheels and the bogie
9  %                   parm, array containing the position and the
10 %                       velocity of both wheels and the bogie
11 %
12 % Output Parameter:  Array with current spring force
13
14 % Authors: % Ulla Uldahl
15 % Date:    % September 24, 2011
16
17 % Read parameters:
18 % b     = parm.b;  % [m]   Half the distance between the two axels
19 % d1    = parm.d1; % [m]   Horizontal distance fro spring to
       gravety
20 % d2    = parm.d2; % [m]   Horizontal distance fro spring to
       gravety
21 % h1    = parm.h1; % [m]   Vertiacl distance fro spring to gravety
22 % h2    = parm.h2; % [m]   Vertiacl distance fro spring to gravety
23 % k1    = parm.k1; % [N/m] Spring coefficient wheel/frame lateral
24 % k2    = parm.k2; % [N/m] Spring coefficient wheel/frame track-
       direction
25 % k3    = parm.k3; % [N/m] Spring coefficient wheel/frame vertical
```

```
% k4    = parm.k4; % [N/m] Spring coefficient frame/carbody
    lateral
% k5    = parm.k5; % [N/m] Spring coefficient frame/carbody
    vertical
% k6    = parm.k6; % [Nm]  Spring coefficient frame/carbody yaw (
    torsion)

% Allocate space:
A = zeros(14,1);

% Compute spring forces:
A(1)  = 2*parm.k1*(q(1)-q(17)-parm.b*q(21)-parm.h1*q(23)); % [N]
A(2)  = 2*parm.k3*(q(3)-q(19)); % [N]
A(3)  = 2*parm.k2*parm.d1*parm.d1*(q(5)-q(21)); % [Nm]
A(4)  = 2*parm.k3*parm.d1*parm.d1*(q(7)-q(23)); % [Nm]
A(5)  = 2*parm.k1*(q(9)-q(17)+parm.b*q(21)-parm.h1*q(23)); % [N]
A(6)  = 2*parm.k3*(q(11)-q(19)); % [N]
A(7)  = 2*parm.k2*parm.d1*parm.d1*(q(13)-q(21)); % [Nm]
A(8)  = 2*parm.k3*parm.d1*parm.d1*(q(15)-q(23)); % [Nm]
A(9)  = 2*parm.k4*(parm.h2*q(23)-q(17)); % [N]
A(10) = 2*parm.k5*q(19); % [N]
A(11) = parm.k6*q(21); % [Nm]
A(12) = 2*parm.k5*parm.d2*parm.d2*q(23); % [Nm]
A(13) = 2*parm.k3*parm.d1*parm.d1*q(5)*q(23); % [Nm]
A(14) = 2*parm.k3*parm.d1*parm.d1*q(13)*q(23); % [Nm]
end
```

E.4 damper_force.m

```
function D = damper_force(t,q,parm)
% DAMPER_FORCE used for calculating the current damper force
%
% Call: damper_force(t,q,parm)
%
% Input Parameters: t, is the time
%                   q, array containg the position end the
%                       velocity of the wheels and the bogie
%                   parm, array containing the position and the
%                       velocity of both wheels and the bogie
%
% Output Parameter:  Array with current damper force

% Authors: % Ulla Uldahl
% Date:    % September 24, 2011

% Read parameters:
% d2    = parm.d2;  % [m]      Horizontal distance from spring to
    gravety
% h2    = parm.h2;  % [m]      Vertiacl distance from spring to
    gravety
```

```
% D1     = parm.D1;  % [N*s/m] Damper coefficient frame/carbody
    vertical
% D2     = parm.D2;  % [N*s/m] Damper coefficient frame/carbody
    lateral
% D3     = parm.D3;  % [N*s/m] Damper coefficient frame/carbody
    lateral

% Allocate space:
D = zeros(5,1);

% Compute damper force (lateral):
D(1) = 2*parm.D2*(parm.h2*q(24)-q(18)); % [N]
% Compute damper force (vertical):
D(2) = 2*parm.D1*q(20); % [N]
% Compute yaw damper (lateral):
D(3) = parm.D3*q(22);
% Compute damper force (roll):
D(4) = 2*parm.D1*q(24)*parm.d2^2;           % [Nm] (contribution
    from vertical damper D1)
D(5) = 2*parm.D2*(parm.h2*q(24)-q(18))*parm.h2; % [Nm] (
    contribution from lateral damper D2)
end
```

E.5 normal_force.m

```
function [Ntrack,Nz,qpen,Rwy,Rwz] = normal_force(t,q,parm,rsgeo,
    wheelset,side)

% NORMAL_FORCE used for calculating the current normal force
%
% Call: normal_force(t,q,parm,rsgeo,'rear','right')
%
% Input Parameters: t, is the time.
%                   q, array containg the position end the
%                       velocity of the wheels and the bogie.
%                   parm, array containing the position and the
%                       velocity of both wheels and the bogie.
%                   rsgeo, table containing precomputed parameters
%                   wheelset, front or rear.
%                   side, left or right.
%
% Output Parameter: Ntrack, array containg the normal force in the
%                       track coridnate system.
%                   Nz, normal force in wheel-rail contact system.
%                   qpen, dynamic penetration depth in wheel-rail
%                       contact system.
%                   Rwy, lateral position of the contact point on
%                       the wheel in the body system.
%                   Rwz, actual roling radius (vertical position
%                       of the contact point on the wheel in the
%                       body system)
```

```

% Authors: % Ulla Uldahl
% Date:    % September 22, 2011

% rsgeo(:,1)  : Lateral diplacement, {q1,q9} [m]
% rsgeo(:,2)  : Static normal force, N0 [N]
% rsgeo(:,3)  : Angle between the axle and the contact plane, delta [rad]
% rsgeo(:,4)  : Biggest semi axis, a0 [m]
% rsgeo(:,5)  : Smallest semi axis, b0 [m]
% rsgeo(:,6)  : Lateral position of the contact point on the wheel in the body system, Rwy [m]
% rsgeo(:,7)  : Actual roling radius (vertical position of the contact point on the wheel in the body system), Rwz [m]
% rsgeo(:,8)  : Kalkers creepage, C11 [-]
% rsgeo(:,9)  : Kalkers creepage, C22 [-]
% rsgeo(:,10) : Kalkers creepage, C23 [-]
% rsgeo(:,11) : Vertical position of the contact point on the rail in the track system, Rrz [m]
% rsgeo(:,12) : Static penetration, qpen0 [m]
% rsgeo(:,13) : Lateral position of the contact point on the rail in the track system, Rry [m]

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Normal forces calculation                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Choose wheelset:
if strcmp(wheelset,'front')
    y      = q(1);
    ydot   = q(2);
    z      = q(3);
    zdot   = q(4);
    psi    = q(5);
    phi    = q(7);
    phidot = q(8);
else % rear wheelset
    y      = q(9);
    ydot   = q(10);
    z      = q(11);
    zdot   = q(12);
    psi    = q(13);
    phi    = q(15);
    phidot = q(16);
end

% Choose side:
if strcmp(side,'left')
    slr =  1;
else % right side
    slr = -1;
end

% Interpolate RSGEO values:
```

```
75 rsgeo_in = [rsgeo(:,2),slr*rsgeo(:,3),slr*rsgeo(:,6),-rsgeo(:,7),
       rsgeo(:,11),rsgeo(:,12),slr*rsgeo(:,13)];
76 % rsgeo_out = interp1(rsgeo(:,1),rsgeo_in,slr*y,'linear');
77 rsgeo_out = linear_interp([rsgeo(:,1),rsgeo_in],slr*y);
78 N0    = rsgeo_out(1);
79 delta = rsgeo_out(2);
80 Rwy   = rsgeo_out(3);
81 Rwz   = rsgeo_out(4);
82 Rrz   = rsgeo_out(5);
83 qpen0 = rsgeo_out(6);
84 Rry   = rsgeo_out(7);
85
86 % Equilibrium position of the body (wheelset) in reference to the
       track system:
87 yeq = 0;
88 zeq = parm.r0;
89
90 % Compute center of mass of body in track system:
91 Rcy = yeq+y;
92 Rcz = zeq+z;
93
94 % Compute dynamic penetration depth:
95 cosdelta = cos(delta);
96 sindelta = sin(delta);
97 qpen = -(Rry-Rcy+phi*(Rrz-zeq)-Rwy)*sindelta+(Rrz-Rcz-phi*(Rry-yeq
       )-Rwz)*cosdelta;
98
99 % Compute dynamic penetration velocity:
100 vpen = (ydot-phidot*(Rrz-zeq))*sindelta-(zdot+phidot*(Rry-yeq))*
       cosdelta;
101
102 % Compute size of normal force in wheel-rail contact system (
       scalar):
103 Nz_spring = N0*(qpen/qpen0)^(3/2);
104
105 % Compute the material damping force:
106 Nz_damper = parm.Dm*vpen;
107
108 % Compute normal force with damping force:
109 Nz = Nz_spring + Nz_damper;
110
111 if ~isreal(Nz) || Nz < 0
112     Nz = 0;
113 end
114
115 % Transform normal force vector from contact to track system:
116 sinphidelta = sin(phi+delta);
117 Ntrack = [psi*sinphidelta;-sinphidelta;cos(phi+delta)]*Nz;
118 end
```

E.6 creep_force.m

```
function [Ftrack,Fx,Fy] = creep_force(t,q,parm,Nz,qpen,rsgeo,
    wheelset,side)

% CREEP_FORCE used for calculating the current normal force
%
% Call: creep_force(t,q,parm,Nz,qpen,rsgeo,'rear','right')
%
% Input Parameters: t, is the time
%                   q, array containg the position end the
%                       velocity of the wheels and the bogie
%                   parm, array containing the position and
%                       the velocity of both wheels and the bogie
%                   rsgeo, table containing precomputed parameters
%                   wheelset, front or rear.
%                   side, left or right.
%
% Output Parameter: Ftrack, array containg the creep force in
%                       the track coridnate system.
%                   Fx, the longitudinal creep forces in contact
%                       system
%                   Fy, the lateral creep forces in contact system

% Authors: % Ulla Uldahl
% Date:    % September 22, 2011

% rsgeo(:,1)  : Lateral diplacement, {q1,q9} [m]
% rsgeo(:,2)  : Static normal force, N0 [N]
% rsgeo(:,3)  : Angle between the axle and the contact plane,
    delta [rad]
% rsgeo(:,4)  : Biggest semi axis, a0 [m]
% rsgeo(:,5)  : Smallest semi axis, b0 [m]
% rsgeo(:,6)  : Lateral position of the contact point on the wheel
     in the body system, Rwy [m]
% rsgeo(:,7)  : Actual roling radius (vertical position of the
    contact point on the wheel in the body system), Rwz [m]
% rsgeo(:,8)  : Kalkers creepage, C11 [-]
% rsgeo(:,9)  : Kalkers creepage, C22 [-]
% rsgeo(:,10) : Kalkers creepage, C23 [-]
% rsgeo(:,11) : Vertical position of the contact point on the rail
     in the track system, Rrz [m]
% rsgeo(:,12) : Static penetration, qpen0 [m]
% rsgeo(:,13) : Lateral position of the contact point on the rail
    in the track system, Rry [m]

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Creep forces calculation                                                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Choose wheelset:
if strcmp(wheelset,'front')
    y       = q(1);
    ydot    = q(2);
    zdot    = q(4);
```

```
48      psi     = q(5);
49      psidot  = q(6);
50      phi     = q(7);
51      phidot  = q(8);
52      betadot = q(25); % Spin pertubation front wheel, [rad]
53  else % rear wheelset
54      y       = q(9);
55      ydot    = q(10);
56      zdot    = q(12);
57      psi     = q(13);
58      psidot  = q(14);
59      phi     = q(15);
60      phidot  = q(16);
61      betadot = q(26); % Spin pertubation rear wheel, [rad]
62  end
63
64  % Choose side:
65  if strcmp(side,'left')
66      slr =  1;
67  else % right side
68      slr = -1;
69  end
70
71  % Interpolate RSGEO values:
72  rsgeo_in = [slr*rsgeo(:,3),rsgeo(:,4),rsgeo(:,5),slr*rsgeo(:,6),-
        rsgeo(:,7),rsgeo(:,8),rsgeo(:,9),rsgeo(:,10),rsgeo(:,12)];
73  % rsgeo_out = interp1(rsgeo(:,1),rsgeo_in,slr*y,'linear');
74  rsgeo_out = linear_interp([rsgeo(:,1),rsgeo_in],slr*y);
75  delta = rsgeo_out(1);
76  a0    = rsgeo_out(2);
77  b0    = rsgeo_out(3);
78  Rwy   = rsgeo_out(4);
79  Rwz   = rsgeo_out(5);
80  C11   = rsgeo_out(6);
81  C22   = rsgeo_out(7);
82  C23   = rsgeo_out(8);
83  qpen0 = rsgeo_out(9);
84
85
86  if t > parm.t0 && parm.a0 ~= 0
87      parm.v = parm.v0 + parm.a0*(t-parm.t0);
88  end
89  cosdelta = cos(delta);
90  sindelta = sin(delta);
91
92  % Read parameters:
93  omega = parm.v/parm.r0; % Nominal spin of wheel, [1/s]
94
95  % Compute the longitudinal creepage:
96  xix = 1+(psi*ydot+Rwz*(omega+betadot)-Rwy*psidot)/parm.v;
97
98  % Compute the lateral creepage:
99  xiy = (ydot-psi*parm.v+phi*zdot-phidot*Rwz)/(parm.v*cosdelta);
100
```

```
% Compute the spin creepage:
xisp = (-sindelta*(omega+betadot)+cosdelta*psidot)/parm.v;
xi = [xix;xiy;xisp];

% Compute dynamic contact ellipse:
ab = a0*b0*(qpen/qpen0); % = a0*b0*(1+(qpen-qpen0)/qpen0);

% Define the Kalkers creepage matrix:
Kalk = [C11,    0,          0;
           0,C22,sqrt(ab)*C23];

% Compute the force along the contact plane:
Ftilde = -parm.G*ab*Kalk*xi;
normFtilde = norm(Ftilde);  % = sqrt(Ftilde(1)^2+Ftilde(2)^2)
normF = parm.mu*Nz;
u = normFtilde/normF;
if u < 3
    normF = normF*(u-1/3*u^2+1/27*u^3);
end

% Compute the adjustment factor:
epsilon = normF/normFtilde;

% Compute the longitudinal and the lateral creep forces in contact
    system:
Fx = epsilon*Ftilde(1);
Fy = epsilon*Ftilde(2);

% Transform creep force vector from contact to track system:
cosphidelta = cos(phi+delta);
Ftrack = [Fx-psi*cosphidelta*Fy;
          psi*Fx+cosphidelta*Fy;
          sin(phi+delta)*Fy];
end
```

E.7 erk.m

```
function [tout, yout, info, perf, err] = erk(fun, tspan, y0, h0,
    varstep, tol, varargin)
disp('---------- ERK ----------')

if isempty(tol)
    tol = 1e-4;
end

% varstep: 1: on, use variable step size, 2: off, fixed step size

% c|A
% ---
% 0|b
% 0|bt
```

```

% Authors: % Carsten Vølcker
% Date:    % 27.08.2008 by cv(a)imm.dtu.dk

% To simplify Input, the options AbsTol and RelTol are replaced by the
% option opts(2). And if we choose AbsTol = opts(2)^2 and RelTol = opts(2),
% then AbsTol + RelTol*norm(x, inf) = opts(2)^2 + opts(2)*norm(x, inf) =
% opts(2)*(opts(2) + norm(x, inf)).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Butcher Tableau                                                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%buttab = '23BS'
%buttab = '45F'
buttab = '45DP' %ode45
%buttab = '21HE' %explicit Euler
switch buttab
    case '21HE' % Heun-Euler (2-stage)
        B = [0 0   0  ;
             1 1   0  ;
             2 1/2 1/2;
             1 1   0  ];
    case '32BS' % Bogacki?Shampine (4-stage)
        B = [0   0    0   0   0  ;
             1/2 1/2  0   0   0  ;
             3/4 0    3/4 0   0  ;
             1   2/9  1/3 4/9 0  ;
             3   2/9  1/3 4/9 0  ;
             2   7/24 1/4 1/3 1/8];
    case '23BS' % Bogacki?Shampine (4-stage, interchanged d and dt)
        B = [0   0    0   0   0  ;
             1/2 1/2  0   0   0  ;
             3/4 0    3/4 0   0  ;
             1   2/9  1/3 4/9 0  ;
             2   7/24 1/4 1/3 1/8;
             3   2/9  1/3 4/9 0  ];
    case '45F' % Fehlberg (6-stage)
        B = [0     0          0          0          0           0       0   ;
             1/4   1/4        0          0          0           0       0   ;
             3/8   3/32       9/32       0          0           0       0   ;
             12/13 1932/2197  -7200/2197 7296/2197  0           0       0   ;
             1     439/216    -8         3680/513   -845/4104   0       0   ;
             1/2   -8/27      2          -3544/2565 1859/4104   -11/40  0   ;
```

```
58              4       25/216      0            1408/2565   2197/4104
                    -1/5    0    ;
59              5       16/135      0            6656/12825  28561/56430
                    -9/50   2/55];
60      case '54CK' % Cash-Karp (6-stage)
61          B = [0    0           0              0           0
                                0          0         ;
62               1/5  1/5         0              0           0
                                0          0         ;
63               3/10 3/40        9/40           0           0
                                0          0         ;
64               3/5  3/10        -9/10          6/5         0
                                0          0         ;
65               1    -11/54      5/2            -70/27      35/27
                                0          0         ;
66               7/8  1631/55296  175/512        575/13824   44275/110592
                    253/4096    0          ;
67               5    37/378      0              250/621     125/594
                                0          512/1771;
68               4    2825/27648  0              18575/48384 13525/55296
                    277/14336   1/4        ];
69      case '45DP' % Dormand?Prince (7-stage)
70          B = [0    0           0              0           0           0
                                0          0    ;
71               1/5  1/5         0              0           0           0
                                0          0    ;
72               3/10 3/40        9/40           0           0           0
                                0          0    ;
73               4/5  44/45       -56/15         32/9        0           0
                                0          0    ;
74               8/9  19372/6561  -25360/2187    64448/6561  -212/729    0
                                0          0    ;
75               1    9017/3168   -355/33        46732/5247  49/176
                    -5103/18656    0          0    ;
76               1    35/384      0              500/1113    125/192
                    -2187/6784     11/84      0    ;
77               4    5179/57600  0              7571/16695  393/640
                    -92097/339200  187/2100     1/40;
78               5    35/384      0              500/1113    125/192
                    -2187/6784     11/84      0    ];
79  end
80
81  A = B(1:end - 2, 2:end)';
82  b = B(end - 1, 2:end)';
83  bt = B(end, 2:end)';
84  c = B(1:end - 2, 1);
85  d = b - bt;
86  k = B(end, 1); % order of error control
87
88  % initial value...
89  t = tspan(1);
90  y = y0(:);
91  f = feval(fun, t, y, varargin{:});
92
```

```
93 % initial step-size (local truncation error of explicit Euler
       method)...
94 if isempty(h0)
95     J = numjac(fun, t, y, f, tol, varargin{:});
96     h = 0.8*(2*tol*(tol + norm(y, inf))/norm(J*f, inf))^(1/2);
97 else
98     h = h0;
99 end
100
101 % controller parameters (PI controller)...
102 kI = 0.3/k;
103 kP = 0.4/k;
104 racc = tol*(tol + norm(y, inf));
105
106 % initialize containers, counters, etc....
107 m = length(y); % no. of ODE's
108 s = length(c); % no. of stages
109 N = round((tspan(2) - tspan(1))/h); % containers in chunks N
110 tout = repmat(t, [1 N]); yout = repmat(y, [1 N]);
111 trace = (nargout > 3); if trace
112     perf = repmat([0; 0; 0], [1 N]);
113     err = repmat(zeros(length(y),1), [1 N]);
114 end
115 F = zeros(m, s); % for storing stages
116 ntot = 0; % no. of steps in total
117 nacc = 0; % no. of accepted steps
118 nrej = 0; % no. of rejected steps, saved in perf -> as (t, nrej)
119 nfun = 1; % no. of function evaluations
120
121 % integrate...
122 %w = waitbar(0,'Integrating, please wait...');
123 while t < tspan(2)
124     ntot = ntot + 1;
125     % show progress...
126     %waitbar(t/tspan(2))
127     % stage 1...
128     T = t;
129     Y = y;
130     F(:, 1) = f;
131     % stage 2-s...
132     for j = 2:s
133         nfun = nfun + 1;
134         i = j;
135         T = t + h*c(i);
136         Y = y + h*F(:, 1:j - 1)*A(1:j - 1, i);
137         F(:, i) = feval(fun, T, Y, varargin{:});
138     end
139     % estimate local truncation error...
140     e = h*F*d;
141     r = norm(e, inf);
142     % accept step if r <= tol*(tol + norm(y, inf))...
143     step_accepted = 0;
144     if r <= tol*(tol + norm(y, inf)) || ~varstep
145         step_accepted = 1;
```

```
146         nacc = nacc + 1;
147         nfun = nfun + 1;
148         % update current point...
149         t = t + h;
150         y = y + h*F*b; % using advancing method
151         %y = y + h*F*bt; % using embedded method
152         f = feval(fun, t, y, varargin{:});
153         % save output and performance...
154         tout(nacc + 1) = t;
155         yout(:, nacc + 1) = y;
156         if trace
157             perf(:, nacc + 1,:) = [h; r; nrej];
158             err(:, nacc + 1,:) = e;
159         end
160         % expand containers in chunks of N...
161         if ~mod(nacc + 1, N)
162             tout = cat(2, tout, repmat(t, [1 N]));
163             yout = cat(2, yout, repmat(y, [1 N]));
164             if trace
165                 perf = cat(2, perf, repmat(perf(:, end), [1 N]));
166                 err = cat(2, err, repmat(err(:, end), [1 N]));
167             end
168         end
169         nrej = 0;
170     end
171     %%%%%%%%% variable step size BEGIN %%%%%%%%%
172     if varstep
173         % update stepsize (assymptotic controller)...
174         %h = h*max(0.1, min((0.8*tol*(tol + norm(y, inf))/r)^(1/k)
                , 10));
175         % update stepsize (PI controller)...
176         if step_accepted
177             h = h*max(0.1, min((0.8*tol*(tol + norm(y, inf))/r)^kI
                    *(racc/r)^kP, 10));
178             racc = r;
179         else
180             nrej = nrej + 1;
181             h = h*max(0.1, min((0.8*tol*(tol + norm(y, inf))/r)
                    ^(1/k), 10));
182         end
183     end
184     %%%%%%%%% variable step size END %%%%%%%%%
185     % ensure t + h <= tend...
186     h = min(h, tspan(2) - t);
187 end
188 %close(w)
189 % build information, solution and performace...
190 info = [ntot ntot-nacc nfun];
191 tout = tout(1:nacc + 1)';
192 yout = yout(:, 1:nacc + 1)';
193 if  trace, perf = perf(:, 1:nacc + 1); err = err(:, 1:nacc + 1);
    end
194
195 function J = numjac(fun, t, y, f, tol, varargin)
```

```
% Approximate Jacobian by forward difference, needs n + 1 function
% evaluations (f(t, y) though is allready given as input).
n = length(y);
J = zeros(n);
yp = y;
% relative perturbation...
%p = tol*(tol + y); % hvorfor ikke goere saadan ???
p = sqrt(eps)*max(y, 1);
% approximate Jacobian...
for i = 1:n
    yp(i) = y(i) + p(i); % perturbate direction i
    %p(i) = yp(i) - y(i); % hvorfor goere det her ???
    fp = feval(fun, t, yp, varargin{:});
    J(:,i) = (fp - f)/p(i);
    yp(i) = y(i); % reset direction i
end
```

E.8 linear_interp.m

```
function [y,dydx] = linear_interp(S,x)

% Call:
%   [y,dydx] = linear_interp(S,x)
% Input:
%   S : Set of m equaly distributed design sites S(:,1) with
    responses
%       S(:,2:n), matrix of size mxn.
%   x : Sample point.
% Output:
%   y    : Linear interpolation of responses at sample point x.
%   dydx : Gradients of responses (1. order approx.) at sample
    point x.

% Authors: % Carsten Vølcker

% Distance between design sites:
dx = S(2,1) - S(1,1);

% Index of nearest design site S(i,1) < x:
i = floor(1 + (x - S(1,1))/dx);
if i < size(S,1)
    %disp('i < size(S,1)')
    if x < S(1,1)
        error(['x must be larger than ',num2str(S(1,1))])
    end
    % Gradient of responses between design sites S(i,1) and S(i
        +1,1):
    %dydx = (S(i+1,2:end) - S(i,2:end))/dx; % dydx = (y2 - y1)/(x2
        - x1)
    dydx = (S(i+1,2:end) - S(i,2:end))/(S(i+1,1) - S(i,1)); % dydx
        = (y2 - y1)/(x2 - x1)
```

```
28      % Linear interpolated responses at sample point x:
29      y = S(i,2:end) + (x - S(i,1))*dydx;
30  else
31      %disp('i >= size(S,1)')
32      if x > S(end,1)
33          error(['x must be smaller than ',num2str(S(end,1))])
34      end
35      % Gradient of responses between design sites S(end-1,1) and S(
            end,1):
36      %dydx = (S(end,2:end) - S(end-1,2:end))/dx; % dydx = (y2 - y1)
            /(x2 - x1)
37      dydx = (S(end,2:end) - S(end-1,2:end))/(S(end,1) - S(end-1,1))
            ; % dydx = (y2 - y1)/(x2 - x1)
38      % Linear interpolated responses at sample point x:
39      y = S(end,2:end);
40  end
```

Bibliography

[1] Kaas-Petersen, Chr.: *Chaos in a Railway Bogie*, Acta Mechanica 61, (p. 89-107) Springer-Verlag 1986.

[2] Engbo Christensen, Lasse.: *The Dynamics of a Railway Vehicle on a Disturbed Track*, Master project, IMM - DTU, DK, June 2001

[3] *Personal communication with professor Hans True*, IMM - DTU, DK, 2012

[4] *Personal communication with professor Allan Peter Engsig-Karup*, IMM - DTU, DK, 2012

[5] Christian Jensen, Jens.: *Teoretiske og eksperimentelle dynamiske undersøgelser af jernbanekøretøjer*, Ph.D., IMM - DTU, DK, 1995

[6] Hoffmann, Mark.: *Dynamics of European two-axle freight wagons*, Ph.D., IMM - DTU, DK, September 2006

[7] Anderson, Berg and Stichel.: *Rail Vehicle Dynamics*, Railway group KTH, 2005

[8] Bigoni, Daniele.: *Curving Dynamics in High Speed Trains PUBLIC VERSION*, Master project, IMM - DTU, DK, 2011

[9] Garg, V. K. and Dukkipati, R. V.: *Dynamics of railway vehicle systems*, Acedemic Press, 1984

[10] Völcker, Carsten.: *Dynamics and Stability of a Railway Vehicle with Realistic Nonlinear Dampers*, Midterm Project, IMM - DTU, DK, June 2004.