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

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 The bogie model</td>
<td>1</td>
</tr>
<tr>
<td>2 Mathematical model</td>
<td>11</td>
</tr>
<tr>
<td>3 Numerical implementation</td>
<td>23</td>
</tr>
<tr>
<td>4 Finding the critical velocities</td>
<td>37</td>
</tr>
</tbody>
</table>

### 1 The bogie model

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Views of the model</td>
<td>2</td>
</tr>
<tr>
<td>1.2 The constants of the model</td>
<td>3</td>
</tr>
<tr>
<td>1.3 System of equations</td>
<td>5</td>
</tr>
<tr>
<td>1.4 Differential equations</td>
<td>9</td>
</tr>
</tbody>
</table>

### 2 Mathematical model

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Coordinate system</td>
<td>11</td>
</tr>
<tr>
<td>2.2 Rotation matrices</td>
<td>12</td>
</tr>
<tr>
<td>2.3 Wheel/rail interaction</td>
<td>13</td>
</tr>
<tr>
<td>2.4 RSGEO - contact table</td>
<td>18</td>
</tr>
</tbody>
</table>

### 3 Numerical implementation

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Known numerical issues of the Cooperrider model</td>
<td>23</td>
</tr>
<tr>
<td>3.2 Verification strategy</td>
<td>24</td>
</tr>
<tr>
<td>3.3 Simplifying the code</td>
<td>24</td>
</tr>
<tr>
<td>3.4 Test and evaluation of different numerical time integration methods</td>
<td>25</td>
</tr>
<tr>
<td>3.5 Verification of the implementation of the bogie model</td>
<td>28</td>
</tr>
<tr>
<td>3.6 Verification of the implementation of the normal and the creep forces</td>
<td>31</td>
</tr>
<tr>
<td>3.7 Testing the system at different velocities</td>
<td>34</td>
</tr>
<tr>
<td>3.8 Comparing two different models</td>
<td>35</td>
</tr>
</tbody>
</table>

### 4 Finding the critical velocities

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1 The bifurcation diagram</td>
<td>37</td>
</tr>
<tr>
<td>4.2 Critical Velocity</td>
<td>40</td>
</tr>
<tr>
<td>4.3 Supercritical Hopf bifurcation</td>
<td>42</td>
</tr>
<tr>
<td>4.4 Finding the subcritical symmetry breaking bifurcation</td>
<td>44</td>
</tr>
</tbody>
</table>
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 comparison to those of the primary suspension the springs are relatively 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, \( F_{\text{spring}} = -ky \). 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.](image)
1.2. THE CONSTANTS OF THE MODEL

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

In the following table the constants of the model are listed. 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

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>0.75 m</td>
<td>Half the track gauge.</td>
</tr>
<tr>
<td>$b$</td>
<td>1.074 m</td>
<td>Half the distance between the two wheel axles.</td>
</tr>
<tr>
<td>$r_0$</td>
<td>0.425 m</td>
<td>The nominal rolling radius of the wheel.</td>
</tr>
<tr>
<td>$d_1$</td>
<td>0.620 m</td>
<td>Horizontal distance from the springs $k_2$ and $k_3$ to a center of gravity.</td>
</tr>
<tr>
<td>$d_2$</td>
<td>0.680 m</td>
<td>Horizontal distance from the springs $k_5$ and the damper $D_1$ to a center of gravity.</td>
</tr>
<tr>
<td>$h_1$</td>
<td>0.0762 m</td>
<td>Vertical distance from the springs $k_1$ to a center of gravity.</td>
</tr>
<tr>
<td>$h_2$</td>
<td>0.6584 m</td>
<td>Vertical distance from the springs $k_4$ and the damper $D_2$ to a center of gravity.</td>
</tr>
</tbody>
</table>

## The masses and moments of inertia (M.o.i.) used in this work

<table>
<thead>
<tr>
<th>Mass or M.o.i.</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m_w$</td>
<td>1022 kg</td>
<td>Mass of the wheel axle.</td>
</tr>
<tr>
<td>$I_{wx}$</td>
<td>678 kg·m²</td>
<td>M.o.i. for the roll motions of the wheel around longitudinal axis.</td>
</tr>
<tr>
<td>$I_{wy}$</td>
<td>80 kg·m²</td>
<td>M.o.i. for the pitch motions of the wheel around lateral axis.</td>
</tr>
<tr>
<td>$I_{wz}$</td>
<td>678 kg·m²</td>
<td>M.o.i. for the yaw motions of the wheel around vertical axis.</td>
</tr>
<tr>
<td>$m_f$</td>
<td>2918.9 kg</td>
<td>Mass of the frame.</td>
</tr>
<tr>
<td>$I_{fx}$</td>
<td>6780 kg·m²</td>
<td>M.o.i. for the roll motions of the frame around longitudinal axis.</td>
</tr>
<tr>
<td>$I_{fz}$</td>
<td>6780 kg·m²</td>
<td>M.o.i. for the yaw motions of the frame around vertical axis.</td>
</tr>
<tr>
<td>$m_c$</td>
<td>44388 kg</td>
<td>Mass of the railcar.</td>
</tr>
</tbody>
</table>

### Primary suspensions

<table>
<thead>
<tr>
<th>Suspension</th>
<th>Stiffness</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_1$</td>
<td>1823 kN/m</td>
<td>Lateral horizontal spring, wheel-frame.</td>
</tr>
<tr>
<td>$k_2$</td>
<td>3646 kN/m</td>
<td>Longitudinal horizontal spring, wheel-frame.</td>
</tr>
<tr>
<td>$k_3$</td>
<td>3646 kN/m</td>
<td>Vertical spring, wheel-frame.</td>
</tr>
</tbody>
</table>

### Secondary suspensions

<table>
<thead>
<tr>
<th>Suspension</th>
<th>Stiffness</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_4$</td>
<td>182.3 kN/m</td>
<td>Lateral horizontal spring, frame-carbody.</td>
</tr>
<tr>
<td>$k_5$</td>
<td>333.3 kN/m</td>
<td>Vertical spring, frame-carbody.</td>
</tr>
<tr>
<td>$k_6$</td>
<td>2710 kN/m</td>
<td>kN/m Torsion spring, frame-carbody.</td>
</tr>
<tr>
<td>$D_1$</td>
<td>20 kN/m</td>
<td>Vertical damper, frame-carbody.</td>
</tr>
<tr>
<td>$D_2$</td>
<td>29.2 kN/m</td>
<td>Lateral horizontal damper, frame-carbody.</td>
</tr>
<tr>
<td>$D_3$</td>
<td>500 kN/m</td>
<td>Lateral horizontal damper, frame-carbody.</td>
</tr>
<tr>
<td>$D_m$</td>
<td>150 kN·s/m</td>
<td>Material damper, contact area. [6]</td>
</tr>
</tbody>
</table>

Table 1.1: Constants of the model. Unless otherwise mentioned, all parameters 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 comparison to the bogie frame. Therefore the position of the car body can be assumed 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 spin perturbation 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 wheel axles and a spin perturbation only for the wheel axles. 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:**

- **Rear axle:**

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 horizontal 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 examined 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 law $F_{spring} = -ky$, 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 approximation is done using:

\[
\sin(\delta) \approx \delta, \quad \cos(\delta) \approx 1 \quad (1.1)
\]

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

\[
q_1 - q_9 - b \sin(q_{11}) - h_1 \sin(q_{12}) \approx q_1 - q_9 - bq_{11} - h_1 q_{12} \quad (1.2)
\]

Note the yaw contribution \( b q_{11} \), 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:

\[
q_5 - q_9 + b \sin(q_{11}) - h_1 \sin(q_{12}) \approx q_5 - q_9 + bq_{11} - h_1 q_{12} \quad (1.3)
\]

and for the bogie itself:

\[
h_2 \sin(q_{12}) - q_9 \approx h_2 q_{12} - q_9 \quad (1.4)
\]

and the damping of the bogie is given by:

\[
2D_2(h_2 \dot{q}_{12} - \dot{q}_9) \quad (1.5)
\]

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:

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

The definition of force moment is \( M_b = r \times F \). By use of the rotation matrices \( A_{bT} \) derived in section 2.2 we project the contact force into the body system.

\[
M_b = R_w \times A_{bT}(F + N) \quad (1.7)
\]

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.
1.3. SYSTEM OF EQUATIONS

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 2.2.3. The latter two equations calculates the difference between the actual angular velocity of the wheelsets and the theoretical value.

\[ m_w q_1 = -A_1 + F_{f_{ty}} + F_{f_{ry}} + N_{f_{ty}} + N_{f_{ry}} \]  \hspace{1cm} (1.8a)  \hspace{1cm} \\
\[ m_w q_2 = -A_2 + F_{f_{tz}} + F_{f_{rz}} + N_{f_{tz}} + N_{f_{rz}} - F_{wfc} \]  \hspace{1cm} (1.8b)  \hspace{1cm} \\
\[ I_{wx} q_3 = -A_3 - a_{fr}(F_{frx} + (F_{fry} + N_{f_{ry}})q_3) \]  \hspace{1cm} (1.8c)  \hspace{1cm} \\
\[ I_{wx} q_4 = -A_4 + a_{fr}(F_{f_{tz}} + N_{f_{tz}} - (F_{f_{ty}} + N_{f_{ty}})q_4) \]  \hspace{1cm} (1.8d)  \hspace{1cm} \\
\[ m_w q_5 = -A_5 + F_{r_{ly}} + F_{r_{ry}} + N_{r_{ly}} + N_{r_{ry}} \]  \hspace{1cm} (1.8e)  \hspace{1cm} \\
\[ m_w q_6 = -A_6 + F_{r_{lz}} + F_{r_{rz}} + N_{r_{lz}} + N_{r_{rz}} - F_{wfc} \]  \hspace{1cm} (1.8f)  \hspace{1cm} \\
\[ I_{wx} q_7 = -A_7 - a_{rr}(F_{rrx} + (F_{rry} + N_{rry})q_7) \]  \hspace{1cm} (1.8g)  \hspace{1cm} \\
\[ I_{wx} q_8 = -A_8 + a_{rl}(F_{rlx} + N_{rlx} - (F_{rl_{ly}} + N_{rl_{ly}})q_8) \]  \hspace{1cm} (1.8h)  \hspace{1cm} \\
\[ m_f q_9 = A_1 + A_5 + A_9 + 2D_2(h_2q_{12} - \dot{q}_9) \]  \hspace{1cm} (1.8i)  \hspace{1cm} \\
\[ m_f q_{10} = A_2 + A_6 - A_{10} - 2D_1 \dot{q}_{10} \]  \hspace{1cm} (1.8j)  \hspace{1cm} \\
\[ I_{fx} \dot{q}_{11} = bA_1 + A_3 - bA_5 + A_7 - A_{11} - D_3 \dot{q}_{11} \]  \hspace{1cm} (1.8k)  \hspace{1cm} \\
\[ I_{fx} \dot{q}_{12} = h_1 A_1 + A_4 + h_1 A_5 + A_8 - h_2 A_9 - A_{12} - 2D_1 \dot{q}_{12} \]  \hspace{1cm} (1.8l)  \hspace{1cm} \\
\[ \dot{\beta}_{fy} q_{13} = -A_{13} + r_{fr}(F_{frx} + (F_{fry} + N_{f_{ry}})q_3) \]  \hspace{1cm} (1.8m)  \hspace{1cm} \\
\[ \dot{\beta}_{fy} q_{14} = -A_{14} + r_{rr}(F_{frx} + (F_{fry} + N_{rry})q_7) \]  \hspace{1cm} (1.8n)  \hspace{1cm} \\
\[ + r_{rl}(F_{rlx} + (F_{rl_{ly}} + N_{rl_{ly}})q_7) \]
CHAPTER 1. THE BOGIE MODEL

Where all the spring forces are:

\[ A_1 = 2k_1(q_1 - q_0 - bq_{11} - h_1 q_{12}) \]  (1.9a)
\[ A_2 = 2k_3(q_2 - q_{10}) \]  (1.9b)
\[ A_3 = 2k_2d_1^2(q_3 - q_{11}) \]  (1.9c)
\[ A_4 = 2k_3d_1^2(q_4 - q_{12}) \]  (1.9d)
\[ A_5 = 2k_1(q_5 - q_0 + bq_{11} - h_1 q_{12}) \]  (1.9e)
\[ A_6 = 2k_3(q_6 - q_{10}) \]  (1.9f)
\[ A_7 = 2k_2d_1^2(q_7 - q_{11}) \]  (1.9g)
\[ A_8 = 2k_3d_1^2(q_8 - q_{12}) \]  (1.9h)
\[ A_9 = 2k_4(h_2 q_{12} - q_0) \]  (1.9i)
\[ A_{10} = 2k_5 q_{10} \]  (1.9j)
\[ A_{11} = k_6 q_{11} \]  (1.9k)
\[ A_{12} = 2k_5d_2^2 q_{12} \]  (1.9l)
\[ A_{13} = 2k_3d_1^2 q_{3} q_{12} \]  (1.9m)
\[ A_{14} = 2k_3d_1^2 q_{7} q_{12} \]  (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:

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

And the rewritten form:

\[
\begin{align*}
\frac{dx_1}{dt} &= \frac{dq_1}{dt} = \dot{q}_1 = x_2 \\ \frac{dx_2}{dt} &= \frac{dq_1}{dt} = \ddot{q}_1 = \sum F_{\text{frontwheel}} m_w \\
&\vdots \\
\frac{dx_{23}}{dt} &= \frac{dq_{12}}{dt} = \dot{q}_{12} = x_{24} \\
\frac{dx_{24}}{dt} &= \frac{dq_{12}}{dt} = \ddot{q}_{12}
\end{align*}
\]
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 normal forces.

<table>
<thead>
<tr>
<th>System</th>
<th>Base</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_T$ : ${O_T ; x_T, y_T, z_T }$</td>
<td>$i_T, j_T, k_T$</td>
<td>Track system</td>
</tr>
<tr>
<td>$R_b$ : ${O_b ; x_b, y_b, z_b }$</td>
<td>$i_b, j_b, k_b$</td>
<td>2 body systems</td>
</tr>
<tr>
<td>$R_c$ : ${O_c ; x_c, y_c, z_c }$</td>
<td>$i_c, j_c, k_c$</td>
<td>4 wheel-rail contact systems</td>
</tr>
</tbody>
</table>

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 direction 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.
CHAPTER 2. MATHEMATICAL MODEL

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, \( A^{-1} = A^T \). At the same time we see that when \( \alpha = 0 \) we get the identity matrix for all the rotation matrices.

\[
\begin{align*}
A_z^{(\alpha)} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix}, \\
A_x^{(\alpha)} &= \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}
\end{align*}
\]

Body system to track system - track to body

1. Rotation around \( z \) by \( \psi \) (yaw)
2. Rotation around \( x \) by \( \phi \) (roll)

\[
A_{Tb} = A_z^{(\psi)}A_x^{(\phi)}, \quad A_{bT} = A_{Tb}^T
\]

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

1. Rotation around \( x_b \) by \( \delta \) - the contact angle is given from RSGEO-table.

\[
A_{bc} = A_x^{(\delta)}, \quad A_{cb} = A_{bc}^T
\]
2.3.

**WHEEL/RAIL INTERACTION**

**Equations of motion**

The velocity \( v \) of the center of mass is expressed with reference to the track base.

\[
\begin{bmatrix}
V \\
y \\
\dot{y}
\end{bmatrix} = \begin{bmatrix}
V \\
0 \\
0
\end{bmatrix} + \begin{bmatrix}
0 \\
\psi \\
\phi
\end{bmatrix} = \begin{bmatrix}
\dot{V} \\
\dot{y} \\
\dot{z}
\end{bmatrix}
\]

(2.1)

and the angular velocity \( \Omega \) of the wheel-axle is expressed in reference to the body system:

\[
\Omega_b = \begin{bmatrix}
1 & \psi & 0 \\
-\psi & 1 & \phi \\
0 & -\phi & 1
\end{bmatrix} \begin{bmatrix}
\dot{\psi} \\
0 \\
\dot{\phi}
\end{bmatrix} = \begin{bmatrix}
\dot{V} \\
\dot{y} \\
\dot{z}
\end{bmatrix} \begin{bmatrix}
\frac{V}{r_0} - \dot{\beta} \\
\psi
\end{bmatrix}
\]

(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 \( E = 2.1 \cdot 10^{11} \text{N/m}^2 \)
- Poissions ratio \( \nu = 0.27 \)
- Shear modulus \( G = \frac{E}{2(1+\nu)} = 8.2677 \cdot 10^{10} \text{N/m}^2 \)
- Friction coefficient \( \mu = 0.3 \)

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. \( R_C \) is a vector defining the position of the center of the mass of the wheelset and \( R_R \) defines the position of the contact point on the rail. Both vectors are with reference to the base of the track system. \( R_{w} \) 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:

\[ \mathbf{R}_{pen} = A_{cb}\left(A_{\delta}T(\mathbf{R}_R - \mathbf{R}_C) - \mathbf{R}_w\right) \]

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

\[ q_{pen} \approx \sin(\delta)(-R_{Ry} + R_{Cy} - \phi(R_{Rz} - R_{Cz}) + R_{wy}) \]
\[ + \cos(\delta)(R_{Rz} - R_{Cz} - \phi(R_{Ry} - R_{Cy}) - R_{wz}) \] (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 one third. [5] page 35. From the theory one get

\[ a \propto N^{\frac{1}{3}} , \quad b \propto N^{\frac{1}{3}} \] (2.4)
The penetration is required in order to calculate the normal force and the relation is given by:

\[ N \propto q_{\text{pen}}^3 \quad (2.5) \]

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

\[ N_z = N_{\text{spring}} + N_{\text{damper}} \quad (2.6) \]

where

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

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

\[ v_{\text{pen}} = (\dot{y} - \dot{\phi}(R_{Rz} - R_{Cz})) \sin(\delta) - (\dot{z} + \dot{\phi}(R_{Ry} - R_{Cy})) \cos(\delta) \quad (2.7) \]

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

\[ \frac{a}{b} = \frac{a_0}{b_0} \quad (2.8) \]

\[ ab = a_0b_0 \left( \frac{N}{N_0} \right)^{\frac{3}{2}} = a_0b_0 \left( \frac{N_0 \left( \frac{q_{\text{pen}}}{q_{\text{pen0}}} \right)^{\frac{3}{2}}}{N_0} \right)^{\frac{3}{2}} = a_0b_0 \frac{q_{\text{pen}}}{q_{\text{pen0}}} \quad (2.9) \]

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

\[
v_{\text{con}} = A_{cb}(A_{bT}v + \Omega_b \times R_w) \\
\approx A_{cb} \left[ \begin{array}{ccc}
1 & \psi & 0 \\
-\psi & 1 & \phi \\
0 & -\phi & 1
\end{array} \right] \left[ \begin{array}{c}
V \\
\dot{y} \\
\dot{z}
\end{array} \right] + \left[ \begin{array}{c}
\left( \frac{V}{r_0} - \dot{\beta} \right) R_{wz} - \dot{\psi} R_{wy} \\
-\dot{\phi} R_{wz} \\
\dot{\phi} R_{wy}
\end{array} \right]
\]

\[
= A_{cb} \left[ V + \psi \dot{y} + \left( \frac{V}{r_0} - \dot{\beta} \right) R_{wz} - \dot{\psi} R_{wy} \\
\dot{y} - \psi V + \phi \dot{z} - \dot{\phi} R_{wz} \\
-\dot{\phi} \dot{y} + \dot{z} + \dot{\phi} R_{wy}
\right]
\]

\[
= \left[ V + \psi \dot{y} + \left( \frac{V}{r_0} - \dot{\beta} \right) R_{wz} - \dot{\psi} R_{wy} \\
\left( \dot{y} - \psi V + \phi \dot{z} - \dot{\phi} R_{wz} \right) \cos(\delta) + \left( \phi \dot{y} V - \dot{\phi} \dot{y} + \dot{z} + \dot{\phi} R_{wy} \right) \sin(\delta) \\
-\sin(\delta) \left( -\psi V + \dot{y} + \phi \dot{z} - \dot{\phi} R_{wz} \right) + \cos(\delta) \left( \psi \dot{y} V - \dot{\psi} \dot{y} + \dot{z} + \dot{\phi} R_{wy} \right)
\]
\]

(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. \((v_{\text{con}}|_z = 0)\). Then we get the following.

\[
v_{\text{con}} \approx \left[ V + \psi \dot{y} + \left( \frac{V}{r_0} - \dot{\beta} \right) R_{wz} - \dot{\psi} R_{wy} \\
\left( \dot{y} - \psi V + \phi \dot{z} - \dot{\phi} R_{wz} \right) / \cos(\delta) \\
0
\right]
\]

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

\[
A_{cb} \Omega_b = \left[ \begin{array}{c}
\phi \\
\cos(\delta) \left( \frac{V}{r_0} - \dot{\beta} \right) + \sin(\delta) \psi \\
-\sin(\delta) \left( \frac{V}{r_0} - \dot{\beta} \right) + \cos(\delta) \psi
\end{array} \right]
\]
2.3. WHEEL/RAIL INTERACTION

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.

\[
\begin{align*}
\xi_x &= \frac{\nu_{\text{con},x}}{V} = 1 + \frac{\psi V - \dot{\delta} R w z - \dot{\psi} R w y}{V} \\
\xi_y &= \frac{\nu_{\text{con},y}}{V} = \frac{\dot{\gamma} V + \dot{\delta} R w z - \dot{\gamma} V}{V} \\
\xi_{sp} &= \frac{(A_{ab} \Omega_{b})_z}{V} = -\sin(\delta) \frac{\nu_{\text{con},z}}{V} + \cos(\delta) \dot{\psi}
\end{align*}
\]

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:

\[
\mathbf{F} = \begin{bmatrix} F_x \\ F_y \\ M_z \end{bmatrix}
\]

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.

\[
\begin{bmatrix} \tilde{F}_x \\ \tilde{F}_y \end{bmatrix} = -G a b \begin{bmatrix} C_{11} & 0 & 0 \\ 0 & C_{22} & \sqrt{a b} C_{23} \end{bmatrix} \begin{bmatrix} \xi_x \\ \xi_y \\ \xi_{sp} \end{bmatrix}
\]

where the coefficients \(C_{ij}\) 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:

\[
|\tilde{\mathbf{F}}| = \sqrt{\tilde{F}_x^2 + \tilde{F}_y^2} > \mu N
\]

Kalker’s theory does not take into account that the creep force can not exceed Coulomb’s law, \(F = \mu N\), where \(\mu\) 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

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

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

\[
\epsilon = \frac{|\mathbf{F}|}{|\tilde{\mathbf{F}}|}
\]

And finally, the longitudinal creep force \(F_x\) and the lateral creep force \(F_y\) are given by

\[
F_x = \epsilon \tilde{F}_x, \quad F_y = \epsilon \tilde{F}_y
\]

These forces are given in the contact coordinate system.
CHAPTER 2. MATHEMATICAL MODEL

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.

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

Table 2.2: Parameters in the RSGEO table file.

In figure 2.4 the static normalforce is shown. When we for example 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

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 Runge-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

$$\text{error} = |q_{9}^{\text{ref}} - q_{9}|$$

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 $q_{9}^{\text{ref}}$ 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 $v = 135 \text{ m/s}$, 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 $q_{9}^{\text{ref}}$ 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 that Dormand-Prince is a 4th order method, meaning that the accumulated global error per step is off the order $O(h^4)$. However, this only applies to smooth functions that are differentiable any number of times. Our model is not

![Figure 3.1: Convergence test with different tolerances using erk.m.](image)
differentiable any number of times, because of the discontinuous parameters in RSGEO, see section 2.4.
CHAPTER 3. NUMERICAL IMPLEMENTATION

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.](image)

![Figure 3.3: No vertical disturbance of the front wheels and fixed rear wheels and bogie frame.](image)
3.5. VERIFICATION OF THE IMPLEMENTATION OF THE BOGIE MODEL

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.](image)

![Figure 3.7: No vertical disturbance of the front wheels and with fixed rear wheels and bogie frame and without normal and creep forces.](image)

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 lose 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.
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 $q_{pen}$ 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.

![Normal force Front wheel, left](image1)

![Normal force Front wheel, right](image2)

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.

![Normal force Front wheel, left](image3)

![Normal force Front wheel, right](image4)

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
3.6. VERIFICATION OF THE IMPLEMENTATION OF THE NORMAL AND THE CREEP FORCES

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 start 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 128 m/s. Figure 3.15: Lateral displacement of the rear wheels. The bogie travels at a velocity of 132 m/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.
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 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 symmetry breaking bifurcation occurs.

The three velocities are related such that

$$v_C < v_H < v_S$$  \hspace{1cm} (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).

From the description above, we see that different velocities generates different solutions, which we will list below:
4.1. **THE BIFURCATION DIAGRAM**

- **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 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

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

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
4.2. CRITICAL VELOCITY

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

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

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).](image)

that the hunting motions start to decrease at \( t = 11.5 \text{ s} \), 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 \( t = 5 \text{ s} \) 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 \( 59.5 < v_C < 59.7 \text{ m/s} \).
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
4.3. **SUPERCRITICAL HOPF BIFURCATION**

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. \( \text{Re } \lambda < 0 \), 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. \( \text{Re } \lambda > 0 \), 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 super-critical Hopf bifurcation. From section 4.2 we know that \( v_C = 59 \text{ m/s} \), 
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 \( 59 < v_H < 128 \text{ m/s} \), 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.](image)

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 
\( v = 125.5 \text{ m/s} \) the system is in a stable stationary solution (the green line). 
In the right figure we see that the amplitude is increasing, so at \( v = 126 \text{ m/s} \) 
the system is in an asymptotically stable periodic symmetric solution (the red 
line). Therefore, we conclude that \( 125.5 < v_H < 126.0 \text{ m/s} \).
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).
4.4. FINDING THE SUBCRITICAL SYMMETRY BREAKING BIFURCATION

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

\( A_{Tb} \) Rotation matrix changing body coordinates to track coordinates
\( A_{bT} \) Rotation matrix changing track coordinates to body coordinates
\( A_{bc} \) Rotation matrix changing contact coordinates to body coordinates
\( A_{cb} \) Rotation matrix changing body coordinates to contact coordinates
\( A_{(\alpha)} \) Rotation matrix. Around axis x with the angle \( \alpha \)
\( A_{z(\alpha)} \) Rotation matrix. Around axis z with the angle \( \alpha \)
\( a \) 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
\( |F| \) Magnitude of the creep force according to Kalker’s linear theory
\( |F| \) 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)
APPENDIX A. LIST OF SYMBOLS

\( i_T, j_T, k_T \) Base of the track coordinate system
\( i_b, j_b, k_b \) Base of the body coordinate system
\( i_c, j_c, k_c \) 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
\( R_{pen} \) Vector from the contact point on the wheel to the contact point on the rail
\( R_c \) Vector defining the position of the center of the mass of the wheelset
\( R_R \) Vector defining the contact point on the rail
\( R_\omega \) Vector defining the contact point on the wheel
\( r_0 \) Nominal rolling radius of the wheel
\( v \) Velocity vector
\( v_{con} \) Relative velocity between the wheel and the rail

Table A.2: Symbols (i-v)

\( \dot{\beta} \) Spin perturbation
\( \delta \) Contact angle
\( \epsilon \) Adjustment factor in Shen-Hedrick-Elkins’ model
\( \mu \) Friction coefficient
\( \nu \) Poisson’s ratio
\( \xi_x \) Longitudinal creep
\( \xi_y \) Lateral creep
\( \xi_z \) Spin creep
\( \phi \) Roll angle
\( \psi \) Yaw angle
\( \omega \) The nominal spin
\( \Omega_b \) 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 \( \alpha \). The bases are related by:

\[
x_1 = (x_1 \cdot x_2)x_2 + (x_1 \cdot y_2)y_2 + (x_1 \cdot z_2)z_2 = x_2 \\
y_1 = (y_1 \cdot x_2)x_2 + (y_1 \cdot y_2)y_2 + (y_1 \cdot z_2)z_2 = \cos \alpha y_2 - \sin \alpha z_2 \\
z_1 = (z_1 \cdot x_2)x_2 + (z_1 \cdot y_2)y_2 + (z_1 \cdot z_2)z_2 = \sin \alpha y_2 + \cos \alpha z_2
\]

\[
A_x^{(\alpha)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha \\ 0 & \sin \alpha & \cos \alpha \end{bmatrix}
\]

For the right wheels the two bases are related by:

\[
x_1 = (x_1 \cdot x_2)x_2 + (x_1 \cdot y_2)y_2 + (x_1 \cdot z_2)z_2 = x_2 \\
y_1 = (y_1 \cdot x_2)x_2 + (y_1 \cdot y_2)y_2 + (y_1 \cdot z_2)z_2 = \cos \alpha y_2 + \sin \alpha z_2 \\
z_1 = (z_1 \cdot x_2)x_2 + (z_1 \cdot y_2)y_2 + (z_1 \cdot z_2)z_2 = -\sin \alpha y_2 + \cos \alpha z_2
\]

\[
A_x^{(\alpha)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & \sin \alpha \\ 0 & -\sin \alpha & \cos \alpha \end{bmatrix}
\]
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 \( \alpha \).

\begin{align*}
x_1 &= (x_1 \cdot x_2)x_2 + (x_1 \cdot y_2)y_2 + (x_1 \cdot z_2)z_2 = \cos \alpha x_2 - \sin \alpha y_2 \\
y_1 &= (y_1 \cdot x_2)x_2 + (y_1 \cdot y_2)y_2 + (y_1 \cdot z_2)z_2 = \sin \alpha x_2 + \cos \alpha y_2 \\
z_1 &= (z_1 \cdot x_2)x_2 + (z_1 \cdot y_2)y_2 + (z_1 \cdot z_2)z_2 = z_2
\end{align*}

\[ A_2^{(\alpha)} = \begin{bmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \]

Wheel-rail contact system to body system

1. Rotation around \( x_b \) by \( \delta \)

\[ A_{bc} = A_x^{(\delta)} \quad , \quad A_{cb} = A_{bc}^T \]

Body system to track system

1. Rotation around \( z_T \) by \( \psi \) (yaw)

2. Rotation around \( x_T \) by \( \phi \) (roll)

\[ A_{Tb} = A_x^{(\psi)}A_y^{(\phi)} \quad , \quad A_{bT} = A_{Tb}^T \]
Rotationmatrices

The following matrices is simplified using trigonometry.

Right/Left wheel
\[ A_{Tb} = A_z(\psi) A_x(\phi) \]
\[ = \begin{bmatrix} c\psi & -s\psi c\phi & s\psi s\phi \\ s\psi & c\psi c\phi & -c\psi s\phi \\ 0 & s\phi & c\phi \end{bmatrix} \]

Left wheel
\[ A_{bc} = A(\delta) \]
\[ = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\delta & -s\delta \\ 0 & s\delta & c\delta \end{bmatrix} \]
\[ A_{Tc} = A_{Tb} A_{bc} \]
\[ = \begin{bmatrix} c\psi & -s\psi c(\phi + \delta) & s\psi s(\phi + \delta) \\ s\psi & c\psi c(\phi + \delta) & -c\psi s(\phi + \delta) \\ 0 & s(\phi + \delta) & c(\phi + \delta) \end{bmatrix} \]

Right wheel
\[ A_{bc} = A(\delta) \]
\[ = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\delta & s\delta \\ 0 & -s\delta & c\delta \end{bmatrix} \]
\[ A_{Tc} = A_{Tb} A_{bc} \]
\[ = \begin{bmatrix} c\psi & -s\psi c(\phi - \delta) & s\psi s(\phi - \delta) \\ s\psi & c\psi c(\phi - \delta) & -c\psi s(\phi - \delta) \\ 0 & s(\phi - \delta) & c(\phi - \delta) \end{bmatrix} \]

Table B.1: Rotationmatrices. Here, \( c = \cos \) and \( s = \sin \)
Right/Left wheel

\[
\mathbf{A}_{T_b} = \mathbf{A}_x^{(\psi)} \mathbf{A}_x^{(\phi)} = \begin{bmatrix} 1 & -\psi & 0 \\ \psi & 1 & -\phi \\ 0 & \phi & 1 \end{bmatrix}
\]

Left wheel

\[
\mathbf{A}_{bc} = \mathbf{A}_x^{(\delta)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\delta & -s\delta \\ 0 & s\delta & c\delta \end{bmatrix}
\]

\[
\mathbf{A}_{T_c} = \mathbf{A}_{T_b} \mathbf{A}_{bc} = \begin{bmatrix} 1 & -\psi c(\phi + \delta) & \psi s(\phi + \delta) \\ \psi & c(\phi + \delta) & -s(\phi + \delta) \\ 0 & s(\phi + \delta) & c(\phi + \delta) \end{bmatrix}
\]

Right wheel

\[
\mathbf{A}_{bc} = \mathbf{A}_x^{(\delta)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\delta & s\delta \\ 0 & -s\delta & c\delta \end{bmatrix}
\]

\[
\mathbf{A}_{T_c} = \mathbf{A}_{T_b} \mathbf{A}_{bc} = \begin{bmatrix} 1 & -\psi c(\phi - \delta) & \psi s(\phi - \delta) \\ \psi & c(\phi - \delta) & -s(\phi - \delta) \\ 0 & s(\phi - \delta) & c(\phi - \delta) \end{bmatrix}
\]

Table B.2: Approximate rotation matrices. 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.

<table>
<thead>
<tr>
<th>Description</th>
<th>Left wheel</th>
<th>Right wheel</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>Given directly from RSGEO</td>
<td>Flipped around the z-axis</td>
</tr>
<tr>
<td>N0</td>
<td>Given directly from RSGEO</td>
<td>Flipped around the z-axis</td>
</tr>
<tr>
<td>δ</td>
<td>Given directly from RSGEO</td>
<td>Flipped around the z-axis and the y-axis</td>
</tr>
<tr>
<td>a0</td>
<td>Given directly from RSGEO</td>
<td>Flipped around the z-axis</td>
</tr>
<tr>
<td>b0</td>
<td>Given directly from RSGEO</td>
<td>Flipped around the z-axis</td>
</tr>
<tr>
<td>Rwy</td>
<td>Given directly from RSGEO</td>
<td>Flipped around the z-axis and the y-axis</td>
</tr>
<tr>
<td>Rwz</td>
<td>Flipped around the y-axis</td>
<td>Flipped around the z-axis and the y-axis</td>
</tr>
<tr>
<td>C11</td>
<td>Given directly from RSGEO</td>
<td>Flipped around the z-axis</td>
</tr>
<tr>
<td>C22</td>
<td>Given directly from RSGEO</td>
<td>Flipped around the z-axis</td>
</tr>
<tr>
<td>C23</td>
<td>Given directly from RSGEO</td>
<td>Flipped around the z-axis</td>
</tr>
<tr>
<td>Rrz</td>
<td>Given directly from RSGEO</td>
<td>Flipped around the z-axis</td>
</tr>
<tr>
<td>q0</td>
<td>Given directly from RSGEO</td>
<td>Flipped around the z-axis</td>
</tr>
<tr>
<td>Rry</td>
<td>Given directly from RSGEO</td>
<td>Flipped around the z-axis and the y-axis</td>
</tr>
</tbody>
</table>
### APPENDIX C. DATA FROM RSgeo TABLE

<table>
<thead>
<tr>
<th>y [m]</th>
<th>N0 [N]</th>
<th>δ [rad]</th>
<th>a0 [m]</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.01700</td>
<td>66604.4707017493</td>
<td>0.043652537</td>
<td>0.00656931</td>
</tr>
<tr>
<td>-0.01699</td>
<td>66604.4385295224</td>
<td>0.043663594</td>
<td>0.00656874</td>
</tr>
<tr>
<td>-0.01698</td>
<td>66604.4174969331</td>
<td>0.043670821</td>
<td>0.00656819</td>
</tr>
<tr>
<td>-0.01697</td>
<td>66604.4075862261</td>
<td>0.043674226</td>
<td>0.00656766</td>
</tr>
<tr>
<td>-0.01696</td>
<td>66604.4087883589</td>
<td>0.043673813</td>
<td>0.00656717</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>h0 [m]</th>
<th>Rwy [m]</th>
<th>Rwz [m]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00352271</td>
<td>0.775173707</td>
<td>0.42454303</td>
</tr>
<tr>
<td>0.00352389</td>
<td>0.775167656</td>
<td>0.42454328</td>
</tr>
<tr>
<td>0.00352509</td>
<td>0.775161599</td>
<td>0.42454354</td>
</tr>
<tr>
<td>0.00352628</td>
<td>0.775155535</td>
<td>0.42454379</td>
</tr>
<tr>
<td>0.00352748</td>
<td>0.775149464</td>
<td>0.42454404</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C11 [-]</th>
<th>C22 [-]</th>
<th>C23 [-]</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.07979826325327</td>
<td>4.80394451402366</td>
<td>2.49187187388858</td>
</tr>
<tr>
<td>5.07904342362409</td>
<td>4.80296585757342</td>
<td>2.49091799021527</td>
</tr>
<tr>
<td>5.0782938146116</td>
<td>4.80199370040575</td>
<td>2.48998163323036</td>
</tr>
<tr>
<td>5.07754943371750</td>
<td>4.80102804220109</td>
<td>2.48904108418390</td>
</tr>
<tr>
<td>5.07680484571060</td>
<td>4.80005937879926</td>
<td>2.48811338946733</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rrz [m]</th>
<th>qpen0 [m]</th>
<th>Rry [m]</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.000052853178</td>
<td>0.000118493</td>
<td>0.7518796300</td>
</tr>
<tr>
<td>-0.000052816988</td>
<td>0.000118492</td>
<td>0.7518861589</td>
</tr>
<tr>
<td>-0.000052780971</td>
<td>0.000118491</td>
<td>0.7518926839</td>
</tr>
<tr>
<td>-0.000052744996</td>
<td>0.000118489</td>
<td>0.7518992057</td>
</tr>
<tr>
<td>-0.000052708895</td>
<td>0.000118488</td>
<td>0.7519057244</td>
</tr>
</tbody>
</table>

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.

59
APPENDIX D. TEST OF COMPONENTS SEPARATELY

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

```matlab
% clear all
% close all
% clc

% 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-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,... % rear wheel
        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,... % rear wheel
%           0, 0, 0, 0, 0, 0, 0,... % bogie frame
```

65
66

APPENDIX E. MATLAB CODE

% 0, 0]; % spin perturbation
% qstart = [0.0, z_start, 0, 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, ... % 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, ... % 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 axle
% q(26) : [rad/s] Spin perturbation rear axle

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Bogie and wheel characteristics
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
parm.a = 0.7563006418; % [m] Half the track gauge
parm.b = 1.074; % [m] Half the distance between the two axle
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] Vertical distance from spring to gravity
parm.h2 = 0.6584; % [m] Vertical 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. lwy = 80; % [kg m^2] M.o.i for the pitch motions of the wheel around the y-axis
parm. lwz = 678; % [kg m^2] M.o.i for the yaw motions of the wheel around the x-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 x-axis
parm. r0 = 0.4248829; % [m] Nominal rolling 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

calling function system

parm. Fwfc = (parm.mw + 1/2 * parm.mf + 1/4 * parm.mc) * parm.g * parm. r0; % [N]

Material damping force of wheel-rail contact

Material properties

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

calling function system

% from Carsten, where you can define the stepsize and tol
123 \[ T, q \] = \texttt{ode45}(@\texttt{bogie}, tspan, qstart, opts, parm, rsgeo); \% This should be the first solver you try - err with butttab = '45DP'
124 \hspace{1em} \texttt{Dormand Prince}
125 \hspace{1em} % \[ T, q \] = \texttt{ode15s}(@\texttt{bogie}, tspan, qstart, opts, parm, rsgeo); \% If \texttt{ode45}
126 \hspace{1em} \texttt{is slow because the problem is stiff.}
127 \texttt{toc}
128
129 \% Plot q
130 \texttt{plot\_result}(T, q, 'front')
131 \texttt{plot\_result}(T, q, 'rear')
132 \texttt{plot\_result}(T, q, 'frame')
133 \texttt{plot\_hyste}
134 \texttt{plot\_normal}
135 \texttt{plot\_creep}
136 \texttt{envelope}
137 \texttt{compare\_maxval}

\textbf{E.2 bogie.m}

\begin{verbatim}
function dq = bogie(t,q,parm,rsgeo)

% BOGIE
% % Call: bogie(t,q,parm,rsgeo)
% % Input Parameters: t, is the time.
% % q, array containing the position and 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 > \texttt{ceil}(t)-0.001
    disp(['\texttt{Time: ',\texttt{num2str}(t)]})
end

% Read parameters:
% mw = parm.mw; \% [kg] Mass of the wheel axle
% lwx = parm.lwx; \% [kg*m^2] M.o.i. for the roll motions of the
% wheel around the x-axis
% lwy = parm.lwy; \% [kg*m^2] M.o.i. for the pitch motions of the
% wheel around the x-axis
% lwz = parm.lwz; \% [kg*m^2] M.o.i. for the yaw motions of the
% wheel around the z-axis
\end{verbatim}
% \( \text{mf} = \text{parm}.m \text{f} \); % [kg] Mass of the frame
% \( \text{Ifx} = \text{parm}.\text{Ifx} \); % [kg\(\cdot\)m\(^2\)] M.o.i. for the yaw motions of the
frame around the x-axis
% \( \text{Ifz} = \text{parm}.\text{Ifz} \); % [kg\(\cdot\)m\(^2\)] M.o.i. for the roll motions of the
frame around the z-axis
% \( \text{a} = \text{parm}.\text{a} \); % [m] Half the track gauge
% \( \text{b} = \text{parm}.\text{b} \); % [m] Half the axle distance
% \( \text{h1} = \text{parm}.\text{h1} \); % [m] Vertical distance from spring to gravity
% \( \text{h2} = \text{parm}.\text{h2} \); % [m] Vertical distance from spring to gravity
% \( \text{ro} = \text{parm}.\text{ro} \); % [m] Nominal rolling radius of the wheel
% \( Fwfc = \text{parm}.Fwfc \); % [N] Gravitational load for one wheel axle, \( Fwfc = (\text{mw}+1/2\times\text{mf}+1/4\times\text{mc})\times g \)

% Allocate space:
\( dq = \text{zeros}(26,1) \);

% Compute spring forces:
\( A = \text{spring\_force}(t,q,\text{parm}) \);

% Compute damper forces:
\( D = \text{damper\_force}(t,q,\text{parm}) \);

% Compute normal forces:
\( \text{if 1 \% used debugging} \)
\[ [\text{NFL}, \text{NZFL}, \text{QPENFL}, \text{RWYFL}, \text{RZWFL}] = \text{normal\_force}(t,q,\text{parm},rsgeo, 'front', 'left'); \]
\[ [\text{NFR}, \text{NZFR}, \text{QPENFR}, \text{RWYFR}, \text{RZWFR}] = \text{normal\_force}(t,q,\text{parm},rsgeo, 'front', 'right'); \]
\[ [\text{NRL}, \text{NZRL}, \text{QPENRL}, \text{RWYRL}, \text{RZWRL}] = \text{normal\_force}(t,q,\text{parm},rsgeo, 'rear', 'left'); \]
\[ [\text{NRR}, \text{NZRR}, \text{QPENRR}, \text{RWYRR}, \text{RZWRR}] = \text{normal\_force}(t,q,\text{parm},rsgeo, 'rear', 'right'); \]
\( \text{else} \)
\( \text{NFL} = \text{zeros}(3,1); \)
\( \text{NFR} = \text{zeros}(3,1); \)
\( \text{NRL} = \text{zeros}(3,1); \)
\( \text{NRR} = \text{zeros}(3,1); \)
\( \text{RWYFL} = \text{parm}.\text{a}; \)
\( \text{RWYFR} = -\text{parm}.\text{a}; \)
\( \text{RWYRL} = \text{parm}.\text{a}; \)
\( \text{RWYRR} = -\text{parm}.\text{a}; \)
\( \text{RZWFL} = -\text{parm}.\text{r0}; \)
\( \text{RZWFR} = -\text{parm}.\text{r0}; \)
\( \text{RZWRL} = -\text{parm}.\text{r0}; \)
\( \text{RZWRR} = -\text{parm}.\text{r0}; \)
\( \text{end} \)

% Compute creep forces:
\( \text{if 1 \% used debugging} \)
\( \text{FCFL} = \text{creep\_force}(t,q,\text{parm},\text{NFL},\text{QPENFL},rsgeo,'front','left'); \)
\( \text{FCFR} = \text{creep\_force}(t,q,\text{parm},\text{NFR},\text{QPENFR},rsgeo,'front','right'); \)
APPENDIX E. MATLAB CODE

FCrl = creep_force(t, q, parm, Nzrl, qpenrl, rsgeo, 'rear', 'left');
FCrr = creep_force(t, q, parm, Nzrr, qpenrr, rsgeo, 'rear', 'right');
else
  FCfl = zeros(3,1);
  FCfr = zeros(3,1);
  FCrl = zeros(3,1);
  FCrr = zeros(3,1);
end

if 1 % used debugging
  % Compute dynamics of front wheel:
  dq(1) = q(2);  % [m] lateral position (q1)
  dq(2) = (-A(1)+FCfl(2)+FCfr(2)+Nfl(2)+Nfr(2))/parm.mw;  % [m/s]
    lateral velocity (q2dot) -(-A1)/mw
  dq(3) = q(4);  % [m] vertical position (q2)
  dq(4) = (-A(2)+FCfl(3)+FCfr(3)+Nfl(3)+Nfr(3)-parm.Fwfc)/parm.mw;
    % [m/s] vertical velocity (q3dot) -(-A2)/mw
  dq(5) = q(6);  % [rad] yaw angle (q3)
  dq(6) = (-A(3)-Rwyfl*(FCfl(1)+q(5))*(FCfl(2)+Nfl(2)))-Rwyfr*(FCfr(1)+q(5))*(FCfr(2)+Nfr(2)))/parm.lwz;
    % [rad/s] yaw angle velocity (q4dot) -(-A3)/lwz
  dq(7) = q(8);  % [rad] roll angle (q4)
  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))...-
    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.lwz;
    % [rad/s] roll angle velocity (q5dot) -(-A4)/lwz
  dq(25) = (Rwzfl*(FCfr(1)+FCfr(2)+Nfr(2))*q(5))+Rwzfl*(FCfl(1)+
    (FCfl(2)+Nfl(2))*q(5)-A(13))/parm.lwz;
    % [rad] Spin perturbation front wheel
end

if 1 % used debugging
  % Compute dynamics of rear wheel:
  dq(9) = q(10);  % [m] lateral position (q5)
  dq(10) = (-A(5)+FCrl(2)+FCrr(2)+Nrl(2)+Nrr(2))/parm.mw;  % [m/s]
    lateral velocity (q6dot) -(-A5)/mw
  dq(11) = q(12);  % [m] vertical position (q6)
  dq(12) = (-A(6)+FCrl(3)+FCrr(3)+Nrl(3)+Nrr(3)-parm.Fwfc)/parm.mw;
    % [m/s] vertical velocity (q7dot) -(-A6)/mw
  dq(13) = q(14);  % [rad] yaw angle (q7)
  dq(14) = (-A(7)-Rwyrl*(FCrl(1)+q(13))*(FCrl(2)+Nrl(2)))-Rwyrr*(FCrr(1)+q(13))*(FCrr(2)+Nrr(2)))/parm.lwz;
    % [rad/s] yaw angle velocity (q8dot) -(-A7)/lwz
  dq(15) = q(16);  % [rad] roll angle (q8)
  dq(16) = (-A(8)+Rwyrl*(FCrl(3)+Nrl(3)-q(15))*(FCrl(2)+Nrl(2))+q(15))+Rwyrr*(FCrl(3)+Nrl(3)-q(15))*(FCrl(2)+Nrl(2))...-
    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.lwz;
    % [rad/s] roll angle velocity (q9dot) -(-A8)/lwz
  dq(26) = (Rwzrl*(FCrl(1)+FCrl(2)+Nrl(2))*q(13))+Rwzrl*(FCrl(1)+
    (FCrl(2)+Nrl(2))*q(13)-A(14))/parm.lwz;
    % [rad] Spin perturbation rear wheel
end

if 1 % used debugging
  % Compute dynamics of bogie frame
  dq(17) = q(18); % [m] lateral position (q9)
  dq(18) = (A(1)+A(5)+A(9)+D(1))/parm.mf; % [m/s] lateral
    velocity (q9dot) (A1+A5+A9+D1)/mf
  dq(19) = q(20); % [m] vertical position (q10)
  dq(20) = (A(2)+A(6)−A(10)−D(2))/parm.mf; % [m/s] vertical
    velocity (q10dot) (A2+A6−A10−D2)/mf
  dq(21) = q(22); % [rad] yaw angle (q11)
  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
  dq(23) = q(24); % [rad] roll angle (q12)
  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+...)
end
end

E.3 spring_force.m

function A = spring_force(t,q,parm)
% SPRING_FORCE used for calculating the current spring force
% %
% % Call: spring_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 spring force
% %
% % Authors: % Ulla Uldahl
% % Date: % September 24, 2011
% %
% % Read parameters:
% % b = parm.b; % [m] Half the distance between the two axels
d1 = parm.d1; % [m] Horizontal distance fro spring to
gavity
% d2 = parm.d2; % [m] Horizontal distance fro spring to
gavity
% h1 = parm.h1; % [m] Vertical distance fro spring to gravity
h2 = parm.h2; % [m] Vertical distance fro spring to gravity
% k1 = parm.k1; % [N/m] Spring coefficient wheel/frame lateral
k2 = parm.k2; % [N/m] Spring coefficient wheel/frame track−
direction
% 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 gravity
% h2 = parm.h2; % [m] Vertical distance from spring to gravity
% 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 containing the position and 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 containing the normal force in the track coordinate 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 rolling radius (vertical position of the contact point on the wheel in the body system).
% Author: Ulla Uldahl
% Date: September 22, 2011

% rsgeo(:,1) : Lateral displacement, \{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, Rw [m]
% rsgeo(:,7) : Actual rolling radius (vertical position of the contact point on the wheel in the body system), Rwz [m]
% rsgeo(:,8) : Kalker's creepage, C11 [-]
% rsgeo(:,9) : Kalker's creepage, C22 [-]
% rsgeo(:,10) : Kalker's 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:
rsgeo_in = [rsgeo(:,2), slr*rsgeo(:,3), slr*rsgeo(:,6), -rsgeo(:,7),
rsgeo(:,11), slr*rsgeo(:,12), slr*rsgeo(:,13)];

rsgeo_out = interp1(rsgeo(:,1), rsgeo_in, slr*y, 'linear');
rsgeo_out = linear_interp([rsgeo(:,1), rsgeo_in], slr*y);

N0 = rsgeo_out(1);
delta = rsgeo_out(2);
Rwy = rsgeo_out(3);
Rwz = rsgeo_out(4);
qpen0 = rsgeo_out(6);
Rry = rsgeo_out(7);

yeq = 0;
zeq = parm.r0;

% Compute center of mass of body in track system:
Rcy = yeq+y;
Rcz = zeq+z;

% Compute dynamic penetration depth:

cosdelta = cos(delta);
sindelta = sin(delta);
qpen = -(Rwy-Rcy+phi*(Rrz-zeq)-Rwy)*sindelta+(Rrz-Rcz-phi*(Rry-yeq)
-Rwz)*cosdelta;

% Compute dynamic penetration velocity:
vpen = (ydot-phiidot*(Rrz-zeq))*sindelta-(zdot+phiidot*(Rry-yeq))*
cosdelta;

% Compute size of normal force in wheel–rail contact system ( scalar):
Nz_spring = N0*(qpen/qpen0)^(3/2);

% Compute the material damping force:
Nz_damper = parm.Dm*vpen;

% Compute normal force with damping force:
Nz = Nz_spring + Nz_damper;

if ~isreal(Nz) || Nz < 0
    Nz = 0;
end

% Transform normal force vector from contact to track system:
sinphidelta = sin(phi+delta);
Ntrack = [psi*sinphidelta; -sinphidelta; cos(phi+delta)]*Nz;

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 containing the position and 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 containing the creep force in the track coordinate 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 displacement, {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 rolling 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); %
psi = q(5);
psidot = q(6);
phi = q(7);
phidot = q(8);
betadot = q(25);  % Spin perturbation front wheel, [rad]
else  % rear wheelset
  y = q(9);
ydot = q(10);
zdot = q(12);
psi = q(13);
psidot = q(14);
phi = q(15);
phidot = q(16);
betadot = q(26);  % Spin perturbation rear wheel, [rad]
end

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

% Interpolate RSGEO values:
rsgeo_in = [slr*rsgeo(:,3),rsgeo(:,4),rsgeo(:,5),slr*rsgeo(:,6),-
            rsgeo(:,7),rsgeo(:,8),rsgeo(:,9),rsgeo(:,10),rsgeo(:,12)];
rsgeo_out = interp1(rsgeo(:,1),rsgeo_in,slr*y,'linear');

delta = rsgeo_out(1);
a0 = rsgeo_out(2);
b0 = rsgeo_out(3);
Rwy = rsgeo_out(4);
Rwz = rsgeo_out(5);
C11 = rsgeo_out(6);
C22 = rsgeo_out(7);
C23 = rsgeo_out(8);
qpen0 = rsgeo_out(9);

if t > parm.t0 && parm.a0 ~= 0
  parm.v = parm.v0 + parm.a0*(t-parm.t0);
end
cosdelta = cos(delta);
sindelta = sin(delta);

% Read parameters:
omega = parm.v/parm.r0;  % Nominal spin of wheel, [1/s]

% Compute the longitudinal creepage:
xi = x+(psidot+phi-zdot*(omega+betadot)-Rwy*psidot)/parm.v;

% Compute the lateral creepage:
xiy = (ydot-psi*parm.v+phi*zdot-phidot*Rwz)/(parm.v*cosdelta);
% Compute the spin creepage:
xisp = (−sin delta*(omega+betadot)+cos delta*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:
coshdelta = cos(phi+delta);
Ftrack = [Fx−psi*coshdelta*Fy;
          psi*Fx+coshdelta*Fy;
          sin(phi+delta)*Fy];

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
To simplify input, the options AbsTol and RelTol are replaced by the option `opts(2)`. And if we choose \( \text{AbsTol} = opts(2)^2 \) and \( \text{RelTol} = opts(2) \), then \( \text{AbsTol} + \text{RelTol} \times \text{norm}(x, \infty) = opts(2)^2 + opts(2) \times \text{norm}(x, \infty) \).

---

### Butcher Tableau

<table>
<thead>
<tr>
<th>Switch</th>
<th>Buttab</th>
<th>Buttab</th>
<th>Buttab</th>
</tr>
</thead>
<tbody>
<tr>
<td>'21HE'</td>
<td>'23BS'</td>
<td>'45F'</td>
<td>'45DP'</td>
</tr>
<tr>
<td>'32BS'</td>
<td>'23BS'</td>
<td>'45F'</td>
<td>'ode45'</td>
</tr>
<tr>
<td>'21HE'</td>
<td>'23BS'</td>
<td>'45F'</td>
<td>'ode45'</td>
</tr>
<tr>
<td>'32BS'</td>
<td>'23BS'</td>
<td>'45F'</td>
<td>'ode45'</td>
</tr>
</tbody>
</table>

```matlab
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 ;
         3/8 3/32 9/32 0 0 0 ;
         0 ;
         12/13 1932/2197 76720/2197 7296/2197 0 0 ;
         1 439/216 2 1639/512 0 0 ;
         0 ;
         1/2 7/28 2 3544/2565 1859/4104 0 ;
         -11/40 0 ];
```
APPENDIX E. MATLAB CODE

4 25/216 0 1408/2565 2197/4104
-1/5 0 ;
5 16/135 0 6656/12825 28561/56430
-9/50 2/55 ;
case '54CK' % Cash-Karp (6-stage)
B = [0 0 0 0 0 0 0 0 0 0 ;
1/5 1/5 0 0 0 0 0 0 0 0 ;
3/10 3/40 9/40 0 0 0 0 0 0 0 ;
3/5 3/10 -9/10 6/5 0 0 0 0 0 0 ;
1 -11/54 5/2 -70/27 35/27 0 0 0 0 0 ;
7/8 1631/55296 175/512 575/13824 44275/110592 253/4096 0 0 0 0 ;
5 37/378 0 250/621 125/594 0 512/1771 ;
4 2825/27648 0 18575/48384 13525/55296 277/14336 1/4 ] ;
case '45DP' % Dormand? Prince (7-stage)
B = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ;
1/5 1/5 0 0 0 0 0 0 0 0 ;
3/10 3/40 9/40 0 0 0 0 0 0 0 ;
4/5 44/45 -56/15 32/9 0 0 0 0 0 0 ;
8/9 19372/6561 -25360/2187 64448/6561 -212/729 0 0 0 0 0 0 ;
1 9017/3168 -355/33 46732/5247 49/176 -5103/18656 0 0 0 0 ;
1 35/384 0 500/1113 125/192 -2187/6784 11/84 0 0 0 0 ;
4 5179/57600 0 7571/16695 393/640 -92097/339200 187/2100 1/40 ;
5 35/384 0 500/1113 125/192 -2187/6784 11/84 0 0 0 0 ;
end
A = B(1:end − 2, 2:end)' ;
b = B(end − 1, 2:end)' ;
b = B(end, 2:end)' ;
c = B(1:end − 2, 1) ;
d = b − bt ;
k = B(end, 1) ; % order of error control
% initial value ...
t = tspan(1) ;
y = y0(:, ) ;
f = leval(fun, t, y, varargin{ : }) ;
% initial step-size (local truncation error of explicit Euler method)...
if isempty(h0)
    J = numjac(fun, t, y, f, tol, varargin{:});
    h = 0.8*(2*tol*(tol + norm(y, inf))/norm(J*f, inf))^(1/2);
else
    h = h0;
end

% controller parameters (PI controller)...
kI = 0.3/k;
kP = 0.4/k;
racc = tol*(tol + norm(y, inf));

% initialize containers, counters, etc....
m = length(y); % no. of ODE's
s = length(c); % no. of stages
N = round((tspan(2) - tspan(1))/h); % containers in chunks N
tout = repmat(t, [1 N]); yout = repmat(y, [1 N]);
trace = (nargout > 3); if trace
    perf = repmat([0; 0; 0], [1 N]);
    err = repmat(zeros(length(y),1), [1 N]);
end
F = zeros(m, s); % for storing stages
ntot = 0; % no. of steps in total
nacc = 0; % no. of accepted steps
nrej = 0; % no. of rejected steps, saved in perf -> as (t, nrej)
nfun = 1; % no. of function evaluations

% integrate...
w = waitbar(0,'Integrating, please wait...');
while t < tspan(2)
    ntot = ntot + 1;
    % show progress...
    % waitbar(t/tspan(2))
    % stage 1...
    T = t;
    Y = y;
    F(:, 1) = f;
    % stage 2-s...
    for j = 2:s
        nfun = nfun + 1;
        i = j;
        T = t + h*c(i);
        Y = y + h*F(:, 1:j-1)*A(1:j-1, i);
        F(:, i) = feval(fun, T, Y, varargin{:});
    end
    % estimate local truncation error...
e = h*F*d;
    r = norm(e, inf);
    % accept step if r <= tol*(tol + norm(y, inf))...
    step_accepted = 0;
    if r <= tol*(tol + norm(y, inf)) || ~varstep
        step_accepted = 1;
nacc = nacc + 1;
nfun = nfun + 1;

% update current point...
t = t + h;
y = y + h*F*b; % using advancing method
% y = y + h*F*bt; % using embedded method
f = feval(fun, t, y, varargin{:});

% save output and performance...
tout(nacc+1) = t;
yout(:, nacc+1) = y;
if trace
    perf(:, nacc+1,:) = [h; r; nrej];
    err(:, nacc+1,:) = e;
end

% expand containers in chunks of N...
if ~mod(nacc + 1, N)
tout = cat(2, tout, repmat(t, [1 N]));
yout = cat(2, yout, repmat(y, [1 N]));
if trace
    perf = cat(2, perf, repmat(perf(:, end), [1 N]));
    err = cat(2, err, repmat(err(:, end), [1 N]));
end
end
nrej = 0;

% variable step size BEGIN

if varstep

    % update stepsize (asymptotic controller)...
    % h = h*max(0.1, min((0.8*tol*(tol + norm(y, inf))/r)^(1/k), 10));
    
    % update stepsize (PI controller)...
    if step_accepted
        h = h*max(0.1, min((0.8*tol*(tol + norm(y, inf))/r)^kI *racc/r)^(1/k), 10));
        racc = r;
    else
        nrej = nrej + 1;
        h = h*max(0.1, min((0.8*tol*(tol + norm(y, inf))/r)^(1/k), 10));
    end
end

% variable step size END

% ensure t + h <= tend...
h = min(h, tsnap(2) - t);

% close(w)

% build information, solution and performance...
info = [ntot ntot-nacc nfun];
tout = tout(1:nacc+1);
yout = yout(:, 1:nacc+1);
if trace, perf = perf(:, 1:nacc+1); err = err(:, 1:nacc+1);
end

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

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

% Call:
% [y, dydx] = linear_interp(S, x)
% Input:
% S : Set of m equally 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
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)
% Linear interpolated responses at sample point x:
y = S(i,:2:end) + (x - S(i,1))\times dydx;
else
    disp('i >= size(S,1)')
    if x > S(end,1)
        error(['x must be smaller than', num2str(S(end,1))])
    end
% Gradient of responses between design sites S(end-1,1) and S(end,1):
    dydx = (S(end,:2:end) - S(end-1,:2:end))/dx;
    dydx = (S(end,:2:end) - S(end-1,:2:end))/(S(end,1) - S(end-1,1))
% Linear interpolated responses at sample point x:
y = S(end,:2:end);
Bibliography


