MASTER

Vehicle modelling by using tensors and a modular structure

Rijnbout, J.

Award date:
2001

Link to publication
Vehicle modelling by using tensors and a modular structure

By Jasper Rijnbout

Master's Thesis
Carried out from September 2000 to August 2001
Commissioned by: Prof. Dr. Ir. P.P.J. van den Bosch
Under supervision of: Ir. B. Stouten
Dr. Ir. A.A.H. Damen
Location: Eindhoven University of Technology
Department of Electrical Engineering
Measurement and Control Systems Group
Control Systems Research Chair
Date: 17 August 2001
Summary

This Master Thesis describes the research done to conclude my study of Electrical Engineering at the Eindhoven University of Technology. The research took place within the scope of a large project (named HOV) for a public transportation system with controlled, all-wheel steered, multiple-articulated busses in the city of Eindhoven.

The research consisted of three separate parts. First a literature survey was done to find out if other researchers had done research on this subject. The most important point of interest was vehicle state estimation techniques for multiple-articulated vehicles.

No references were found on this specific subject. There were however a lot of references found on vehicle state estimation, but these all used 2 or 4 wheel vehicles. It was interesting to see that almost every reference used some sort of Kalman observer for the state estimation. A number of references were found describing articulated vehicles and platooning systems (where unconnected vehicles were to follow a leader). But none of these described for HOV2 useful state estimation techniques. The few references that did use articulated vehicles assumed perfect knowledge about the position of the vehicle and/or the vehicle states, and were therefore of little interest.

The second part of the research gives a description of the use of tensors for coordinate system transformations. An extensive introduction of how tensors operate is given with four increasingly complex examples. This chapter finally concludes that using tensors can result in much compactier notation for coordinate transformations of vectors and points, and reveals an important difference between vectors and points. Furthermore when used properly, the high degree of coordinate system independency of tensors can prolong the choice for the appropriate coordinate system to the very end of a calculation, where a numeric implementation is required.

An appendix describes a special way of using a combination of homogenous coordinates and tensors to better deal with movements containing translations. It is added as an appendix because a number of points needed for a good comprehension are still unclear in this method.

The third part describes a number of modules that can be used to build various implementations of the same dynamic vehicle model and some other useful structures. All these modules are fully described with their mathematical equations. After the description two vehicle models are constructed by using these modules. A special structure can be used to find the required "vehicle inputs" (steering angles, and longitudinal tire forces) to realise a specified path in time.

The use of this "path-to-inputs" structure in vehicle control systems is then proposed and roughly described. Finally the possible advantages and disadvantages of the modular system are listed, concluding that the modular structure could be advantageous especially for people that are new in the field of vehicle modelling because it may be easier understood and handled than just the set of equations.

Finally, conclusions are drawn and recommendations are made for further research.
Introduction

For the master thesis to conclude my Electrical Engineering study at the Eindhoven University of Technology I had to carry out a research project. I chose to do this project at the research chair Control Systems of the section Measurement and Control Systems at the Electrical Engineering department. One of the projects the people of the Control Systems chair were working on was the HOV project (Hoogwaardig Openbaar Vervoer: Dutch for high quality public transportation). This project researched a public transportation system with all-wheel steered, multiple-articulated busses in the city of Eindhoven. The busses must follow magnetic markers placed in the road. The advantages of this system over traditional busses would be: more passenger capacity (due to the increased size of the bus), a decrease in the required dimensions of the road (which can be a great advantage in city centres) and more accurate stops at bus stops. The advantages over traditional railroad vehicles that already have a minimal road requirement are the reduced cost of the infrastructure. Additionally, if the system works as expected, the driver would technically not be needed, although this last advantage is very dependent on public acceptation. The HOV system will be operational in the year 2003.

As a follow up for this project (named HOV2) initial research was started for a system where all-wheel steered, multiple-articulated busses would not need any fixed markers, resulting in a far more flexible transportation system. However, a driver is necessary in this system. The general idea of HOV2 was a system where the driver would somehow indicate the path for the bus, and the control system for the bus would steer the bus to follow the indicated path in an optimised way (the optimisation could for example be to keep the required road-width for movements of the bus as small as possible).

During the research on the HOV project, it became clear that good vehicle state estimation was essential for good control of multiple-articulated vehicles, because the vehicle states are often hard to measure directly with acceptable accuracy. This is exactly why I start my research with a literature survey focussing on vehicle state estimation for articulated vehicles. The main objective of this literature survey is to find out if methods of vehicle state estimation for this purpose are (being) developed that are different from the method used in the HOV project, and potentially better. If other useful methods are found, these can be implemented and tested in the HOV2 project.

For vehicle state estimation, the HOV project uses a Kalman observer that uses (amongst others) a number of acceleration sensors on the vehicle. It turned out that it was sometimes hard to see exactly which accelerations were measured (for example what the influence of Earth’s rotation was). The mathematical representation of transformations between coordinate systems that was used, although being mathematically correct, made it relatively easy to make a mistake in these transformations and hard to find these mistakes. Hoping to avoid similar problems in HOV2 another method of describing these transformations is proposed, that represents the transformations with tensors instead of matrices. The big advantage of tensors is that they basically operate independent of coordinate systems. The hypothesis that the use of tensors will make vehicle modelling more straight forward and can decrease the occurrence of errors in coordinate system transformations will have to be shown in this thesis to support the hypothesis. Of course the way tensors operate will have to be investigated first. Although the vehicle model used in the HOV project can be adapted to use tensors and will definitely be useful in HOV2, another way of constructing a vehicle model will be investigated, hopefully showing some new insights and possibly providing a model that is better in some way than the model already developed. The possible improvements will most likely not result in a more accurate model; the basis and the initial assumptions will after all be the same. However, improvements in ease of use, adaptability and ease of interpretation are possible.

A number of techniques exist that can be used to construct a vehicle model. One method looks at the entire vehicle and directly derives the motion equations from this. This method can only be worked out analytically if some approximations are used (for example constant speed and/or small deviations from a straight line). The result is a linear vehicle model. The method of Lagrange is another way of obtaining a vehicle model, using the kinetic and potential energies of the vehicle. For fairly simple vehicle systems, it is often overly complex compared to other methods. The method of Lagrange is far more useful with complex vehicle system.

This thesis describes another method that actually splits the entire model in functional blocks. These blocks (modules) can be described and understood separately from each other. This way it should be easier to recognise the physical structure in the model (compared to just the set of motion equations). Constructing a dynamic vehicle model will start with a basic module describing the kinematic movement of the vehicle. Other modules are then added to take into account the forces and tire-slip effects. Possibly other useful structures can be constructed by combining the different modules.
Chapter 1: Literature survey

The main objective of this literature survey done for the HOV2 project was to find information about state-estimation (especially the side-slip). Besides this main objective any literature found with a connection to the HOV2 project, was of interest (especially driver-modelling).

1.1 Articles found on vehicle state-estimation

In [9] a fairly complex estimation system is proposed, using dynamic tire load model to calculate the tire loads (vertical force on each road/road contact). A fuzzy speed estimator using the rotational wheel speeds and the estimated sideslip angle makes an estimation of the longitudinal wheel slip (for each wheel) and the vehicle speed. These longitudinal wheel slips, together with the tire loads are used in an RLS (recursive least squares) estimator, to estimate tire cornering-stiffness. Finally the measured yaw-rate and steering angle are used with the estimated vehicle speed and estimated cornering stiffness in an extended Luenberger observer. Some real-life tests are performed. Article [10] uses a simple 2 DOF bicycle model in a Kalman filter observer. The estimated slip angle is used in a desired model following controller for an active rear steering system. The system is tested in a real-life situation.

Article [11] uses a 9 DOF bicycle model, with analytic tire force model as reference. The model used for the extended Kalman filter is based on this model, but neglects the 4 DOF for the suspension modelling, and has no knowledge of the true tire force characteristics. The observer makes estimates of the lateral and longitudinal tire forces, lateral slip angles, longitudinal slips, front and rear axle velocities. Real-life tests are not performed, the observer (with white gaussian measurement noise) is compared to the full 9 DOF model. The observer performs very well in this simulation.

Article [12] describes field test results with an extended version (4 wheel instead of bicycle) of the system described in [11]. Field test results are comparable to simulation results. The influence of different road surfaces (friction coefficient ($\mu$)) needs further investigation, as this appears to introduce an offset in the slip estimates.

[13] and [14] are basically the same articles, with [14] being the most recent and more extensive one. These articles again use a Kalman filter to estimate the vehicle states. This time an extended Kalman filter is applied, which later on is made adaptive to increase performance. The observer is based on a 4 DOF model. The observer is compared to a high order model, with a Pacejka tire model. Final RMS error (relative to the signal) for the sideslip velocity is some 20-30 %.

Article [15] uses a combination of direct integration (independent of vehicle and road properties, but susceptible to sensor drift) and an observer (problems in the non-linear region of tire characteristics, but not $(\mu)$) susceptible to sensor drift. Some real-life tests were done on a snowy road.

[16] Starts with a Kalman filter to estimate sideslip angles. The observer is then made adaptive to compensate for changing cornering stiffness of the tires during different manoeuvres. The observer is tested in a real-life situation. In appendix 2 an overview of all different articles found is given, with various properties. The table can be used as a quick overview.

1.2 Articles found on multi-body vehicle control

Article [17] describes a control system for a TLV (train-like vehicle) consisting of multiple, flexibly connected cart. Each cart can be independently steered, by means of differential steering, and has only one axle. A kinematic model (without slip) is used as base for the controller. The absolute position of the leading cart is known (without explaining how) and used in a given algorithm to reconstruct the leading cart path. The following carts are steered to follow this path. Measures are taken to prevent (as much as possible) both mechanical and mathematical singularities. Real life tests are done, showing very good results.

[18] Is by the same author as [17], who started working on this subject in 1994. This article describes the same system as [17], it is however more recent but shorter.

[19] Proposes 5 different control algorithms for control of a platooning system, where multiple cars follow the leading car without any markers in the road (thus leaving only the position of the leader relative to the controlled vehicle available). Lateral control is fully automatic (relative to the leader), while longitudinal control is manual (a driver operating the accelerator). The five algorithms range from very simple (which only uses the relative position of the leading vehicle and the wheel base (distance between front and rear axle)) to complex (using full knowledge of state variables (velocity, acceleration, yaw rate and side slip angle) and parameters (cornering power of tires, mass, yaw inertia, wheel base and distance from axles to centre of gravity)). These algorithms are described, and
tested in simulation and real-life situations. The simulation uses a 3 DOF bicycle model. The simulations are carried out using different speed/distance combinations, and also with vehicle mass and cornering powers different from the values used in the control algorithm (if used at all). The simulations show unacceptable performance for three algorithms, which don’t use knowledge of slip angle. The other two algorithms show a tracking error of only 2-3 cm. Real-life road tests using the simplest of the two side slip algorithms show maximum tracking errors of 10 cm. For further improvement of the system, non-linearity, saturation and lag effect, and sensor errors should be taken in consideration.

Article [20] describes a way of slip-angle estimation to be used in an automated highway system. A kinematic bicycle model is again used, but this time magnetic markers are placed along the trajectory, which should be followed. The system proposed here estimates the slip angle from finding these markers, combined with accelerometers placed on the vehicle. Measures are taken to minimize the tracking errors occurring from the fact that due to marker spacing, marker position isn’t always available. Real life test are carried out, showing good sideslip prediction. However because of the markers, this article may have little use for the HOV2 project.

[21] Very shortly describes a lateral control system for a car following system (=platooning). In this system only the position of the leading vehicle relative to a controlled vehicle is known. The modelling used is very simple (it assumes perfect sensor data, and uses a kinematic Ackermann model which is limited to small velocities and small lateral accelerations), it is said however that extending the model (e.g. with slip effects) is pretty straightforward. The system transforms the position of the controlled vehicle and the relative position of the leading vehicle, to a coordinate system at rest. This way a sampled trajectory of the leading vehicle is acquired. Not the position of the leading vehicle, but the position of a trajectory point closer to the controlled vehicle is used as input for the lateral controller. The controller isn’t described, but in the tests a system that interpolates an arch is used. The simulation results show a maximum tracking error of 30-40 cm. A real-life test was also carried out, showing an error of up to 60 cm. These values are not acceptable for HOV2, but improvement is proposed by making the look-ahead distance (the distance of the trajectory point used as controller input) proportional to the vehicle speed. Furthermore, a platooning system allows for far greater deviations from the leader trajectory than an articulated vehicle (the distance between leader and following car in these tests was 25 m). This article, and article [19] could be useful to compare performance (e.g. turning corridor, and lateral tracking) of platooning to train-like vehicles (like Phileas)

1.3 References found on driver modelling

The literature survey on driver modelling is only preliminary. Some references are found on this subject, but the article texts have not been retrieved yet. Therefore only a short description based on the articles abstract is given. Because the complete texts have not been read, no selection could be made, so it is very well possible that many of these articles are actually not of interest to the HOV2 project.

[22] Addresses various subjects to improve vehicle stability control systems, of which driver intent recognition is only one.

[23] Presents a control theoretic model of driver steering behaviour, which compares favourably with driver simulations. The model contains both preview and compensatory steering dynamics.

[24] Develops a predictive driver model, consisting of low and high frequency compensation elements. The high frequency compensation is obtained by the application of the structural model of the human driver. Simulation shows accurate prediction.

[25] Proposes a model of environment-driver-vehicle interaction as a tool for analysing and designing a vehicle controller. It uses an extended Petri-net form using the hierarchical fuzzy integral (HFI) as a multipurpose decision making technique. Simulation shows good agreement with driving tests.

[26] Investigates handling characteristics of a medium bus during a single lane change, by computer simulation using DADS. A dynamic full bus model is used. The driver was modelled as a PID controller.

[27] A driver model with vehicle yaw and lateral motions is developed, based on data obtained from a driving simulator.

[28] Does not describe a driver model, but describes which factors influence driver (and passenger?) perception of ride quality. This could be useful for final evaluation of the HOV2 system. The most influential factors for ride quality are described, together with the best describing signal. Striking result is that the international ride quality standard (ISO2631) is not a good basis for assessing ride quality.

[29] Describes a driver model describing the dynamic individual driving behaviour in relation to the vehicle, the road environment and other traffic participants. The driver model includes free driving
(speed control), car following (distance control) and lane change behaviour. The driver model actually uses the vehicle controls with the aim to reach or maintain the individual, situation dependent desired speed, or following distance. The model was i.e. used to study the effects of adaptive cruise control on the traffic flow on a highway.

[30] Is a general article on the use of driver behaviour prediction by means of driver models, for a prospective assessment of vehicle developments. Several fields of industrial application are presented and future developments outlined. This article probably does not describe any driver models, but rather their application.

1.4 Main conclusions on found literature

Most literature found on the subject of vehicle state-estimation uses an observer similar to a Kalman-filter (linear or extended and/or adaptive). None of this vehicle state-estimation literature involves articulated vehicles.

Literature on articulated vehicles (and related multi-body set-ups) was found, but only described the control systems for these vehicles (no state estimation). It consisted of two subjects: train-like vehicles (with multiple connected vehicles, following the first vehicle) and platooning/automated highway systems (with multiple unconnected vehicles, following the first vehicle).

Concluding from the found literature on vehicle state estimation and articulated vehicles, the best way to construct a vehicle state estimation system for multiple-articulated vehicles appears to be trying to expand the vehicle state estimation methods (for non-articulated vehicles) to fit an articulated vehicle.

The references found on the subject of driver modelling indicate that this is pretty much an open issue. No practically useful conclusion can be drawn from the found references.
Chapter 2 : Summary of the use of tensors for coordinate transformations

After the literature survey was finished, a first simple vehicle model was described and discussed. This first implementation of the model will not be described in this thesis. While discussing this model, the same problems appeared as with the HOV project: confusion tends to appear when performing coordinate system (CS) transformations with transformation matrices and the components of vectors. Especially while calculating derivatives of points and vectors in a moving CS (coordinate system) it is not always easy to see if it is correct to take the time derivative in a certain CS, and the exact meaning of the result. We decided that before continuing the work on the vehicle model, an alternative method of describing CS transformations should be looked at. The use of tensors for this purpose looked promising, and was therefore proposed as the alternative method.

This chapter extensively describes the use of tensors in the field of coordinate transformations. First a simple 2-dimensional, 1 DOF case, with only a rotation is looked at. After that the situation is extended to a more general 3-dimensional, 3 DOF case, with one rotation and two translations. Four examples are given to illustrate the use of tensors. The last example describes a more practical situation that shows the importance of presuming the right coordinate system as the "motionless world".

2.1 Notation

The following notation will be used throughout the rest of this thesis, and is partly adapted from [2].

- $a, \vec{a}$: Scalar, vector
- $A, \vec{A}$: Column of scalars, column of vectors
- $A^T$: Transpose of $A$
- $A^{-1}$: Inverse of $A$
- $x_n$: $x$ relative to coordinate system $n$
- $\vec{n} \rightarrow \vec{m} A$: The matrix used in a matrix-representation of a tensor, $n$ and $m$ indicates the vector basis that should be used to represent the intended tensor: $\vec{A} = \sum_{m=1}^{n} A_{mn} \vec{e}_m$
- $c\varphi, s\varphi$: Cosine of $\varphi$, sine of $\varphi$
- $\text{Rot}(\vec{\varphi}, \varphi)$: Matrix for rotation over $\varphi$, around $\vec{\varphi}$ (see [1] for an extensive description)
- $\text{Trans}(\vec{\varphi}, p_x, p_y)$: Matrix for translation of $p_x$ and $p_y$, in direction $\vec{\varphi}$ and $\vec{\varphi}$ (see [1])
- $\vec{QR}$: Vector from point Q to point R
- $R = \overrightarrow{CR}$: This means: point R can be described with the vector $\overrightarrow{CR}$ (usually C will be the origin of a CS)
- $\vec{A}, \vec{\vec{A}}$: The vector indicating the 1st respectively 2nd time derivative of point $A$ (these vectors are independent of the CS)
- $\overrightarrow{OA}, \overrightarrow{\vec{OA}}$: The vector indicating the 1st respectively 2nd time derivative of vector $\overrightarrow{OA}$. Only when point O is the origin of a motionless CS $\vec{A} = \overrightarrow{OA}$ and $\vec{\vec{A}} = \overrightarrow{\vec{OA}}$ are valid.
- $||\overrightarrow{OA}||$: Length of vector $\overrightarrow{OA}$
- $X \cdot Y$: The dot operator is used for most product operations; the type of variables indicates the type of product operation. For example if $X$ or $Y$ is a scalar the dot indicates a normal multiplication, if $X$ is a tensor and $Y$ is a vector (or vice versa) the dot indicates a dot product (see Eqn. 10))
- $\overrightarrow{OA} \times \overrightarrow{OB}$: The $\times$ indicates the vector crossproduct.
Rearand front steering angles (angles between the rear and front wheels, and the vehicle’s length axis)

Rear and front velocity angles (angles between the actual velocity vectors at the rear and front point, and the vehicle’s length axis)

Mass of the vehicle and the rotational inertia of the vehicle at the mass centre point.

The cornering stiffness of the rear and front tire. This is an indication of the ability of the tire to create a lateral tire force with a certain slip angle.

2.2 Coordinate systems and vector basis

For the 2-dimensional case, the coordinate system (CS) can be described by a vector basis consisting of a set of 2 orthonormal basevectors. The superscript on the left indicates the index of the CS (in this case 0, which will be used as the index for the world CS):

\[ \mathbf{e}_0 = \begin{bmatrix} \mathbf{e}_0^1 \\ \mathbf{e}_0^2 \end{bmatrix} \quad \mathbf{e}_0 = \begin{bmatrix} \mathbf{e}_0^1 \\ \mathbf{e}_0^2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \]

An arbitrary vector can now be expressed by its components \( \mathbf{a} \) in a vector basis in combination with this vector basis:

\[ \mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \quad \mathbf{a} = \mathbf{e}_0^T \mathbf{a} \]

Which is equivalent to:

\[ \mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \quad \mathbf{a} = \mathbf{e}_0^T \mathbf{a} \]

The same can be done with any other CS, by simply changing the CS index. Of course in general the components will be different in each CS.

2.3 Example 1a: Finding the components in CS1 with known components in CS0, using matrices

Consider the vector basis CS0 and CS1, where CS1 is rotated over an angle \( \varphi \) around the origin relative to CS0. The components of vector \( \mathbf{a} \) are known in CS0. With this the components of this vector in CS1 must be found. This can be accomplished by rotating the vector over an angle \( -\varphi \), and calling the components of this new vector the components in CS1. The following 2x2 rotation matrix can be used to describe this rotation:

\[ \text{Rot}_{2x2}(-\varphi) = \begin{bmatrix} \cos \varphi & \sin \varphi \\ -\sin \varphi & \cos \varphi \end{bmatrix} \]

Multiplying this rotation matrix with the vector components \( \mathbf{a} \) results in \( \mathbf{a} \), the components of the vector in CS1. All this can be illustrated by the following:

![Figure 1: Example of 2D-rotation](image)
The components of \( \vec{a} \) in CS0, and the angle between CS0 and CS1 are:

\[
\begin{bmatrix}
0.3 \\
0.7
\end{bmatrix}, \quad \phi = \frac{\pi}{4}
\]

with these, the components of \( \vec{a} \) in CS1 can be calculated:

\[
\begin{bmatrix}
0.3 \\
0.7
\end{bmatrix} = \begin{bmatrix}
\frac{1}{2}\sqrt{2} & \frac{1}{2}\sqrt{2} \\
-\frac{1}{2}\sqrt{2} & \frac{1}{2}\sqrt{2}
\end{bmatrix} \begin{bmatrix}
0.3 \\
0.7
\end{bmatrix} = \begin{bmatrix}
0.707107 \\
0.282843
\end{bmatrix}
\]

Which can be verified to be the correct components of the vector in CS1.

2.4 Tensor properties

Transformations like Eqn. 4 can also be described with a rotation tensor. First some properties of (second order) tensors will be described.

A second order tensor is a linear transformation of a vector into another vector:

\[
B \cdot \vec{p} = \vec{q}
\]

\[
B \cdot (\alpha \cdot \vec{m} + \beta \cdot \vec{n}) = \alpha \cdot B \cdot \vec{m} + \beta \cdot B \cdot \vec{n}
\]

Two special tensors are the unity tensor and the zero tensor:

\[
I \cdot \vec{a} = \vec{a}
\]

\[
0 \cdot \vec{a} = 0
\]

A second order tensor is mathematically represented by a finite, nonunique, sum of dyads:

\[
B = \alpha_1 \cdot \vec{a}_1 \vec{b}_1 + \alpha_2 \cdot \vec{a}_2 \vec{b}_2 + \ldots
\]

Note: a third order tensor would exist of dyads of three vectors.

A dyad is a linear vector transformation, and works on a vector as follows:

\[
\begin{bmatrix}
\vec{a} \\
\vec{b}
\end{bmatrix} \cdot \vec{p} = \vec{a} \cdot \left( \vec{b} \cdot \vec{p} \right)
\]

\[
\begin{bmatrix}
\vec{p} \\
\vec{q}
\end{bmatrix} \cdot \begin{bmatrix}
\vec{a} \\
\vec{b}
\end{bmatrix} = \vec{b} \cdot \left( \vec{a} \cdot \vec{p} \right)
\]

From Eqn. 10 and Eqn. 11 it is clear that the order of the two vectors in a dyad is very important towards the result. Changing the order of the dyad vectors results in the conjugate dyad:

\[
\begin{bmatrix}
\vec{a} \\
\vec{b}
\end{bmatrix} = \vec{b} \vec{a} \neq \vec{a} \vec{b}
\]

Taking the conjugate of every dyad in a tensor results in a conjugate tensor, so Eqn. 9 would become:

\[
B^C = \alpha_1 \cdot \vec{a}_1 \vec{b}_1 + \alpha_2 \cdot \vec{a}_2 \vec{b}_2 + \ldots
\]

Where the conjugate tensor works on a vector as follows:

\[
B^C \cdot \vec{p} = \vec{p} \cdot B
\]

Suppose two orthonormal, Cartesian CS's with the same dimension exist. All combinations of the base vectors of these CS's can be used as dyads in Eqn. 9. For 2-dimensional CS's, this would result in the following four dyads:

\[
\begin{bmatrix}
\vec{e}_1 \\
\vec{e}_2
\end{bmatrix} = \begin{bmatrix}
\varepsilon_1 \\
\varepsilon_2
\end{bmatrix}, \quad \begin{bmatrix}
a_{11} & a_{12} \\
a_{21} & a_{22}
\end{bmatrix} \begin{bmatrix}
\vec{e}_1 \\
\vec{e}_2
\end{bmatrix} = \begin{bmatrix}
a_{11} \varepsilon_1 + a_{12} \varepsilon_2 \\
a_{21} \varepsilon_1 + a_{22} \varepsilon_2
\end{bmatrix} = \begin{bmatrix}
\vec{c}_1 \\
\vec{c}_2
\end{bmatrix}
\]

Eqn. 15 can be conveniently described with a matrix:

\[
\begin{bmatrix}
\vec{c}_1 \\
\vec{c}_2
\end{bmatrix} = \begin{bmatrix}
a_{11} & a_{12} \\
a_{21} & a_{22}
\end{bmatrix} \begin{bmatrix}
\vec{e}_1 \\
\vec{e}_2
\end{bmatrix}
\]

With appropriately chosen \( a_{ij} \) anything can be chosen for \( n \) and \( m \) without changing the tensor.

Therefore any single matrix representation of a tensor is only one of the many matrix representations for the same tensor. The index \( m \rightarrow n \) with the matrix indicates which CS's should be used in Eqn. 16. Also it indicates that by using this matrix representation, the result will be expressed in the vector basis \( n \), and vector basis \( m \) is the most convenient (but not required) vector basis to represent the vector the tensor is working on (e.g. when using the matrix representation of Eqn. 16, calculating \( \vec{q} = A \cdot \vec{p} \) will be most convenient when expressing \( \vec{p} \) relative to CSm; \( \vec{q} \) will be expressed in CSn).
Eqn. 17 can be used to calculate the corresponding matrix for a given tensor (Eqn. 16):
\[
\begin{bmatrix}
\hat{e}_1^m & \hat{e}_2^m \\
\hat{e}_1^n & \hat{e}_2^n
\end{bmatrix}
= A \cdot \begin{bmatrix}
\hat{e}_1^m & \hat{e}_2^m \\
\hat{e}_1^n & \hat{e}_2^n
\end{bmatrix}^T
\]
The inverse of the tensor in Eqn. 16 is given by:
\[
A^{-1} = \begin{bmatrix}
\hat{e}_1^m & \hat{e}_2^m \\
\hat{e}_1^n & \hat{e}_2^n
\end{bmatrix}^T (m \rightarrow n) A^{-1} , A^{-1} \cdot A = I
\]

It works on a vector as follows:
\[
A \cdot \bar{p} = \bar{q} \Rightarrow A^{-1} \cdot \bar{q} = \bar{p}
\]

2.5 Example 1b: Finding the components in CS1 with known components in CS0, using tensors

This example is the same as example 1, but this time tensors will be used. The vector components in CS1 of the vector \( \vec{a} \) (pointing to the point A) must be found, while the vector components in CS0 are known. Because this CS transformation does not change anything about the vector \( \vec{a} \), the tensor describing this transformation is actually the unity tensor.

Looking at Eqn. 16 it becomes clear that the matrix describing the unity tensor will be different for each choice of n and m. A convenient choice for n and m will be to take \( m=0 \) (for CS0) and \( n=1 \) (for CS1).

This way, when the tensor works on a vector described in CS0 the inner products of the base vectors can be easily calculated due to the orthonormality:
\[
\begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix}
\]

The vector that is the result of the tensor product (actually the same vector as the vector the tensor works on) will be conveniently described in CS1 with this choice for n and m, and the corresponding matrix is the same as the matrix in Eqn. 4.

This results in the following matrix representation of the unity tensor:
\[
I = \begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix}
\]

Expanding Eqn. 22 results in:
\[
\vec{a} = \bar{a} = \begin{bmatrix}
0 & \vec{e}_2 \cdot a \equiv \vec{e}_2 \cdot a
\end{bmatrix} = \begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix} \begin{bmatrix}
0 & \vec{e}_2 \cdot a \equiv \vec{e}_2 \cdot a
\end{bmatrix} = \begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix}
\]

When putting the known components in CS0 in Eqn. 23, and solving for the components in CS1 this results in the same vector in CS1 as in example 1. From Eqn. 23 it is clear that by using both tensors and the combination of vector basis and vector components it is easy to see what is happening during the calculations, because the vector basis is used in the calculations.

2.6 Time derivative of a vector

Problems arise when the time derivative of a vector is taken. The cause of these problems is the CS in which the vector is expressed. Intuition tells that to calculate a speed-vector, the time derivative of the position-vector should be taken. However, when the system consists of a rigid body moving in 2D space (e.g. in a circle, with constant angular speed), it can be convenient for some purposes to have a CS that is fixed to the body. Therefore in this CS the position-vector of any point on the body is a constant vector, resulting in zero for any order time derivative of this vector.

This is correct. Non-zero time derivatives in the body CS can only be interpreted as deformation of the body e.g. when the body would be stretched.
When an acceleration-sensor would be placed fixed to the body, this sensor can definitely sense acceleration during the body's movement. This seems contradictive to the fact that in the body CS no accelerations are present on the body itself.

When the position-vector of a point on the body is expressed in an inertial CS that does not move at all, taking the time derivative of the vector in this CS will result in the non-zero speed- and acceleration-vectors that would be measured on the body. This means that to take into account every acceleration and velocity of a moving body, an absolutely motionless CS would be needed. In theory this should be something like the point in the universe where all matter in an expanding universe is moving away from (if such a truly inertial point would actually exist). This is however far from practical, and for most applications it will suffice to chose a "motionless" CS fixed to either the earth's surface or to the earth's rotation axis (to include the forces due to the rotation). The movement of the earth around the sun and of the solar system etc. will be neglected this way, which is acceptable because of the small influence of these movements on the movement of a small object on earth. An example is the Coriolis force caused by earth's rotation. This force has a strong effect on large-scale systems (e.g. weather systems), but a smaller influence on something as small as a vehicle. It depends on the required precision and specific situation whether the Coriolis force (and thus the rotation of the earth) may be neglected.

From all this can be concluded that to take into account all forces working on an object one should take the time derivatives of a vector described in the inertial world CS, or use a moving CS while taking into account the time derivative of the transformation from the inertial world CS to this moving CS. For this purpose the use of tensors and vector basis prevents a lot of confusion about the correct way to take the time derivative of a vector.

In the following, \( \vec{a} \) is a vector describing a point \( A \) of a moving vehicle (only rotating around the origin), while CS1 is fixed to this vehicle. The first and second time derivatives can be taken in the inertial world CS (=CS0):

\[
\text{Eqn. 24} \quad \frac{d}{dt} \left( \begin{array}{c} 0 \\ \vec{a} \end{array} \right) = \left( \begin{array}{c} 0 \\ \dot{\vec{a}} \end{array} \right) = \dot{\vec{a}} = \ddot{\vec{a}} = \left( \begin{array}{c} 0 \\ \ddot{\vec{a}} \end{array} \right)
\]

\[
\text{Eqn. 25} \quad \frac{d}{dt} \left( \begin{array}{c} 0 \\ \vec{a} \end{array} \right) = \left( \begin{array}{c} 0 \\ \dot{\vec{a}} \end{array} \right) = \dot{\vec{a}} = \ddot{\vec{a}} = \left( \begin{array}{c} 0 \\ \ddot{\vec{a}} \end{array} \right)
\]

Note that the time derivative of the world vector basis is zero, because this vector basis is inertial. The same can be done in the vehicle CS (=CS1):

\[
\text{Eqn. 26} \quad \frac{d}{dt} \left( \begin{array}{c} 0 \\ \vec{a} \end{array} \right) = \left( \begin{array}{c} 0 \\ \dot{\vec{a}} \end{array} \right) = \dot{\vec{a}} = \ddot{\vec{a}} = \left( \begin{array}{c} 0 \\ \ddot{\vec{a}} \end{array} \right)
\]

\[
\text{Eqn. 27} \quad \frac{d}{dt} \left( \begin{array}{c} \dot{\vec{a}} \\ \vec{a} \end{array} \right) = \left( \begin{array}{c} \dot{\dot{\vec{a}}} \\ \ddot{\vec{a}} \end{array} \right) = \dot{\dot{\vec{a}}} = \dddot{\vec{a}} = \left( \begin{array}{c} \dot{\dot{\vec{a}}} \\ \dddot{\vec{a}} \end{array} \right)
\]

where the time derivative of the component column \( \left( \begin{array}{c} \dot{\vec{a}} \\ \vec{a} \end{array} \right) \) on the vehicle expressed in CS1 is zero, because the vehicle is presumed to be rigid.

Calculating Eqn. 24 and Eqn. 25 will be quite straightforward, when the component column in CS0 is known. Eqn. 26 and Eqn. 27 are not so easy. The problem is to find the time derivatives of the vector basis for CS1. For this, one can express the vector basis for CS1 as a function of the vector basis for CS0. Introducing a new tensor, which will be called a rotation tensor, can accomplish this. The rotation tensor rotates a vector over an angle. Note the difference with the tensor in Eqn. 21 (=the unity tensor) that does not change the vector; with the right matrix representation (a rotation matrix) it can however be used to find the vector components in CS1 when they are known in CS0.

The two matrix representations of the unity tensor that will be used to transform vector components from CS1 to CS0 and vice versa are:

\[
\text{Eqn. 28} \quad I = \left( \begin{array}{cc} 1 & 0 \\ \varepsilon & \varepsilon \end{array} \right) = \left( \begin{array}{cc} 1 & 0 \\ \varepsilon & \varepsilon \end{array} \right)^T
\]

\[
\text{Eqn. 29} \quad I^{-1} = I = \left( \begin{array}{cc} 1 & 0 \\ \varepsilon & \varepsilon \end{array} \right)^T \left( \begin{array}{cc} 1 & 0 \\ \varepsilon & \varepsilon \end{array} \right)^{-1}
\]

The newly introduced rotation tensor can be represented by a matrix:

\[
\text{Eqn. 30} \quad R = \left( \begin{array}{cc} 1 & 0 \\ \varepsilon & \varepsilon \end{array} \right) \left( \begin{array}{cc} 1 & 0 \\ \varepsilon & \varepsilon \end{array} \right)^T
\]
With the rotation tensor, the vector basis for CS1 can be expressed as follows:

Eqn. 31 \[ \begin{align*}
\frac{1}{2} e^T = R \cdot \frac{1}{2} e^T
\end{align*} \]

With Eqn. 31, the first time derivative of the vector basis for CS1 can now be written as:

Eqn. 32 \[ \begin{align*}
\frac{\partial}{\partial t} (R \cdot \frac{1}{2} e^T) = R \cdot \frac{1}{2} e^T + \frac{1}{2} R \cdot \frac{1}{2} e^T = \frac{1}{2} R \cdot R^{-1} \cdot \frac{1}{2} e^T
\end{align*} \]

The fact that any time derivative of the vector basis for CS0 equals zero, and the fact that a tensor working over its inverse working over a vector results in the untransformed vector are used in Eqn. 32.

In the same way, the second order time derivative becomes:

Eqn. 33 \[ \begin{align*}
\frac{\partial^2}{\partial t^2} (R \cdot \frac{1}{2} e^T) = \frac{1}{2} \frac{\partial}{\partial t} (R \cdot \frac{1}{2} e^T) = \frac{1}{2} R \cdot R^{-1} \cdot \frac{1}{2} e^T
\end{align*} \]

Using Eqn. 32 and Eqn. 33, Eqn. 26 and Eqn. 27 can be written as:

Eqn. 34 \[ \begin{align*}
\dot{a} = e \cdot a = R \cdot R^{-1} \cdot e^T \cdot a = \dot{R} \cdot R^{-1} \cdot e^T \cdot a
\end{align*} \]

Eqn. 35 \[ \begin{align*}
\ddot{a} = \frac{\partial}{\partial t} \dot{a} = e^T \cdot a = R \cdot R^{-1} \cdot e^T \cdot a
\end{align*} \]

When Eqn. 34 and Eqn. 35 are calculated (by using the matrix representation in Eqn. 30), it becomes clear that a number of innerproducts of the form \( e \cdot e^T \) must be calculated. While this results in the correct outcome, the result can be calculated more easily by adding an appropriately chosen unity tensor. The matrix representation will be \( e^T \left( \frac{1}{2} e^T \cdot e^T \right) e \), and therefore it contains exactly these innerproducts. This way these innerproducts need to be calculated only when the tensor is constructed instead of each time it is used. The chosen matrix representation of the unity tensor is actually the same as Eqn. 28:

Eqn. 36 \[ \begin{align*}
\frac{\partial}{\partial t} (e^T) = e \cdot e^T = \dot{R} \cdot R^{-1} \cdot e^T \cdot a = \dot{R} \cdot R^{-1} \cdot e^T
\end{align*} \]

Eqn. 37 \[ \begin{align*}
\frac{\partial}{\partial t} (e^T) = e \cdot e^T = \dot{R} \cdot R^{-1} \cdot e^T \cdot a = \dot{R} \cdot R^{-1} \cdot e^T
\end{align*} \]

The result is exactly the same as Eqn. 34 and Eqn. 35 (the extra tensor is after all the unity tensor). When the matrix representations from Eqn. 28 and Eqn. 30 are used, the time derivatives we now see will be expressed in CS0. If the vector components are needed in CS1, the unity tensor can be added again; but this time taking the matrix representation in Eqn. 29:

Eqn. 38 \[ \begin{align*}
\frac{\partial}{\partial t} (e^T) = e \cdot e^T = \dot{R} \cdot R^{-1} \cdot e^T \cdot a = \dot{R} \cdot R^{-1} \cdot e^T
\end{align*} \]

Eqn. 39 \[ \begin{align*}
\frac{\partial}{\partial t} (e^T) = e \cdot e^T = \dot{R} \cdot R^{-1} \cdot e^T \cdot a = \dot{R} \cdot R^{-1} \cdot e^T
\end{align*} \]

\( \dot{a} \) and \( \ddot{a} \) indicate not the first respectively second time derivative of the component column in CS1, but the first respectively second time derivative of the component column in CS0, which is then transformed to CS1. To come to the final results, the series of tensors in Eqn. 36 to Eqn. 39 can be grouped into one tensor:

Eqn. 40 \[ \begin{align*}
F = \frac{\partial}{\partial t} (e^T) = e \cdot e^T = \dot{R} \cdot R^{-1} \cdot e^T
\end{align*} \]

Eqn. 41 \[ \begin{align*}
s = \frac{\partial}{\partial t} (e^T) = e \cdot e^T = \dot{R} \cdot R^{-1} \cdot e^T
\end{align*} \]
2.7 Example 2: the time derivative of a vector in a rotating CS

To illustrate the correct calculation of the time derivatives, an example similar to example 1a and 1b will be used. With this example, the rotation angle is time-dependent (constant angular speed), and the components of vector $\vec{a}$ are constant in CS1. The time derivatives of $\vec{a}$ will be calculated.

![Figure 2: A time-dependent transformation. The components of vector $a$ are constant in CS1.](image)

The components of $\vec{a}$ in CS1 are:

$$\begin{bmatrix}
1 \hat{a}_1 \\
1 \hat{a}_2
\end{bmatrix}$$

The following tensors can be described now:

$$I = c(\omega t) \begin{bmatrix}
c(\omega t) & s(\omega t)
\end{bmatrix} = c(\omega t) \begin{bmatrix}
1 \hat{e}_1 & 0 \hat{e}_2
\end{bmatrix} = c(\omega t) \begin{bmatrix}
1 \hat{e}_1 & 0 \hat{e}_2
\end{bmatrix}$$

$$R = \begin{bmatrix}
c(\omega t) & -s(\omega t)
s(\omega t) & c(\omega t)
\end{bmatrix} = \begin{bmatrix}
1 \hat{e}_1 & 0 \hat{e}_2
0 \hat{e}_2 & 1 \hat{e}_1
\end{bmatrix}$$

$$\dot{R} = \begin{bmatrix}
\omega \cdot s(\omega t) & -\omega \cdot c(\omega t)
\omega \cdot c(\omega t) & -\omega \cdot s(\omega t)
\end{bmatrix}$$

$$R^{-1} = \begin{bmatrix}
c(\omega t) & s(\omega t)
-\omega \cdot s(\omega t) & -\omega \cdot c(\omega t)
\end{bmatrix} = \begin{bmatrix}
1 \hat{e}_1 & 0 \hat{e}_2
0 \hat{e}_2 & 1 \hat{e}_1
\end{bmatrix}$$

$$F = \dot{R} \cdot R^{-1} = \begin{bmatrix}
\omega \cdot s(\omega t) & -\omega \cdot c(\omega t)
\omega \cdot c(\omega t) & -\omega \cdot s(\omega t)
\end{bmatrix}$$

Now the first time derivative can be calculated and expressed relative to CS0 and CS1:

$$\dot{a} = F \cdot \dot{a} = F \cdot \begin{bmatrix}
\dot{a}_1 \\
\dot{a}_2
\end{bmatrix}$$

$$\Rightarrow \dot{a} = F \cdot \begin{bmatrix}
\dot{1} \hat{e}_1 \\
\dot{1} \hat{e}_2
\end{bmatrix} = \begin{bmatrix}
\dot{a}_1 \cdot \omega \cdot s(\omega t) & -\dot{a}_2 \\
\dot{a}_1 \cdot \omega \cdot c(\omega t) & \dot{a}_2 \cdot \omega \cdot s(\omega t)
\end{bmatrix} = \begin{bmatrix}
\dot{a}_1 \\
\dot{a}_2
\end{bmatrix}$$

$$\Rightarrow \dot{a} = F \cdot \begin{bmatrix}
\dot{1} \hat{e}_1 \\
\dot{1} \hat{e}_2
\end{bmatrix} = \begin{bmatrix}
\dot{a}_1 \cdot \omega \\
\dot{a}_2 \cdot \omega
\end{bmatrix} = \begin{bmatrix}
\dot{a}_1 \\
\dot{a}_2
\end{bmatrix}$$
\( \dot{\vec{a}} \) can (in this case) also be found directly by means of simple trigonometry. Taking the derivative of this will result in \( \dot{\vec{a}} \), which is the same as by using the tensor notation:

\[
\begin{align*}
\dot{\vec{a}} &= \begin{bmatrix} r \cdot c(\omega + \beta) \\ r \cdot s(\omega + \beta) \end{bmatrix}, \quad r = \sqrt{\left(\frac{a_1}{r}\right)^2 + \left(\frac{a_2}{r}\right)^2}, \quad \beta = \arccos\left(\frac{a_1}{r}\right) = \arcsin\left(\frac{a_2}{r}\right) \\
\dot{\vec{a}} &= \begin{bmatrix} -r \cdot \omega \cdot s(\omega + \beta) \\ r \cdot \omega \cdot c(\omega + \beta) \end{bmatrix} = -\begin{bmatrix} a_1 \cdot \omega \cdot s(\omega) \\ a_2 \cdot \omega \cdot c(\omega) \end{bmatrix} = \begin{bmatrix} -a_2 \cdot \omega \\ a_1 \cdot \omega \end{bmatrix},
\end{align*}
\]

Note that expressing \( \dot{\vec{a}} \) in CS1 results in \( \dot{\vec{a}} \cdot e_\perp = 1 \cdot 0 \cdot e_\perp = \dot{\vec{a}} \cdot e_\perp = 1 \cdot \frac{a_2}{a_1} \cdot \omega \), which is indeed equal to the \( \dot{\vec{a}} \cdot e_\perp \) as found with tensors.

### 2.8 The extended 3-DOF case

The 3-DOF in this extended case consist of a rotation over \( \phi \) and translations over \( 0C_1 \) and \( 0C_2 \). Although the movements described in this section are fully planar (2-dimensional) the space is taken to be 3-dimensional. This way the cross product of two vectors can be calculated without problems, in addition expansion to more than 3-DOF will be easier this way. The first two base vectors span the plane of movement. When translations are introduced between CS's, it is important to make a distinction between a vector and a point in space. In the previously discussed 1-DOF case (with only a rotation) this distinction is not necessary, due to the fact that the origin for every CS is the same. Therefore, in the 1-DOF case the length of a vector pointing to a point in space is the same in every CS, and the interpretation as a point or a vector can be interchanged at will.

In the 3-DOF case a translation is introduced which causes the possibility of a different origin for a different CS. The length and direction of the vector describing a point in space can therefore differ between CS's, making the vector interpretation and point interpretation not interchangeable. This can be shown in the following figure:

![Figure 3: Distinction between point and vector interpretation.](image)

With the point interpretation, \( \overrightarrow{CA} \) is transformed into \( \overrightarrow{OA} \). This is a different vector in space (different length and direction), but it is pointing to the same point in space (point A). With the vector interpretation, \( \overrightarrow{CA} \) is transformed into \( \overrightarrow{OB} \). This is the same vector in space (the same length and direction), but pointing to a different point.

When looking at application for a vehicle model, the point interpretation should be used for CS transformations of a point on the moving vehicle. When the speed and acceleration of a point on the vehicle must be calculated, the vector interpretation must be used to transform the speed and acceleration vectors to another CS; only the magnitude and the direction of the speed and acceleration are of interest, not the point where these vectors are pointing to.
The points $A$, $B$ and $C$ in Figure 3 can be indicated by vectors as follows:

Eqn. 44 $A = \overrightarrow{CA} = e^T \cdot A$

Eqn. 45 $B = \overrightarrow{OB} = e^T \cdot B$

Eqn. 46 $A = \overrightarrow{OA} = e^T \cdot A = \overrightarrow{CA} + \overrightarrow{OC} = \overrightarrow{OB} + \overrightarrow{OC}$

Eqn. 47 $C = \overrightarrow{OC} = e^T \cdot C$

Eqn. 44 and Eqn. 45 actually describe the same vector (i.e. same direction and length), they only work from another origin.

Similar to the 1-DOF case, a matrix representation of the unity tensor $I$ can be used to find the components in CS0, when the components in CS1 are known. This tensor only takes incorporates the rotation between CS0 and CS1. The translation is not included in this tensor.

Eqn. 48 $R = \text{Rot}(\vec{e}_3) = \begin{bmatrix} c\varphi & -s\varphi & 0 \\ s\varphi & c\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Eqn. 49 $I = e^T \cdot R = e^T = c\varphi \cdot e^T \cdot e_1 + s\varphi \cdot e^T \cdot e_2 + s\varphi \cdot e^T \cdot e_2 + c\varphi \cdot e^T \cdot e_1 + c\varphi \cdot e^T \cdot e_1 + e^T \cdot e_3$

The same can be done vice versa (from CS0 to CS1) this results in the following matrix representation for the unity tensor. To make the difference between the two matrix representations more clear, this tensor is called $I^{-1}$ instead of $I$ although it is still the unity tensor:

Eqn. 50 $R = R^{-1} = \text{Rot}(\vec{e}_3, -\varphi) = \begin{bmatrix} c\varphi & s\varphi & 0 \\ -s\varphi & c\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Eqn. 51 $I^{-1} = e^T \cdot R = e^T = c\varphi \cdot e^T \cdot e_1 + s\varphi \cdot e^T \cdot e_2 + s\varphi \cdot e^T \cdot e_2 + c\varphi \cdot e^T \cdot e_1 + c\varphi \cdot e^T \cdot e_1 + e^T \cdot e_3$

With these two matrix representations of the unity tensors, the components of vectors in one CS can be found, when they are known in the other CS1, using vector interpretation:

Eqn. 52 $\overrightarrow{OB} = I \cdot \overrightarrow{CA}$

Eqn. 53 $\overrightarrow{CA} = I^{-1} \cdot \overrightarrow{OB}$

Introducing the translation will be done by adding the translation vector ($\overrightarrow{OC}$ in Figure 3), pointing from the origin of CS0 (point O) to the origin of CS1 (point C). This is already shown in Eqn. 46. Of course the translation $\overrightarrow{OC}$ should be known in this case.

The transformations with point interpretation will become (from CS1 to CS0):

Eqn. 54 $A = \overrightarrow{OA} = e^T \cdot A = \overrightarrow{CA} + \overrightarrow{OC} = \overrightarrow{OB} + \overrightarrow{OC} = I \cdot \overrightarrow{CA} + \overrightarrow{OC}$

In matrix representation this would become:

Eqn. 55 $A = \overrightarrow{OA} = e^T \cdot A = I \cdot \left( e^T \cdot A \right) = e^T \cdot \left( I^{-1} \right) \cdot \overrightarrow{CA} + \overrightarrow{OC}$

From CS0 to CS1 it will be:

Eqn. 56 $A = \overrightarrow{CA} = e^T \cdot A = \overrightarrow{OA} - \overrightarrow{OC} = \overrightarrow{OB} = I^{-1} \cdot \left( \overrightarrow{OA} - \overrightarrow{OC} \right) = I^{-1} \cdot \overrightarrow{OB}$

And in matrix representation:

Eqn. 57 $A = \overrightarrow{CA} = e^T \cdot A = e^T \cdot \left[ e^T \cdot \overrightarrow{OA} - \overrightarrow{OC} \right]$

For correct calculation of the time derivatives, a transformation tensor must be found to express CS1 in CS0. This tensor will only incorporate the rotation between CS1 and CS0, because the translation between CS1 and CS0 does not contribute to the length and direction of the base vectors from CS0 and CS1; it only contributes to the position of the origin.
Therefore, similar to the 1-DOF case, the rotation tensor will be:

\[ R^{0} = \mathbf{e}_{0} \cdot \mathbf{e}_{0} \cdot \phi_{0} \]

\[ R^{0} = \mathbf{e}_{0} \cdot \mathbf{e}_{0} \cdot \phi_{0} \]

CS1 can now be expressed in CS0:

\[ e^{T} \cdot R^{0} \cdot e^{T} = \begin{bmatrix} e_{1}^{T} \\ e_{2}^{T} \\ e_{3}^{T} \end{bmatrix} = \begin{bmatrix} \phi_{0}^{0} \cdot e_{1} + s \phi_{0}^{0} \cdot e_{2} \\ \phi_{0}^{0} \cdot e_{2} - s \phi_{0}^{0} \cdot e_{1} \\ 0 \end{bmatrix} \]

Again, like the 1-DOF case, other useful tensors can be derived from the previous ones:

The first time derivative of \( R \) will be:

\[ \dot{R}^{0} = \mathbf{e}_{0} \cdot \mathbf{e}_{0} \cdot \phi_{0} \]

The second time derivative of \( R \) will be:

\[ \ddot{R}^{0} = \mathbf{e}_{0} \cdot \mathbf{e}_{0} \cdot \phi_{0} \]

Because in the 3-DOF case there is a distinction between a point and a vector, there is also a distinction between the time derivative of a point and a vector. For application in a vehicle model, it is usually interesting to know the (second) time derivative of a point (on the vehicle) rather than of a vector. For the general time derivatives calculated hereafter, \( \phi_{0}^{0} \mathbf{C}_{0} \), and \( \phi_{0}^{0} \mathbf{C}_{1} \), are all assumed to be some function of time. Furthermore, \( \mathbf{A}^{0} \) is not restricted to a constant. This way a more widely applicable derivative will be found.

Using point interpretation, the \( n \)-th order derivative of point \( \mathbf{A} \) can be expressed as follows:

\[ \frac{\partial^{n}}{\partial t^{n}} (\mathbf{A}) = \frac{\partial^{n}}{\partial t^{n}} (\mathbf{O} \cdot \mathbf{A} + \mathbf{O} \cdot \mathbf{C}) = \frac{\partial^{n}}{\partial t^{n}} (\mathbf{C} \cdot \mathbf{A} + \mathbf{O} \cdot \mathbf{C}) \]

Just like in the 1-DOF case, \( \mathbf{e}_{0}^{T} \) is zero because CS0 is considered a motionless CS.

The first time derivative of point \( \mathbf{A} \) will become:

\[ \dot{\mathbf{A}} = \frac{\partial}{\partial t} (\mathbf{O} \cdot \mathbf{A}) = \mathbf{e}^{T} \cdot \dot{\mathbf{A}} \]

\[ \dot{\mathbf{A}} = \frac{\partial}{\partial t} (\mathbf{O} \cdot \mathbf{A}) = \mathbf{e}^{T} \cdot \dot{\mathbf{A}} + \mathbf{e}^{T} \cdot \dot{\mathbf{C}} \]

The second time derivatives of point \( \mathbf{A} \) can be expressed as:

\[ \ddot{\mathbf{A}} = \frac{\partial^{2}}{\partial t^{2}} (\mathbf{O} \cdot \mathbf{A}) = \mathbf{e}^{T} \cdot \ddot{\mathbf{A}} + \mathbf{e}^{T} \cdot \ddot{\mathbf{C}} \]

\[ \ddot{\mathbf{A}} = \frac{\partial^{2}}{\partial t^{2}} (\mathbf{O} \cdot \mathbf{A}) = \mathbf{e}^{T} \cdot \ddot{\mathbf{A}} + \mathbf{e}^{T} \cdot \ddot{\mathbf{C}} \]

\[ \dddot{\mathbf{A}} = \frac{\partial^{3}}{\partial t^{3}} (\mathbf{O} \cdot \mathbf{A}) = \mathbf{e}^{T} \cdot \dddot{\mathbf{A}} + \mathbf{e}^{T} \cdot \dddot{\mathbf{C}} \]
The previous four expressions can now be written using tensor notation. Similar to the 1-DOF case, the combined first and second order time derivative tensors will be used for this purpose. The combined first order time derivative tensor (with two different matrix representations) will be:

\[ \begin{align*}
\text{Eqn. 72} &= \begin{bmatrix}
-\phi \cdot s\varphi & -\phi \cdot c\varphi & 0 \\
\phi \cdot c\varphi & -\phi \cdot s\varphi & 0 \\
0 & 0 & 0
\end{bmatrix} \\
\text{Eqn. 73} &= \begin{bmatrix}
0 & -\phi & 0 \\
\phi & 0 & 0 \\
0 & 0 & 0
\end{bmatrix}
\end{align*} \]

The combined second order time derivative tensor will become:

\[ \begin{align*}
\text{Eqn. 74} &= \dot{R} \cdot R^{-1} = \ddot{R} \cdot R^{-1} \cdot \dot{I} = I^{-1} \cdot \ddot{R} \cdot R^{-1} \cdot \dot{I} = e^{-T}_{1-0} e^{1}_{0-1} = e^{T}_{1-0} e^{1}_{1-0}
\end{align*} \]

The time derivatives of the \( e \) can be written as:

\[ \begin{align*}
\text{Eqn. 75} &= \frac{1}{2} T_{e} F \cdot e = \frac{1}{2} T_{e} S \cdot e \\
\text{Eqn. 76} &= \frac{1}{2} T_{e} \end{align*} \]

With these, the general form of \( \text{Eqn. 69} \) and \( \text{Eqn. 71} \) will become:

\[ \begin{align*}
\text{Eqn. 80} &= \frac{1}{2} A = \frac{1}{2} T \cdot \dot{e} + \frac{1}{2} e \cdot \dot{C} \\
\text{Eqn. 81} &= \frac{1}{2} e \cdot \frac{1}{2} A + \frac{1}{2} e \cdot \frac{1}{2} e = \frac{1}{2} T \cdot \dot{e} + \frac{1}{2} e \cdot \dot{C}
\end{align*} \]

These general forms can be simplified, when \( A \) is constant (which will be the case in the vehicle model constructed in this report):

\[ \begin{align*}
\text{Eqn. 82} &= \frac{1}{2} A = \frac{1}{2} T \cdot \dot{e} + \frac{1}{2} e \cdot \dot{C} \\
\text{Eqn. 83} &= \frac{1}{2} e \cdot \frac{1}{2} A + \frac{1}{2} e \cdot \frac{1}{2} e = \frac{1}{2} T \cdot \dot{e} + \frac{1}{2} e \cdot \dot{C}
\end{align*} \]

Depending on which of the matrix representations from \( \text{Eqn. 71} \) to \( \text{Eqn. 76} \) is used for \( F \) and \( S \), the result of the first term (containing \( F \) or \( S \) ) of \( \text{Eqn. 82} \) and \( \text{Eqn. 83} \) can be expressed in either CS0 or CS1. When the total result is needed in CS1, the last term (containing \( C \) or \( 0 \cdot C \) ) should be multiplied with the unity tensor (with the proper matrix representation) as is shown in the last part of \( \text{Eqn. 82} \) and \( \text{Eqn. 83} \) can be expressed as:

\[ \begin{align*}
\text{Eqn. 84} &= \frac{1}{2} T \cdot \dot{e} + \frac{1}{2} e \cdot \dot{C} \\
\text{Eqn. 85} &= \frac{1}{2} e \cdot \frac{1}{2} A + \frac{1}{2} e \cdot \frac{1}{2} e = \frac{1}{2} T \cdot \dot{e} + \frac{1}{2} e \cdot \dot{C}
\end{align*} \]
Just like in Eqn. 38 and Eqn. 39, the component columns $\hat{1}_{\hat{A}}$ and $\hat{2}_{\hat{A}}$ are not the first respectively second time derivative of $\hat{1}_{\hat{A}}$ (these would be zero for constant $\hat{1}_{\hat{A}}$); the $\hat{}$ symbol is added to indicate this difference.

The newly introduced component columns $\hat{1}_{\hat{A}}$ and $\hat{2}_{\hat{A}}$ describe the velocity respectively acceleration vectors of point A relative to CS0 (which is presumed to be inertial), but expressed in CS1 using the vector interpretation.

2.9 Example 3: Calculating the Coriolis force on earth

Finally, a more practical example is worked out. In this example the acceleration vector on a rigid body moving in a north-south direction over the surface of our planet will be calculated for two cases. First the rotation of the earth around the north-south axis will be ignored. In the second case this rotation will be taken into account, resulting in an acceleration due to the Coriolis force.

![Figure 4: CS2 and CS1 with a stationary earth.](image)

Figure 4 describes the first situation. A rigid body is moving over the earth’s surface from pole to pole. For simplicity, the earth is modelled as a perfectly smooth sphere with the same radius as the earth. The shape of the rigid body (i.e. all the points of the body) can be described conveniently in the body CS (CS2), which has its origin (O2) on the surface of the earth and the direction of the base vectors are as shown in Figure 4. Because CS2 is fixed relative to the body, a general point on the body (point P) can be described with:

$$P = \overrightarrow{O_2}P = \begin{bmatrix} 2P_1 \\ 2P_2 \\ 2P_3 \end{bmatrix}$$

The movement of O2 along the surface can be described in the world CS (CS1). Initially it will be assumed that the earth is not rotating or moving in any way. Therefore CS1 is assumed to be a motionless CS. O2 can now be described in CS2 and CS1 as follows:

$$O_2 = \overrightarrow{O_2}O_2 = 2\hat{e}_T \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$O_2 = \overrightarrow{O_1}O_2 = 2\hat{e}_T \cdot \begin{bmatrix} 0 \\ R_{earth} \cdot \cos(\varphi) \\ R_{earth} \cdot \sin(\varphi) \end{bmatrix}$$

$$\varphi = \omega_1 \cdot t + \varphi_0 = \frac{V}{R_{earth}} \cdot t + \varphi_0$$

When an acceleration sensor would be placed in point P, the measured acceleration vector in P can be calculated by using Eqn. 83:

$$\ddot{P} = \overrightarrow{S} \cdot \overrightarrow{O_2}P + \overrightarrow{O_1}O_2$$
In this case the $S$ tensor will be called $S_1$ to indicate the fact that it based on the assumption that CS1 is a motionless CS.

One matrix representation of tensor $S_1$ is given below. The relationship between $R_A$ and $I_A$ is the same as in Chapter 2:

\[
S_1 = I_A^{-1} \cdot \tilde{R}_A \cdot R_A^{-1} \cdot I_A = 2 \tilde{e}^T \left( \begin{array}{c}
1 & \omega_x & \omega_y & \omega_z \\
\omega_x & 1 & 0 & 0 \\
\omega_y & 0 & 1 & 0 \\
\omega_z & 0 & 0 & 1
\end{array} \right) \tilde{e} = 2 \tilde{e}^T 
\]

Now that a matrix representation for tensor $S_1$ is chosen, the acceleration of point P can actually be calculated:

\[
\ddot{P} = 2 \tilde{e}^T \left[ 0 \quad - \frac{V^2 (2P_3 + R_{\text{earth}})}{R_{\text{earth}}^2} \quad - \frac{V^2 P_3}{R_{\text{earth}}^2} \right] \tilde{e}
\]

This means a small acceleration vector pointing to the centre of the earth is needed to keep the body on the prescribed path from pole to pole. A numerical example gives an indication of the magnitude of this vector, in which $\varphi_0$ is approximately equal to the latitude of the Netherlands ($=52^\circ$).

\[
V = 20 \text{ m/s} \quad R_{\text{earth}} = 6.378 \times 10^6 \text{ m} \quad \frac{\dot{V}^2 P_3}{R_{\text{earth}}^2} = 0 \quad \varphi_0 = 52^\circ
\]

\[
\Rightarrow \ddot{P} = 2 \tilde{e}^T \left[ 0 \quad -6.27 \times 10^{-5} \quad -9.83 \times 10^{-12} \right] \tilde{e}
\]

However this result is only valid when the assumption of a motionless CS1 holds.

For the second case, a second world CS is introduced (CS0) which takes into account the earth's rotation. With this CS a more accurate approximation for the measured acceleration vector can be calculated.

\[
\text{Figure 5: The rotation between CS0 and CS1}
\]

The rotation angle ($\gamma$) around the pole-to-pole axis of the earth is given by:

\[
\gamma = \omega \cdot t = \frac{2\pi \cdot t}{24 \times 3600}
\]

The rotation between CS0 and CS1 is actually the same as described in example 1b. Therefore point $O_2$ can be easily described in CS0 by:

\[
O_2 = O_0 O_1 = I \cdot O_1 O_2
\]

Point $O_2$ will now follow a path as shown in Figure 6 (from a observer at a fixed position in CS0).
Figure 6: The path of the rigid body, seen from an observer at a fixed position in CS0.

To calculate the acceleration in point \(P\), the same equation can be used as earlier (substituting CS1 by CS0):

\[
\overrightarrow{P} = \overrightarrow{S} \cdot \overrightarrow{O_2 P} + \overrightarrow{O_0 O_2}
\]

But now the \(S\) tensor will also include the rotation of the earth, and will be called \(S_0\). The relationship between \(R_B\) respectively \(R_C\) and \(I_B\) respectively \(I_C\) is the same as in Chapter 2.

\[
S_0 = I_C^{-1} \cdot \dot{R}_C \cdot R_C^{-1} \cdot I_C = \begin{bmatrix} \dot{e}_x & \dot{e}_y & \dot{e}_z \end{bmatrix} = \begin{bmatrix} -\omega_x^2 & \frac{2V \cdot \omega_z \cdot \ell_0}{R_0} & \frac{2V \cdot \omega_z \cdot \ell_0}{R_0} \\ 0 & \omega_y^2 & \frac{2V \cdot \omega_z \cdot \ell_0}{R_0} \\ 0 & \frac{2V \cdot \omega_z \cdot \ell_0}{R_0} & \frac{2V \cdot \omega_z \cdot \ell_0}{R_0} \end{bmatrix}
\]

\[
R_R = R_A \cdot R_B = \begin{bmatrix} e_x & e_y & e_z \end{bmatrix} \begin{bmatrix} c\gamma & -s\gamma & 0 \\ s\gamma & c\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c\gamma & -s\gamma \cdot c\varphi & s\gamma \cdot s\varphi \\ s\gamma \cdot c\varphi & c\gamma & -s\gamma \cdot s\varphi \\ 0 & s\varphi & c\varphi \end{bmatrix} \begin{bmatrix} e_x \\ e_y \\ e_z \end{bmatrix} = \begin{bmatrix} \dot{e}_x \\ \dot{e}_y \\ \dot{e}_z \end{bmatrix} = \begin{bmatrix} \dot{e}_x \\ \dot{e}_y \\ \dot{e}_z \end{bmatrix}
\]

\[
R_C = R_A \cdot \dot{R}_B = \begin{bmatrix} \dot{e}_x \\ \dot{e}_y \\ \dot{e}_z \end{bmatrix} = \begin{bmatrix} c\gamma \cdot c\varphi & s\gamma \cdot s\varphi \\ -s\gamma \cdot c\varphi & c\gamma \cdot s\varphi \\ 0 & c\varphi \end{bmatrix} \begin{bmatrix} \dot{e}_x \\ \dot{e}_y \\ \dot{e}_z \end{bmatrix} = \begin{bmatrix} \dot{e}_x \\ \dot{e}_y \\ \dot{e}_z \end{bmatrix}
\]

When the acceleration vector in CS2 is calculated again with

\[
V = 20 m/s \quad R_0 = 6.378 \cdot 10^6 m \quad \ell = \begin{bmatrix} 0 & 1 \end{bmatrix} \quad \varphi_0 = 52^\circ
\]

The result will be time-dependent as shown in Figure 7. At \(t=0\) the rigid-body is approximately at the same latitude as the Netherlands (\(=52^\circ\)).
Figure 7: The components of the acceleration vector at point P: \( \mathbf{e}_1 \) (A for \( V=20\text{m/s}, \phi=0^\circ \)), \( \mathbf{e}_2 \) (B for \( V=20\text{m/s}, B_0 V=0\text{m/s} \)) and \( \mathbf{e}_3 \) (C for \( V=20\text{m/s}, C_0 V=0\text{m/s} \)). At \( t=0 \) the angle \( \phi \) equals 52°.

From the previous it is obvious that the choice that CS is assumed to be motionless should not be taken lightly. Depending on the required accuracy and the expected mobility of the object on which the acceleration is to be measured, the rotation of the earth can have a significant influence on the accuracy. For example: for high accuracy acceleration measurement on a object fixed to the earth's surface CS1 can be assumed motionless, but the measurements should be compensated with the pre-calculated constant values from Figure 7.

2.10 Main conclusions about the use of tensors

Using tensors in the field of CS transformations can be very useful, for the following reasons:

- Because tensors operate independent of CS's, the choice of the most convenient CS's and matrix representations can be delayed to the very end of the calculation.
- When finally using the matrix representation of the tensors, the influence of the CS's is more obvious in the calculations with tensors. What is happening during transformations can be clearly seen this way.
- Tensor notation is very compact. Only when finally using the matrix representation of the tensors, matrices and a number of indices become necessary.

The only real disadvantage of using tensors is that it takes time to get used to this way of describing transformations.

To sum it up, using tensors will make the description CS transformations compact and, it can make interpretation of these transformations clearer and more straightforward.
Chapter 3: Analysis of a “modular” planar vehicle model

The first implementation of a simple vehicle model that was developed will not be described here; but while discussing this early model, it turned out that the description of the CS transformations in this model was not very clear (a problem that was already encountered in the HOV project). This led to the tensor description introduced in the previous chapter. After the tensor notation clarified a lot about using CS transformations, work on the vehicle model was continued by constructing a new model based on a combination of the early model, and the knowledge gained in the previous chapter.

To gain a better understanding of vehicle modelling, a full vehicle model was split into separate “modules”. The method of splitting the model into separate modules actually originated from the way I personally learned to understand a general vehicle model. I started with the most basic vehicle model: the kinematic model. After fully understanding and describing this model, two functional blocks (describing the forces working on the vehicle, and introducing the tire characteristics) were added to get a more accurate model. This way of describing a vehicle was then expanded by adding a number of modules (e.g. the inverse modules of those already described).

The full model is based on a so-called planar bicycle model as shown in Figure 8. \( \delta_R \) and \( \delta_F \) are the steering angles (i.e. the angles the rear and front wheels make with the length axis). \( \phi_R \) and \( \phi_F \) are the velocity angles (i.e. the angles of the actual velocity vectors at the rear and front points). All four of these angles are assumed to be in the range \([-\pi, \pi]\). For the steering angles, this is a practical assumption due to the mechanical limits of the steering angles. The velocity angles will automatically stay within the same range as the steering angles. This assumption makes a number of calculations easier, especially when solving the steering angles in the inverse tire force module (see paragraph 3.7). This introduces the problem that some modules will not operate properly while driving in reverse. This should not be a very big problem, because most of the time a forward movement will be the interesting situation. When reverse movement is specifically needed, the model can be easily adapted to represent a pure reverse movement (basically by mirroring the vehicle). Q is the steering pole (i.e. momentary point around which the vehicle momentarily describes a circular trajectory). A full path can be imagined as a series of connected circle segments. For a linear movement (i.e. \( \phi_R = \phi_F \) ) Q cannot be calculated. In some modules, an extra equation is needed to solve a variable properly; in these cases the vehicle is assumed to be rear-wheel driven.

\[ \text{Figure 8: A picture of the bicycle model used.} \]

3.1 The kinematic module

A tireless model (\( \delta_R = \phi_R \) and \( \delta_F = \phi_F \)) was used to create the first of the three module-blocks, describing the kinematic movement. This module’s inputs are the angles \( \phi_R \) and \( \phi_F \) of the front and rear velocity vectors and the angular velocity vector \( \omega \) around Q (see Figure 8). The distances \( a_R \)
and \( \alpha_F \) are also needed in this module. An inertial coordinate system (CS) that does not move at all is assumed to be present. This CS (CS0) will be called the "world", or world CS; its origin is called \( O \). To make the "world" easily imaginable, it may be seen as the earth's surface. However, keep in mind that by doing so a number of accelerating forces (e.g. the Coriolis force) will be neglected. One can imagine a second CS on the vehicle, which is fixed relative to the vehicle. This CS will be called CS1; its origin is called \( C \) (the vehicle's mass centre point). At \( t=0 \), CS0 and CS1 are assumed to be the same.

![Figure 9: The world- and vehicle coordinate systems.](image)

The essential output of the kinematic module are \( \Omega \) and \( \bar{\omega} \) (the latter is also an input, but for the sake of symmetry with the inverse kinematic module, it is also considered an output). The path of any point that is at a fixed position relative to CS1 can easily be calculated in this module as well.

First the component column of the rotation point \( Q \) (see Figure 8) is most easily calculated in CS1. The points \( C, R \) and \( F \) can be described in CS1 as follows:

\[
\text{Eqn. 86} \quad C = CC = \mathbf{e}_1^T \cdot C = \begin{bmatrix} C_1 \\ C_2 \\ C_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} \cdot 0
\]

\[
\text{Eqn. 87} \quad R = CR = \mathbf{e}_1^T \cdot R = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} \cdot \begin{bmatrix} -\alpha_R \\ 0 \end{bmatrix}
\]

\[
\text{Eqn. 88} \quad F = CF = \mathbf{e}_1^T \cdot F = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} \cdot \begin{bmatrix} \alpha_F \\ 0 \end{bmatrix}
\]

To calculate the point \( Q \) (in CS1), the intersection of the two lines, through \( R \) and \( F \), perpendicular to the velocity vectors at \( R \) and \( F \) (specified by \( \varphi_R \) and \( \varphi_F \)) must be found. Using basic trigonometry the following can be found:

\[
\text{Eqn. 89} \quad Q = CQ = \mathbf{e}_1^T \cdot Q = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} \cdot \begin{bmatrix} \frac{\tan(\varphi_R) - \tan(\varphi_F)}{\tan(\varphi_R) - \tan(\varphi_F)} \\ \frac{\tan(\varphi_R) - \tan(\varphi_F)}{\tan(\varphi_R) - \tan(\varphi_F)} \\ 0 \end{bmatrix}
\]
To calculate the path of a random point fixed to the vehicle, a new point \( N \) is introduced:

\[
N = \overline{CN} = \overline{CQ} - \overline{CN} = \ell N = \ell Q - \ell N_1, \quad N_1, \quad N_3
\]

Finally a new vector indicating the direction and distance to \( N \) relative to \( Q \) is introduced:

\[
\overline{QN} = \overline{CN} - \overline{CQ}
\]

With Eqn. 87 to Eqn. 91 and \( \overrightarrow{\omega} \); it is possible to calculate the velocity vector at any point on the vehicle (i.e., the general point \( N \)). This is because the angular velocity of any point on the vehicle is the same. The general vector equation for the relation between angular velocity, tangential speed and position is given by:

\[
\overrightarrow{v} = \overrightarrow{\omega} \times \overrightarrow{r}
\]

For point \( N \) this becomes:

\[
\overrightarrow{N} = \overrightarrow{\omega} \times \overrightarrow{QN}
\]

In the case of a planar motion in the plane spanned by \( \overrightarrow{e}_1 \) and \( \overrightarrow{e}_2 \), \( \overrightarrow{\omega} \) will have a zero first and second component. The component column of \( \overrightarrow{\omega} \) will be the same in every CS for which the third base vector is equal to that of CS0 or CS1:

\[
\overrightarrow{\omega} = \begin{bmatrix} \omega_1 \\ 0 \\ 0 \end{bmatrix}
\]

The magnitude of \( \overrightarrow{N} \) and the angle it makes with \( \overrightarrow{e}_1 \) can now be calculated:

\[
\|\overrightarrow{N}\| = \|\overrightarrow{QN}\| = \|\overrightarrow{Q}\| = \|\overrightarrow{QN}\|
\]

\[
\phi_N = \arctan \left( \frac{\omega \cdot (\overrightarrow{N}_1 - \overrightarrow{Q}_1)}{\omega \cdot (\overrightarrow{N}_2 - \overrightarrow{Q}_2)} \right) = \arctan \left( \frac{\overrightarrow{N}_1 - \overrightarrow{Q}_1}{\overrightarrow{N}_2 - \overrightarrow{Q}_2} \right)
\]

Eqn. 96 is only correct when \( \Phi_N \) is in the range of \([-\pi, \pi]\]. If reverse driving were to be introduced, the full \([-\pi, \pi]\) range is needed, which could be achieved with a 2-argument \text{acrtan2} function.

For point \( C \) Eqn. 93 becomes:

\[
\overrightarrow{N} = \overrightarrow{Q} \times \overrightarrow{QN} = \overrightarrow{Q} \times \overrightarrow{QN} = \overrightarrow{Q}
\]

To transform a point in space from CS1 to CS0 (using the point interpretation) the tensor given in Eqn. 54 and Eqn. 55 can be used. Now both the translation and the rotation must be found. The rotation (\( \Phi \) in Eqn. 48 and further) will be called \( \rho \) to avoid confusion with the angles of the speed vectors.

Because CS0 and CS1 start out as equal, \( \rho \) can be found by integrating the angular velocity \( \omega \) and indicates the orientation of the vehicle:

\[
\rho = \int_0^1 \omega \cdot \omega \, d\tau
\]

With this the two matrix representations of the unity tensor can be constructed. First, the one to be used for transformation from CS1 to CS0:

\[
\text{Eqn.} \quad R = \text{Rot}(\alpha, \rho) = \begin{bmatrix} \alpha \rho - sp \rho \omega \rho \end{bmatrix}
\]

\[
\text{Eqn.} \quad R = \text{Rot}(\alpha, \rho) = \begin{bmatrix} \alpha \rho - sp \rho \omega \rho \end{bmatrix}
\]

\[
\text{Eqn.} \quad R = \text{Rot}(\alpha, \rho) = \begin{bmatrix} \alpha \rho - sp \rho \omega \rho \end{bmatrix}
\]

\[
\text{Eqn.} \quad R = \text{Rot}(\alpha, \rho) = \begin{bmatrix} \alpha \rho - sp \rho \omega \rho \end{bmatrix}
\]
Second, the one to be used for transformation from CS0 to CS1:

\[ \mathbf{R} = \text{Rot}(0e_3, -\rho) = \begin{bmatrix}
    c\rho & s\rho & 0 \\
    -s\rho & c\rho & 0 \\
    0 & 0 & 1
\end{bmatrix} \]

\[ I^{-1} = \begin{bmatrix}
    e_1^T \\
    e_2^T \\
    e_3^T
\end{bmatrix}_0 = c\rho \begin{bmatrix}
    e_1^T \\
    e_2^T \\
    e_3^T
\end{bmatrix} + s\rho \begin{bmatrix}
    -e_2^T \\
    e_1^T \\
    0
\end{bmatrix} - s\rho \begin{bmatrix}
    e_2^T \\
    -e_1^T \\
    0
\end{bmatrix} + c\rho \begin{bmatrix}
    e_2^T \\
    e_1^T \\
    0
\end{bmatrix} + l_4^T + l_3^T + l_2^T + l_1^T
\]

To find the translation, \( \overrightarrow{C} \) can be expressed in CS0 (with the tensors in Eqn. 99 to Eqn. 102) and then integrated, to get point \( C \) expressed in CS0.

The translations can be calculated with the following equations:

\[ 0 C_1 = C_1 \cdot c\rho - l_3^T \cdot C_2 \cdot s\rho \cdot d\tau = \omega \cdot \int_0^1 Q_2 \cdot c\rho \cdot Q_1 \cdot s\rho \cdot d\tau \]

\[ 0 C_2 = C_2 \cdot s\rho + C_2 \cdot c\rho \cdot d\tau = \omega \cdot \int_0^1 Q_2 \cdot s\rho^{-1} Q_1 \cdot c\rho \cdot d\tau \]

Now that the rotation and translation are defined the path in CS0 can be obtained of any point that is fixed relative to CS1 and vice versa (using Eqn. 54 or Eqn. 57). As was shown in the previous chapter (Eqn. 58 and further) the time derivatives can now be calculated both in CS0 and CS1, by replacing \( \varphi \) by \( \rho \). With these, the general form of the second order time derivative of point \( N \) would be:

\[ N = \frac{d^2}{dt^2} \]

\[ C = S \cdot CN + OC = S \cdot CN + I^{-1} \cdot OC \]

And in matrix representation:

\[ \begin{bmatrix}
    0 C_1 \\
    0 C_2 \\
    0 \end{bmatrix} = \begin{bmatrix}
    0 S \cdot 1 N + 0 Q_2 \cdot 1 C_2 \\
    0 S \cdot 1 N + 0 Q_2 \cdot 1 C_2 + 0 Q_1 \cdot 1 C_1 \\
    0 S \cdot 1 N + 0 Q_2 \cdot 1 C_2 + 0 Q_1 \cdot 1 C_1
\end{bmatrix} \]

When \( N \) is taken equal to \( C \), only the translational part \( \overrightarrow{OC} \) is left because all the components of \( \overrightarrow{OC} \) are zero:

\[ \overrightarrow{C} = \overrightarrow{OC} = I^{-1} \cdot \overrightarrow{OC} \]

And in matrix representation:

\[ \begin{bmatrix}
    0 C_1 \\
    0 C_2 \\
    0 \end{bmatrix} = \begin{bmatrix}
    0 S \cdot 1 N + 0 Q_2 \cdot 1 C_2 \\
    0 S \cdot 1 N + 0 Q_2 \cdot 1 C_2 + 0 Q_1 \cdot 1 C_1 \\
    0 S \cdot 1 N + 0 Q_2 \cdot 1 C_2 + 0 Q_1 \cdot 1 C_1
\end{bmatrix} \]

Note 1: Sometimes it might be more convenient to use the magnitude of the velocity vector on a point of the vehicle instead of \( \dot{\omega} \) as input for the kinematic module. \( \dot{\omega} \) can then be found by solving Eqn. 93 with the known velocity vector. When point \( R \) or \( F \) is chosen for this purpose, this is very easy because \( \varphi_R \) and \( \varphi_F \) are already known. When point \( C \) is chosen, \( \varphi_C \) can be found with basic trigonometry with the point \( Q \).

Note 2: Mathematical problems arise, when \( \varphi_R \) and \( \varphi_F \) are equal (linear movement) or \( \omega = 0 \) (linear movement, or no movement). For these cases the algorithm must be adapted in software to prevent divide-by-zero errors.
3.2 The inverse kinematic module

This module performs the inverse operation of the previous one. It has $\vec{C}$ and $\vec{\omega}$ as input, and will calculate $\varphi_R$ and $\varphi_F$. Operation of this module is similar, but of course in reverse, to the kinematic module. Therefore it will only be described shortly.

First the rotation angle $\rho$ between CS0 and CS1 must be calculated from $\vec{\omega}$ with Eqn. 98. Now the tensors $I$ and $I^{-1}$ are defined as in Eqn. 99 to Eqn. 102. With these, $\vec{C}$ can be calculated in CS1:

Eqn. 110 $\vec{C} = I \cdot \vec{OC} = 0 \cdot \vec{e}^T \cdot \left( 1 \cdot 0 \cdot \vec{R} \cdot \vec{e}^T \cdot \vec{C} \right) = 0 \cdot \vec{e}^T \cdot \vec{C}$

Eqn. 111 $\vec{C} = \vec{OC} = 0 \cdot \vec{e}^T \cdot \left( 0 \cdot \vec{C} \cdot \vec{e}^T \cdot \left( 0 \cdot \vec{R} \cdot \vec{e}^T \cdot \vec{C} \right) = \vec{e}^T \cdot \vec{C}$

Vector $\vec{CQ}$ can be solved from Eqn. 97:

Eqn. 112 $\vec{Q} = \vec{CQ} = \vec{e}^T \cdot \vec{1} \cdot \vec{e}^T \cdot \left[ \begin{array}{c} -\tilde{\tilde{\varphi}}_R \\tilde{\tilde{\varphi}}_F \\end{array} \right]$

Finally, to find $\varphi_R$ and $\varphi_F$ the equations formed by Eqn. 112 combined with Eqn. 89 must be solved:

Eqn. 113 $\left\{ \begin{array}{l} -\tilde{\tilde{\varphi}}_R \\frac{\dot{\tilde{\varphi}}_2 - \frac{1}{\omega} \cdot \frac{\dot{\tilde{\varphi}}_1}{\omega} \cdot \tan(\varphi_R) - \frac{\dot{\tilde{\varphi}}_1}{\omega} \cdot \tan(\varphi_F)}{\omega} = \frac{\dot{\tilde{\varphi}}_2 - \frac{1}{\omega} \cdot \frac{\dot{\tilde{\varphi}}_1}{\omega} \cdot \tan(\varphi_R) - \frac{\dot{\tilde{\varphi}}_1}{\omega} \cdot \tan(\varphi_F)}{\omega} \\
\frac{\dot{\tilde{\varphi}}_1}{\omega} = \frac{\dot{\tilde{\varphi}}_2 - \frac{1}{\omega} \cdot \frac{\dot{\tilde{\varphi}}_1}{\omega} \cdot \tan(\varphi_R) - \frac{\dot{\tilde{\varphi}}_1}{\omega} \cdot \tan(\varphi_F)}{\omega} \end{array} \right.$

The solution of this set of equations is:

Eqn. 114 $\left\{ \begin{array}{l} \varphi_R = \arctan \left( \frac{\dot{\tilde{\varphi}}_2}{\dot{\tilde{\varphi}}_1} \right) \\
\varphi_F = \arctan \left( \frac{\dot{\tilde{\varphi}}_2}{\dot{\tilde{\varphi}}_1} \right) \end{array} \right.$

3.3 The vehicle force module

The vehicle force module has $\vec{C}$ and $\vec{\omega}$ as inputs, and the forces $\vec{f}_R$ and $\vec{f}_F$ relative to the vehicle CS as outputs. As explained later in this paragraph the actual output is the lateral component of $\vec{f}_R$, the lateral component of $\vec{f}_F$ and the sum of the longitudinal components of $\vec{f}_R$ and $\vec{f}_F$. These forces are sketched in Figure 10. The mass-centre point CS (CS1) is used again, as shown in Figure 10.

![Figure 10: Overview of vectors used in the vehicle force calculations](image-url)
Calculating the forces is done by solving \( \vec{F}_F \) and \( \vec{F}_R \) from the following two vector equations (from these can be seen that the mass \( m \) and rotational-inertia \( I_c \), are also needed):

\[
\text{Eqn. 115} \quad \begin{cases} 
\vec{f}_{\text{in,c}} = \vec{f}_F + \vec{f}_R = m \cdot \hat{\omega} \\
\vec{M}_{\text{ic}} = \vec{F} \times \vec{f}_F + \vec{R} \times \vec{f}_R = I_c \cdot \hat{\omega}
\end{cases}
\]

The following information must be used to solve Eqn. 115.

\[
\text{Eqn. 116} \quad \hat{\omega} = \begin{bmatrix} 0 \\ 0 \\ \hat{\omega} \end{bmatrix}, \quad \begin{bmatrix} \hat{f}_F \cdot \hat{e}_1 \\ \hat{f}_F \cdot \hat{e}_2 \\ \hat{f}_F \cdot \hat{e}_3 \end{bmatrix} = \begin{bmatrix} \hat{f}_{R,1} \cdot \hat{e}_1 \\ \hat{f}_{R,1} \cdot \hat{e}_2 \\ \hat{f}_{R,1} \cdot \hat{e}_3 \end{bmatrix}, \quad \begin{bmatrix} \hat{f}_{R,2} \cdot \hat{e}_1 \\ \hat{f}_{R,2} \cdot \hat{e}_2 \\ \hat{f}_{R,2} \cdot \hat{e}_3 \end{bmatrix} = \begin{bmatrix} \hat{C}_1 \\ \hat{C}_2 \\ 0 \end{bmatrix}
\]

According to [4] a planar movement of a body from one point to the next can be interpreted in two ways. First it can be interpreted as a series of pure rotations over an angle around a rotation point. For every rotation segment, the angle of rotation and the rotation point can be different (e.g. the movement could be 40 degrees of rotation around \( Q_1 \), followed by 20 degrees of rotation around \( Q_2 \)).

The second way of looking at the movement is to see it as one translation of a point of the body, followed by one rotation around the same point. When the first description would exist of only one rotation, the angle of this rotation will be the same as the rotation angle in the equivalent second description. In the kinematic model, a complete vehicle path exists of a number of very small rotation segments (see the integration in Eqn. 98, Eqn. 103 and Eqn. 104). According to the previous, each of these rotation segments can also be seen as a translation of \( C \), followed by a rotation around \( C \) over the same angle as the according pure rotation segment. Therefore, the angular velocity vector in Eqn. 115 is the same as the one in Eqn. 97. Due to the movement being planar, the third component of all vectors, except the angular velocity vector is zero. The vectors describing the front and rear point also have a second component equal to zero, because of the chosen vector basis (see Figure 10).

Now Eqn. 115 may be solved:

\[
\text{Eqn. 117} \quad \begin{bmatrix} \hat{f}_{F,1} + \hat{f}_{R,1} = m \hat{C}_1 \\ \hat{f}_{F,2} + \hat{f}_{R,2} = m \hat{C}_2 \\ \hat{f}_{F,1} + \hat{R}_1 \cdot \hat{f}_{R,2} = I_c \cdot \hat{\omega} \end{bmatrix}
\]

\( \hat{f}_{F,2} \) and \( \hat{f}_{R,2} \) follow directly from Eqn. 117:

\[
\text{Eqn. 118} \quad \begin{bmatrix} \hat{f}_{F,2} = \frac{I_c \cdot \hat{\omega} - \hat{R}_1 \cdot m \hat{C}_2}{\hat{F}_1 \cdot \hat{R}_1} \\ \hat{f}_{R,2} = \frac{\hat{F}_1 \cdot m \hat{C}_2 - I_c \cdot \hat{\omega}}{\hat{F}_1 \cdot \hat{R}_1} \end{bmatrix}
\]

So with Eqn. 118 the lateral components of the forces can be calculated. The longitudinal components however cannot be calculated explicitly (see Eqn. 117). This is due to the fact that it makes no difference at which point along the vehicle's length these components work (e.g. \( \hat{f}_{F,1} = 0 \), \( \hat{f}_{R,1} = f \) has exactly the same effect as \( \hat{f}_{F,1} = f , \hat{f}_{R,1} = 0 \)). Therefore the total longitudinal force will be used as an output, instead of the separate front and rear parts. In another module (i.e. the tire force module) these separate parts can be calculated by using the additional assumption of a rear-wheel driven vehicle.
3.4 The inverse vehicle force module

This module has the forces \( \vec{f}_R \) and \( \vec{f}_F \) relative to CS1 as input, and outputs \( \vec{C} \) and \( \vec{\omega} \). Looking at Eqn. 117, it is easy to find the equations for the inverse vehicle force module:

\[
\begin{align*}
\vec{C}_1 &= \frac{1}{m} (\vec{f}_{F,1} + \vec{f}_{R,1}) \\
\vec{C}_2 &= \frac{1}{m} (\vec{f}_{F,2} + \vec{f}_{R,2}) \\
\vec{\omega} &= \int \frac{1}{I_c} (\vec{f}_{F,2} + \vec{f}_{R,2}) d
\end{align*}
\]

Eqn. 119

3.5 The tire-force module A

This tire-force module has the following inputs: \( \delta_F \), \( \delta_R \), the lateral components of \( \vec{f}_R \) and \( \vec{f}_F \) relative to CS1 (i.e. \( f_{R,1} \) and \( f_{R,2} \)) and the sum of the longitudinal components of \( \vec{f}_R \) and \( \vec{f}_F \) relative to CS1 (i.e. \( \{ f_{R,1}, f_{F,1} \} \)). The outputs of this module are \( \varphi_R \) and \( \varphi_F \).

For simplicity, the tire-force module uses a linear tire model (see Eqn. 120 and Eqn. 121) the cornering stiffnesses (\( C_R \) and \( C_F \)) indicate the ability of the tires to generate a lateral force, depending on the slip angle (i.e. \( \delta_R - \varphi_R \) and \( \delta_F - \varphi_F \)). This model only takes into account lateral slip, and only the linear part of a true tire-slip characteristic. The linear slip model is described by the following equations:

\[
\begin{align*}
\text{Eqn. 120 } & \quad \varphi_F = \frac{2 f_{F,2}}{C_F} \\
\text{Eqn. 121 } & \quad \varphi_R = \frac{3 f_{R,2}}{C_R}
\end{align*}
\]

\( f_{F,2} \) (the lateral force at the front tire) and \( f_{R,2} \) (the lateral force at the rear tire) can be calculated from the forces \( f_{F,1}, f_{F,2}, f_{R,1}, \) and \( f_{R,2} \).

![Figure 11: Forces and CS's used in tire-force calculation module.](image)

The front and rear forces in CS2 and CS3 can be expressed by:

\[
\text{Eqn. 122 } \quad \vec{f}_F = \begin{bmatrix} f_{F,1} \\ f_{F,2} \\ 0 \end{bmatrix}, \quad \vec{f}_R = \begin{bmatrix} f_{R,1} \\ f_{R,2} \\ 0 \end{bmatrix}
\]

28
The forces at the front point relative to CS1 can be expressed in CS2 (similar to example 1b) with:

\[ \mathbf{F}_f = I \cdot \mathbf{f}_f = \begin{bmatrix} 2 \mathbf{e}^T \mathbf{e} & \mathbf{e}^T \mathbf{e} \end{bmatrix} \begin{bmatrix} \mathbf{f}_{F,1} \mathbf{f}_{F,2} \end{bmatrix} = \begin{bmatrix} 3 \mathbf{e}^T \mathbf{e} & \mathbf{e}^T \mathbf{e} \end{bmatrix} \begin{bmatrix} \mathbf{f}_{F,1} \mathbf{f}_{F,2} \end{bmatrix} = \begin{bmatrix} 3 \mathbf{f}_{F,1} \mathbf{f}_{F,2} \end{bmatrix} \]

The forces at the front point relative to CS1 can be expressed in CS2 (similar to example 1b) with:

\[ \mathbf{F}_f = I \cdot \mathbf{f}_f = \begin{bmatrix} 2 \mathbf{e}^T \mathbf{e} \text{ Rot}(1) \mathbf{e} \end{bmatrix} \begin{bmatrix} \mathbf{f}_{F,1} \mathbf{f}_{F,2} \end{bmatrix} = \begin{bmatrix} 2 \mathbf{e}^T \mathbf{e} \end{bmatrix} \begin{bmatrix} \mathbf{f}_{F,1} \mathbf{f}_{F,2} \end{bmatrix} = \begin{bmatrix} 2 \mathbf{f}_{F,1} \mathbf{f}_{F,2} \end{bmatrix} \]

Equation 121 can be solved from Eqn. 125 and Eqn. 126, resulting in:

\[ \mathbf{f}_{F,1} = \mathbf{f}_{F,1} \mathbf{c}_{8F} + \mathbf{f}_{F,2} \mathbf{s}_{8F} \]

When the tire roll-resistance is assumed zero, there can only be a non-zero longitudinal tire force at the driving tire, which is the rear tire because of the assumption of a rear-wheel driven vehicle. This means the front-tire only has a lateral force working on it. The longitudinal component \( \mathbf{f}_{F,1} \) must therefore be zero (i.e. \( \mathbf{f}_{F,1} = 0 \)). This means \( \mathbf{f}_{F,1} \) and \( \mathbf{f}_{F,2} \) are now related by

\[ \mathbf{f}_{F,1} = \mathbf{f}_{F,2} \mathbf{c}_{8F} \]

With this, \( \mathbf{f}_{F,2} \) can be solved from Eqn. 125 and Eqn. 126, resulting in:

\[ \mathbf{f}_{F,2} = \mathbf{f}_{F,2} \mathbf{c}_{8F} + \mathbf{f}_{F,2} \mathbf{s}_{8F} \]

Equation 127 combined with Eqn. 117 can be used to calculate \( \mathbf{f}_{R,1} \):

\[ \mathbf{f}_{R,1} = \mathbf{f}_{R,1} \mathbf{c}_{8R} + \mathbf{f}_{R,2} \mathbf{s}_{8R} \]

\( \mathbf{f}_{R,1} \) must be interpreted as a single variable, because only the sum of the longitudinal components is an input of this module, not the separate parts.

Similar to Eqn. 125 and Eqn. 126, two equations can be made for the rear point:

\[ \mathbf{f}_{R,1} = \mathbf{f}_{R,1} \mathbf{c}_{8R} + \mathbf{f}_{R,2} \mathbf{s}_{8R} \]

These equations can be solved with Eqn. 129:

\[ \mathbf{f}_{R,1} = \mathbf{f}_{R,1} \mathbf{c}_{8R} + \mathbf{f}_{R,2} \mathbf{s}_{8R} \]

\( \mathbf{f}_{R,2} \) can be solved from Eqn. 130 and Eqn. 128, resulting in:

\[ \mathbf{f}_{R,2} = \mathbf{f}_{R,2} \mathbf{c}_{8R} + \mathbf{f}_{R,2} \mathbf{s}_{8R} \]

Substituting Eqn. 134 and Eqn. 128 in Eqn. 121 respectively Eqn. 120 will result in the final equations describing the tire force module:
3.6 The tire-force module B

The second tire-force module is very similar to tire-force module B, but the inputs for tire-force module B are: $\delta_F, \delta_R, \varphi_R, \varphi_F$, the longitudinal component of $\vec{f}_R$ relative to CS3 (i.e. $f_{R.1}$: the driving force at the rear tire) and the longitudinal component of $\vec{f}_F$ relative to CS2 (i.e. $f_{F.1}$: the driving force at the front tire). The outputs are $\vec{f}_R$ and $\vec{f}_F$ relative to CS1. The slip model used is the same as in tire-force module A, but it may be rewritten to better fit this module:

Eqn. 136 $\vec{f}_{F.2} = C_F(\delta_F - \varphi_F)$

Eqn. 137 $\vec{f}_{R.2} = C_R(\delta_R - \varphi_R)$

These lateral components together with the longitudinal components from the input form $\vec{f}_R$ and $\vec{f}_F$ relative to CS3 respectively.

These only have to be transformed to CS1 to get the output:

Eqn. 138 $\vec{f}_F = I^{-1} \cdot \vec{f}_F = \begin{bmatrix} \hat{e}_r^T \cdot \text{Rot}(\hat{\hat{e}}_r, \delta_F)^2 \cdot \hat{e}_r^T \cdot \bigg[ \begin{array}{c} 2 f_{F.1} \\ 2 f_{F.2} \\ 0 \end{array} \bigg] \\ \hat{e}_r^T \cdot \begin{array}{c} 1 e_r^T \\ 1 e_r^T \\ 0 \end{array} \end{bmatrix}$

Eqn. 139 $\vec{f}_R = I^{-1} \cdot \vec{f}_R = \begin{bmatrix} \hat{e}_r^T \cdot \text{Rot}(\hat{\hat{e}}_r, \delta_R)^3 \cdot \hat{e}_r^T \cdot \bigg[ \begin{array}{c} 3 f_{R.1} \\ 3 f_{R.2} \\ 0 \end{array} \bigg] \\ \hat{e}_r^T \cdot \begin{array}{c} 1 e_r^T \\ 1 e_r^T \\ 0 \end{array} \end{bmatrix}$

Which finally results in:

Eqn. 140 $\vec{f}_F = \begin{bmatrix} c \delta_F \cdot 2 f_{F.1} + s \delta_F \cdot 2 f_{F.2} \\ s \delta_F \cdot 2 f_{F.1} - c \delta_F \cdot 2 f_{F.2} \\ 0 \end{bmatrix}$

Eqn. 141 $\vec{f}_R = \begin{bmatrix} c \delta_R \cdot 3 f_{R.1} - s \delta_R \cdot 3 f_{R.2} \\ s \delta_R \cdot 3 f_{R.1} + c \delta_R \cdot 3 f_{R.2} \\ 0 \end{bmatrix}$

3.7 The inverse tire-force module

This final module uses $\varphi_R, \varphi_F$, the lateral components of $\vec{f}_R$ and $\vec{f}_F$ relative to CS1 (i.e. $f_{R.2}$ and $f_{F.2}$) and the sum of the longitudinal components of $\vec{f}_R$ and $\vec{f}_F$ relative to CS1 (i.e. $f_{R.1} + f_{F.1}$) as inputs. The outputs for this module are $\delta_F, \delta_R, \vec{f}_R$ relative to CS3 and $\vec{f}_F$ relative to CS2.

Eqn. 142 $\delta_F = \varphi_F + \frac{1}{C_F} f_{F.2}$

Eqn. 143 $\delta_R = \varphi_R + \frac{1}{C_R} f_{R.2}$

Using Eqn. 128 and Eqn. 134 these can be expressed as a function of the input variables:

Eqn. 144

$$
\begin{bmatrix}
    c \delta_F (\delta_F - \varphi_F) - \frac{f_{F.2}}{C_F} = 0 \\
    C_R \cdot (\delta_R - \varphi_R) - \frac{f_{R.2}}{C_R} \cdot c \delta_R + \left( \frac{1}{C_R} f_{R.1} + \frac{1}{C_F} f_{F.1} \right) \cdot \tan(\delta_F) \cdot s \delta_R = 0
\end{bmatrix}
$$
To find $\delta_F$ and $\delta_R$ Eqn. 144 must be solved. Actually this means there is an algebraic loop within this module. This set of equations cannot be solved analytically and a numerical method must be used for finding $\delta_F$ and $\delta_R$. The correct solution for $\delta_F$ and $\delta_R$ must be in the interval $[-\pi/2, \pi/2]$. The standard multi-purpose solvers in Simulink often found (mathematically correct) solutions outside this interval. In the Simulink models, [5] was used to find a numerical method with fast convergence (combination of Bi-section and Newton-Raphson) to solve this problem. This method worked fine during the testing.

In the interval $[-\pi/2, \pi/2]$ there can be 0, 1 or 2 solutions for $\delta_F$. It should be noted however that for the first equation in Eqn. 144, in the case of 0 solutions $\delta_F$ will converge to $\pm \pi/2$ which could be detected, this indicates that the input path cannot be realised (e.g. because of to tight turns). The case of one solution occurs when the path can barely be realised. Two solutions is the nominal situation in which the solution closest $\phi_F$ should be selected. The implemented solver starts with an initial guess equal to $\phi_F$ it is unlikely (but in extreme situations not impossible) that this results in the wrong solution of the two. For $\delta_R$ it is much harder to say how much solutions there will be on the interval.

However after numeric testing, the preliminary conclusion was drawn that (at least for realistic values) there is always one, and only one solution in the interval for $\delta_R$.

The mentioned adaptations (0 solution detection and better solution selection in the case of 2 solutions) of the solver could be implemented to make it more robust.

The longitudinal components of $\vec{f}_R$ and $\vec{f}_F$ relative to CS1 can now be found by solving Eqn. 132 and Eqn. 126 with the $\delta_R$ and $\delta_F$ solved from Eqn. 144 and $f_{R,2}$ and $f_{F,2}$ (that can be easily found with the solved steering angles and Eqn. 136 and Eqn. 137):

$$\text{Eqn. 145} \quad f_{F,1} = \frac{f_{R,2} \cdot c \delta_R - f_{R,2}}{s \delta_R}$$

$$\text{Eqn. 146} \quad f_{F,1} = \frac{f_{F,2} \cdot s \delta_R - f_{F,2}}{s \delta_R}$$

With the lateral components from the input, the forces $\vec{f}_R$ and $\vec{f}_F$ relative to CS1 are now completely defined, and can be transformed to CS3 respectively CS2 with a unity tensor:

$$\text{Eqn. 147} \quad \vec{f}_F = I \cdot \vec{f}_F = \begin{pmatrix} \vec{e}_T \cdot \text{Rot}((1,1,1,1)) \cdot \vec{e} \end{pmatrix} \cdot \begin{pmatrix} f_{F,1} \\ f_{F,2} \end{pmatrix}$$

$$\text{Eqn. 148} \quad \vec{f}_R = I \cdot \vec{f}_R = \begin{pmatrix} \vec{e}_T \cdot \text{Rot}((1,1,1,1)) \cdot \vec{e} \end{pmatrix} \cdot \begin{pmatrix} f_{R,1} \\ f_{R,2} \end{pmatrix}$$

resulting in:

$$\text{Eqn. 149} \quad \vec{f}_F = \begin{pmatrix} c \delta_F \cdot f_{F,1} + s \delta_F \cdot f_{F,2} \\ c \delta_F \cdot f_{F,2} + s \delta_F \cdot f_{F,1} \end{pmatrix} = \begin{pmatrix} c \delta_F \cdot f_{F,1} + s \delta_F \cdot f_{F,2} \\ c \delta_F \cdot f_{F,2} + s \delta_F \cdot f_{F,1} \end{pmatrix}$$

$$\text{Eqn. 150} \quad \vec{f}_R = \begin{pmatrix} c \delta_R \cdot f_{R,1} + s \delta_R \cdot f_{R,2} \\ c \delta_R \cdot f_{R,2} + s \delta_R \cdot f_{R,1} \end{pmatrix} = \begin{pmatrix} c \delta_R \cdot f_{R,1} + s \delta_R \cdot f_{R,2} \\ c \delta_R \cdot f_{R,2} + s \delta_R \cdot f_{R,1} \end{pmatrix}$$

31
3.8 The full vehicle model

The modules described before can now be used to build a complete vehicle model. One point of interest is the lack of dynamic behaviour in any of the modules, while full vehicle models that use slip characteristics always show dynamic behaviour. This would indicate that some sort of feedback loop between modules must be present in the complete model that causes the dynamic behaviour. The two model structures looked at in this paragraph indeed show this feedback loop.

There are a number of ways to build a complete model from the separate modules. Simulations in Simulink have been constructed and tested for two different model structures. The first of these uses the kinematic module, the vehicle force module and tire force module A; it will be called vehicle model A. This structure was tested first because of a personal (and consequently subjective) preference for the interpretation of this structure. Figure 12 shows the block diagram of vehicle model A.

The feedback that causes the dynamic behaviour is clearly visible. One way of interpretation this model is by starting at the kinematic module, while imagining the connection between tire force module A and the kinematic module as disconnected. Assuming that \( \varphi_K, \varphi_F \) and \( \tilde{\omega} \) are known, the kinematic module can calculate the acceleration of the mass-centre point. An initial condition for the integrations is needed but these can be chosen to be zero; meaning that for \( t=0 \) CS1 is equal to CS0. The vehicle force module calculates the forces relative to CS1 that are needed to get the motion defined by \( \dot{\tilde{C}} \) and \( \tilde{\omega} \). The forces relative to CS1 combined with the steering angles \( \delta_K \) and \( \delta_F \) are used in tire force module A to calculate a new \( \varphi_K \) and \( \varphi_F \). These new angles will in general not be the same as the angles that entered the kinematic module. To restore the connection between tire force module A and the kinematic module (which were considered disconnected) the angles entering the kinematic module must be "chosen" such that they are equal to the output of tire force module A. With a implementation in Simulink this means that during every time step, Simulink must solve a non-linear multivariable set of equations to find the correct \( \varphi_K \) and \( \varphi_F \) (i.e. the model contains an algebraic loop). Due to the complex nature of the equations Simulink must solve each step, this leads to a model requiring a lot of computing power. Furthermore the model proved to be stable only when the inputs were small or changed slowly. It may be possible to improve the computational cost and the stability problems by incorporating a customised solver instead of the standard general-purpose Simulink solvers (something similar was done in the path-to-inputs module). The information that the angles must be in the interval \([ -\pi/2, \pi/2 ]\) might be used for example.

The second model structure investigated uses tire force module B, the inverse vehicle force module and the inverse kinematic module; this model will be called vehicle model B. The following figure shows the block diagram.

![Figure 12: Block diagram of the first full model structure; vehicle model A](image)

![Figure 13: Block diagram of the second full model structure; vehicle model B](image)
Vehicle model B can be implemented in Simulink, without creating an algebraic loop. This is because the feedback loop contains at least one integrator for every variable in the loop. In vehicle model A the feedback loop contains no integrator for some variables, creating a direct feed through path resulting in an algebraic loop. The lack of an algebraic loop makes vehicle model B more stable and faster than vehicle model A. One problem with this structure lies with the initial condition of the integration in Eqn. 111. The initial condition (actually 3 values for the three components of \( \dot{\mathbf{C}} \)) represents the velocity components of the vehicle relative to CS1 at t=0. Setting this initial condition equal to zero (meaning the vehicle initially does not move relative to CS0) will cause extreme behaviour of \( \phi_k \) and \( \phi_f \) during the first part of the simulation. This extreme behaviour shows angles that rapidly approach, and then maintain values of \( \pm \frac{\pi}{2} \) before settling at more acceptable, but incorrect values. The cause of these problems lies in the fact that Eqn. 114 is actually not defined if the longitudinal component of the velocity at point C is zero, the value this equation will give in this case is \( \pm \frac{\pi}{2} \). These extreme initial values will result in the observed behaviour.

Setting the initial condition for the integrator to a non-zero value can prevent this; an initial vehicle velocity can be assumed this way. If the initial velocity must be zero, the initial condition may be set to a small (but not too small), non-zero value. In the simulations, an initial condition of \( \begin{bmatrix} 0^{-2} & 0 & 0 \end{bmatrix} \) was used. The velocity offset would be fairly small, but large enough to prevent extremely small step sizes in the Simulink model (which would result in a very slow implementation). Both model structures mentioned here have the advantage over many simple models, that the vehicle velocity does not have to be constant and known. A simplified model could be constructed by changing the equations in the modules to fit a constant, known velocity.

### 3.9 The path-to-inputs structure

Another interesting structure can be build with the inverse tire force module, the inverse kinematic module and the vehicle force module; this structure will be called the path-to-inputs structure.

The figure below shows the connection of the modules.

![Block diagram of the path-to-inputs structure.](image)

The new conversion module simply converts the position and orientation to the acceleration and angular velocity used in the other modules. The equations used for this are:

\[
\mathbf{\ddot{C}} = \begin{bmatrix} 0 & 0 & \dot{\rho} \end{bmatrix}^T
\]

\[
\mathbf{\ddot{C}} = I^{-1} \cdot \mathbf{\ddot{C}} = \left( e^{-T} \cdot \text{Rot}(\mathbf{e}_1, -\rho) \cdot \mathbf{e}_2 \right) \cdot \left( \dot{e}_2 - \mathbf{e}_2 \cdot \mathbf{\dot{C}} \right) = e_2^{-1} \cdot \dot{\mathbf{C}}
\]

This structure can be used to find the steering angles and longitudinal tire forces that are needed to get a desired path. A possible application for the path-to-inputs structure can be in a control system, which has to make a vehicle follow a desired path. The next chapter describes an example of the basic structure of such a controller.
Some tests were done with the path-to-inputs structure. For this purpose, a path was generated by using the kinematic module. The created path was then fed into the path-to-inputs structure, resulting in the inputs $\delta_F$, $\delta_R$, the longitudinal component of $\tilde{f}_F$ relative to CS3 and the longitudinal component of $\tilde{f}_F$ relative to CS2. These inputs are then fed into both vehicle model A and B, finally resulting in two paths. These paths should theoretically be identical to the path initially created with the kinematic module. The three paths are not exactly identical however. The three paths are shown in Figure 15; there is hardly any visible difference between the three paths.

The input path was created in the kinematic module, with angles $\phi_R$ and $\phi_F$ gradually increasing from 0 to −0.1 respectively 0.15, and then decreasing back to zero. The velocity gradually increased from 0 to 5.5 m/s. As shown below:

The output variables of the path-to-inputs module (and input variables of vehicle model A and B) are shown in Figure 17.
From Figure 15 it appears the path-to-inputs structure works fine after visual inspection. To get a better idea of the path error, the difference between the input and output paths of both the first and second component are shown in the two following figures.

![Figure 18: Absolute error between input and output path (left: vehicle model A, right: vehicle model B) for the first (solid) and second (dashed) component.](image)

First of all, it should be noted that vehicle model A stops the simulation at t=13.8 seconds, because Simulink could not solve the algebraic loop at that time. In all structures the Simulink solver was set to the variable ODE45 solver, while leaving all solver-settings to the standard setting (except for the maximum step size that was set to 0.05 seconds). When the maximum step size was set to auto (resulting in a maximum step size of 0.3 seconds) the simulation worked properly during the full 15 seconds. This has been encountered often while working with vehicle model A: changing the solver and/or its settings often resulted in better stability. However, what worked in one case often did not work in other situations.

It is also clear that, at least in this situation, vehicle model A (until the simulation is broken off) recreates the initial path with a much smaller error than vehicle model B. But even with vehicle model B the error remains relatively small. Around t=7.5 seconds, the rotation of the vehicle is almost zero, and the movement becomes linear shortly after. At this time, the deviation from the input path has caused a small error in the orientation at the start of the linear part of the path. The absolute error will of course increase over time while continuing to move in a straight line that has a slightly different angle. During the rotation, the error does not get larger than a few centimetres.

A possible reason why the error is so much larger with vehicle model B could lie in the initial condition that had to be set to a non-zero value for the velocity integrator. This seems however less likely because when the initial condition was set to a lower value the magnitude of the error showed hardly any change. The shape of the error did change however, and the model became extremely slow because the step size the variable step size solver needed became extremely small at the beginning of the simulation. The fact that there is a small error (also in vehicle model A) might be caused by numerical errors. One of these numerical problems might be caused by the fact that the three steps (building the path, finding the needed inputs, and using these inputs in the vehicle model) were implemented in three different Simulink files, where the output was saved to the workspace. The input for the next block was read from the workspace. Due to the use of variable step size solvers, the data will not be read at the exact sample times the data was saved in. Simulink uses linear interpolation between the closest data samples to find the estimated value. This might cause a small error (like the error seen in vehicle model A), but this can only explain the difference in error size between vehicle model A and vehicle model B if this small error would somehow (e.g. by integration) be increased in vehicle B, but not (or not as much) in vehicle model A. The real cause for the large difference in the error magnitude remains unknown at this moment. The relative error is however not very large in model B either, so it depends on the application if this error is acceptable.
3.10 Vehicle control systems using the path-to-inputs structure

As mentioned earlier, the path-to-inputs structure can be put to use in a vehicle control system. One way of using this structure is in a pure feed-forward function, as shown below:

![Figure 19: Feed forward usage of the path-to-inputs structure](image)

Although Figure 19 doesn't show this, the "vehicle" may contain some sort of controller on its own, for example to reduce the influence of actuator errors, or an observer to estimate certain model states. The system in Figure 19 has all the usual problems of feed forward control: if the vehicle behaves not exactly as expected (even if this deviation is short), the input delivered from the path-to-inputs block does not get compensated in any way. The resulting error can build up and creates a serious deviation over time.

Adapting the path fed into the path-to-inputs block could compensate this. This would result in a structure like Figure 20.

![Figure 20: Feedback usage of the path-to-inputs structure](image)

The path could be adapted in many ways, one possibility would be to construct the real path from measurements on the true vehicle, and adapt the input path for the difference between the "measured" and input path. This updating of the path is not necessary continuously, because the deviation from the input path will in general be fairly small initially, but increasing over time. The updating of the path could for example happen every 5 seconds, or depending on the vehicle speed. Extreme situations that change the path drastically in short time should be detected to update the path sooner (for example strong sideways wind gusts), or stop the vehicle completely (for example in the case of a collision).

3.11 Main conclusions for the modular vehicle models

Splitting a vehicle model into separate functional modules can be an effective way to make vehicle models easier to handle and understand, for the following reasons:

When the model is implemented, the obvious advantages are:

- Understanding how the model operates physically can be easier.
- While implementing the model (e.g. in Simulink) changes to parts of the model are easier to make, because the modular structure makes it easy to change only part of the model (for example to implement a better tire model).
- Testing the separate modules will be more easy than testing the entire model at once.

Of the two vehicle model structures shown in this chapter, vehicle model B will be the most useful because of the better stability and computational load, although vehicle model A appears to be more accurate when it actually works. From the few tests that were done with the path-to-inputs structure, can be concluded that this structure is operating properly.
Chapter 4: Conclusions and recommendations

4.1 Conclusions

Most literature found on the subject of vehicle state-estimation uses an observer similar to a Kalman-filter (linear or extended and/or adaptive). None of this vehicle state-estimation literature involves articulated vehicles.

From the literature survey can now be concluded that some form of a Kalman observer is the most useful method for estimating the vehicle states that are hard to measure properly, mainly the slip angles. Because a small majority of the found literature on vehicle state-estimation uses a planar 2-DOF model (both bicycle and 4-wheel). The estimator performance does not appear to be very dependent on the vehicle complexity (number of DOF), so one could conclude that a Kalman observer based on a simple 2-DOF model is sufficient for most applications. In the case of a bus this might very well be different; e.g. the roll and pitch motions can become much larger with a bus than with a small car.

Literature found on articulated vehicles (and related multi-body set-ups) was found, but only described the control systems for these vehicles (no state estimation). It consisted of two subjects: train-like vehicles (with multiple connected vehicles, following the first vehicle) and platooning/automated highway systems (with multiple unconnected vehicles, following the first vehicle).

There was too much variation between the different articles about articulated vehicles to draw a conclusion about the best way to deal with these. Looking further into platooning systems might prove useful, although the absence in a platooning system of the mechanical constrained caused by the connection between cars (that is present in an articulated vehicle) will make a big difference. On one hand the lack of the constrained allows for far greater path deviations, but on the other hand the mathematics will become less complex.

At the moment, the best way to construct a vehicle state estimation system for multiple-articulated vehicles appears to be trying to expand the vehicle state estimation methods (for non-articulated vehicles) to fit an articulated vehicle. A good starting point would be to begin with a platooning system (there appears to be more literature on these systems then on articulated vehicles), and transform these into an articulated system by placing constraints that connect the separate vehicles; this will however be not so easy as it might seem.

The references found on driver modelling only describe a limited set of manoeuvres (often only lane changing on a highway), or a complete environment-driver-vehicle model. Driver modelling is mostly used for estimation of vehicle handling characteristics for designing new cars or for predicting driver behaviour for research on traffic flow (mostly on highways). Almost all references found on this subject are published in conference papers. This could indicate that research on this subject is not yet developed enough to justify many publications in journals (this can not be concluded with any certainty though). Therefore driver-modelling seems to be a pretty open issue, and no practically useful conclusion can be drawn from the found references.

By using tensors to describe CS transformations, the following advantages can be achieved:

• Because tensors operate independent of CS’s, the choice of the most convenient CS’s and matrix representations can be delayed to the very end of the calculation.

• When finally using the matrix representation of the tensors, the influence of the CS’s is more obvious in the calculations with tensors. What is happening during transformations can be clearly seen this way.

• Tensor notation is very compact. Only when finally using the matrix representation of the tensors, matrices and a number of indices become necessary.

The only real disadvantage of using tensors is that it takes time to get used to this way of describing transformations.

Despite these advantages, one must keep in mind that there is nothing that CAN be calculated with tensors, which CANNOT be calculated with other methods (like matrices and component columns as in example 1a). In situations where CS transformations are very simple and infrequent, the time it takes to get accustomed to using tensors might not be worth the investment. When CS transformations get more complex and frequent, the advantages will weigh up against the learning time for using tensors.

Comparing the tensor method described here to the method using matrices, one could say that using only matrices is a simplified representation of the tensor method. The simplification involves a certain choice of which CS’s are worked with. The amount of simplification depends on the exact
Tensors were used in the transformations. Therefore tensors can be interpreted as a more general way of describing CS transformations.

Tensors were used in the modules described in Chapter 3, but a complete model build with these models will not be entirely CS independent. This is because the forces that are passed between different modules are expressed relative CS1, CS2 or CS3. Although this would actually happen in the implementation, it might be better during the mathematical description on paper to take separate vectors for the different components. The rear tire force expressed in CS3 would for example become two separate vectors for both the lateral and longitudinal components, without directly assigning a CS to these vectors.

Splitting the vehicle model in a limited number of modules may help to better understand the operation of the model. With the modules it can for example be easier to understand the presence of the kinematic model inside the dynamic model than with just the complete set of equations. The kinematic model is sometimes thought to be just a simplified vehicle model; this is certainly true, but it is also part of every vehicle model (just not always visible). It consequently describes the actual movement of the vehicle, provided the right inputs are used (e.g. not $\delta_R$ and $\delta_L$, but $\varphi_R$ and $\varphi_L$).

Another advantage is that it may be easier to change certain parts of the model, because the modules split the model in functional blocks. To use a more complicated tire model for example, the (inverse) tire force module needs to be altered, while the other modules can remain untouched. It might be more complicated to see how far the influence of the change reaches into the model when provided with a set of equations (especially when the user is new to vehicle modelling).

When implementing the full vehicle model, it is much easier to test and debug several small modules than the entire model at once. A final advantage (as was shown with the structures in paragraphs 3.8 to 3.10) is that because of the modular properties, it’s easier to build different implementations and structures from the same set of modules. This can however also be a disadvantage, because taking certain equations out of a module might be useful sometimes, which won’t be obvious while sticking to the modular structure.

The strongest disadvantage of the modular structure lies in choosing the inputs and outputs of the different modules and the entire structure. Although this makes it far easier to imagine the system (often it comes almost natural for people to think in inputs and outputs), the fact that inputs and outputs are chosen introduces some problems.

An example of these problems has been encountered in vehicle model A. The choice of the inputs and outputs was the actual cause for the occurrence of the algebraic loop. This could have been predicted in advance, by first taking the equations that describe the different modules without assigning input and output variables. This way there is no distinction between modules and their inverse. From these modules without assigned inputs and outputs, it would have been easier to see the best way (e.g. without algebraic loops) to assign the inputs and outputs. It is not wrong to assign the inputs and outputs directly (as was done in this thesis), but one should always remember that this assignment is not just a trivial choice; it might very well have a strong influence on the final implementation of the model.

The final conclusion about the idea of using a modular structure for building a vehicle model as it is presented in this thesis is therefore two-sided. There are distinct advantages to using the modular structure: comprehension and handling of the total model are made easier. These advantages will be most apparent when choosing the inputs and outputs of the modules directly. However this choice can have a strong influence on the final implementation of the model and can cause problems in this implementation. These problems can be prevented by not assigning the inputs and outputs directly, but this would partially negate some of the advantages. Therefore a choice has to be made between either optimal ease of comprehension and handling in a system with possible mathematical problems, or somewhat less ease of comprehension and handling in a system that prevents these mathematical problems.

When looking at the two vehicle model structures shown in paragraph 3.8, the following can be said about vehicle model A:

- Vehicle model A contains an algebraic loop.
- Due to this algebraic loop, the implementation of the model often gets unstable after a (short) while and the algebraic loop can not be solved. This problem can sometimes be prevented or postponed by adjusting the solver settings in Simulink, but this is not a structural solution.
- Using a variable solver in Simulink, the step size will quickly become fairly large.
• Testing of the path-to-inputs structure showed that vehicle model A can be very accurate, when the algebraic loop does not cause problems (i.e. cannot be solved or results in extreme behaviour).

The following remarks concern vehicle model B:
• Vehicle model B contains no algebraic loop.
• The initial condition for the integration of the velocity vector on point C is very important. Setting the initial condition to zero results in unacceptably extreme behaviour.
• Depending on the magnitude of the previous mentioned initial condition; a variable solver in Simulink will initially need a very small stepsize. The larger this initial condition the faster the initial stepsize increases, at the cost of a velocity offset. A small value for the initial condition results in an initial stepsize that increases very slowly, but without the velocity offset.
• Vehicle model B does not get unstable over time.

4.2 Recommendations

The following recommendations can be made to improve upon the concepts shown here, and possibly prevent some of the flaws.

The use of tensors as described in Chapter 2 can be expanded to a full 3-dimensional 6-DOF situation. This will however be a straightforward extension. For example, instead of a rotation tensor describing a rotation over only one axis (as described in Chapter 2), the rotation tensor will now describe three rotations over three axes. This can easily be accomplished by taking the product (in the right sequence) of the three single rotation tensors.

Improvement in the use of tensors could also be accomplished by combining both the rotation- and translation effects in one tensor. The appendix "The extended 3-DOF case using homogenous coordinates" describes a way that could accomplish this by combining tensors with homogenous coordinates (the latter is described in [1]). Some questions still remain with this method, so further investigation is needed. With the current state of the discription in this appendix, the conclusion would be that it can be used to get the proper result, but some of the advantages of tensors (it is not even certain the "tensors" can actually be called tensors) appear to be cancelled and some interpretation problems remain.

Finally, it will be very useful to see if an expression similar to Eqn. 32 for integration instead of differentiation can be found. As integration can be tricky in the implementation of a model, another way of integration will be welcome. On first inspection, taking the inverse of the tensor $\mathbf{F}$ seems to work: for the situation in example 2 the components of the integral of $\mathbf{a}$ in CS0 can be calculated directly by taking the integral from the components in CS0 described on the top of page 14; taking the inverse of $\mathbf{F}$ and applying this tensor to $\mathbf{a}$ in CS1 results in the same values for the components in CS0. There was no time for further investigation, so looking deeper into this issue is advisable.

Improving the general idea of the modular model, could be accomplished by using the general equations for a block (which are exactly the same for its inverse) instead of directly assigning certain variables to be inputs and outputs. This would make it possible to predict certain problems (like algebraic loops) with the implementation, and is mathematically speaking a more general description of the module.

The modular vehicle model described in Chapter 3 could be improved by a combination of the following:
• Making the modules CS independent by taking vectors for the components of a force vector in a certain CS.
• Using a four-wheel vehicle instead of a bicycle.
• Connecting multiple vehicles, to get an articulated vehicle.
• Extension to a 3-dimensional movement, instead of planar.

The first option is actually easy, instead of the lateral component in CS3 of the rear-tire force vector the lateral rear-tire force vector is now introduced. Furthermore this can only be done in the mathematical description on paper. In an implementation the results will have to be calculated numerically, which can only be done by choosing a CS, and working with the vector components. Using a four-wheel vehicle instead of a bicycle will of course introduce two extra tires. Two situations can be distinguished: the 4 wheels can be steered independently, or two wheels are connected somehow (e.g. by an axle). Usually, this connection will be between the two front wheels, and the two
rear wheels. The only module that might cause problems is the kinematic module, adaptation of all other modules will be quite easy (basically adding two extra wheels).

The only correct position of the steering pole is when the velocity angles at two of the "wheel points" define a steering pole, that is exactly the same as the steering pole defined by the velocity angles at the other two "wheel points". This requires a strict constrained on the 4 velocity angles. The inverse kinematic module does not have this problem, because it calculates the velocity angles instead of using them as inputs. This is another example of the influence of assigning inputs and outputs to a module. In real life this is not a problem because situations where the velocity angles indicate different steering poles simply don't occur. If everything in the implementation of vehicle model A would operate perfectly, the correct velocity angles (with only one steering pole) would be achieved by the algebraic loop. However, the algebraic loop will most likely be even less stable.

Building an articulated vehicle model can be done by pretending the parts of the articulated vehicle are separate vehicles. When (for simplicities sake) an "articulated bicycle" is assumed (all wheels behind each other), the simplest articulated vehicle has three wheels, and one joint somewhere between two wheels. The two "separate" vehicles in this case would both have two wheels. One of the two vehicles would also have a joint between its two wheels. Because the rear wheel of the "front" vehicle is the same as the front wheel of the "rear" vehicle, the velocity vector at the front wheel of the "rear" vehicle will always be exactly the same as the velocity vector at the rear wheel of the "front" vehicle. This guarantees that both vehicles will always stay connected. The "front" vehicle and the "rear" vehicle will in general have different steering poles, but they will always be positioned on the line perpendicular to the velocity vector at the shared wheel. This basically means an articulated vehicle could be formed by taking two separate bicycle model as described in Chapter 3, and connecting them introducing a constrained that makes the velocity vector of the shared wheel the same for both vehicles.

Extension to the 3D requires a lot of (mostly) straightforward changes to the modules. Again sticking to a bicycle model for simplicity, extension to 3D would be useful to describe a motorcycle for example. The most important thing is that the effects of gravity must be added to the model, because the vertical forces, accelerations and velocities play an important role now. The relatively large rolling motion a motorcycle makes in a turn could be modelled this way, and the gravity is essential to know if the motorcycle can make a tight turn without falling over. Some of the other things that must be changed are:

- In general all points and vectors will now have three components.
- Three velocity angles per wheel are needed now.
- Calculating $Q$ will be far more complex because of the above reasons, but can still be done with basic trigonometry.
- Three orientation angles are needed instead of one ($\rho$), which also makes the transformation tensor contain three rotations instead of one.
- Three rotational inertias are needed (instead of one), for roll-, yaw- and pitchmotions.

What remains unchanged is the fact that there is only one steering angle and slip-angle per wheel.

First of all vehicle model B should be investigated thoroughly, to find out if the specific reason for the lower accuracy seen in paragraph 3.9 can be found and maybe avoided. To improve upon vehicle model A and B one could combine the two models. One possible option might be to start with vehicle model A, to avoid the small initial stepsize that will be present in vehicle model B. After a certain time (or another criteria like when the magnitude of the velocity vector at point C is higher than a certain threshold) model B can be used, with the initial condition in the integrator set to the last value the velocity vector has in model A.

If the difference in accuracy between both models cannot be prevented, another option might be to use vehicle model A whenever possible (for greatest accuracy), and only switch to vehicle model B when vehicle model A causes problems. The problem with this system is that the problems with vehicle model A must be detected before they become too much of a problem. This might prove difficult, and could even be impossible sometimes. Therefore the first way of combining the two models is probably the better.

Finally, the path-to-inputs structure seems to operate properly, and did not show any problems. To guarantee the right angle of the two nominal solutions is found, it may be necessary however to alter the Newton-Raphson solver or take another solver. This is because the Newton-Raphson solver does not guarantee the steering angles closest to the velocity angles are found, although under most practical circumstances the right angles will be found.
Appendices

The extended 3-DOF case using homogenous coordinates

Another possible way to express transformations between CS's and time derivatives in different CS's is by using homogenous coordinates (explained in [1], page 141). The benefit of using homogenous coordinates is that the whole transformation (rotations and translations) can be captured in a single matrix. However it is more difficult to interpret physical meaning of the calculations, and it adds another dimension in the vector/matrix calculations.

The decision was made to stop using this method because it was hard to interpret every aspect in an acceptable manner. Shortly after this decision was made, Bart Stouten (one of my coaches) went to a conference (20th Benelux Meeting on Systems and Control, March 26-28 2001, Houffalize Belgium) where a minicourse was given about “Geometry and Screw Theory for Robotics”. The book of abstracts of the conference (see [8]) showed that one of the concepts the course discussed was a technique, named “Projective geometry and kinematics”, that appeared similar in some ways to what is described in this appendix. The mathematical background of this technique is referred to (and explained briefly) in the minicourse.

If the text below (describing the combination of tensors and homogenous coordinates) should prove to be interesting to the reader, far more extensive information might be found by searching for literature about “Projective geometry and kinematics”. A starting point for this search might be [8].

The following paragraph describes the same situation as paragraph 2.8, which didn't use homogenous coordinates (i.e a rotation over and translations of and ).

The use of homogenous coordinates introduces a 4th dimension to all vectors and matrices, while the movement still takes place in 3 dimensional space. To explain the meaning of the homogenous coordinates, a 2 dimensional space is looked at. This makes the homogenous coordinates 3-dimensional, and makes it easier to represent the situation graphically.

A general vector in homogenous coordinates (2D space) will be:

\[
\begin{bmatrix}
\mathbf{e}_1 \\
\mathbf{e}_2 \\
p
\end{bmatrix}
= \begin{bmatrix}
\mathbf{e}_1^{'} \\
\mathbf{e}_2^{'} \\
p
\end{bmatrix}
\begin{bmatrix}
1_x \\
1_y \\
p
\end{bmatrix}
\begin{bmatrix}
\mathbf{C} \\
\mathbf{C}_o
\end{bmatrix}
= \begin{bmatrix}
1_x \\
1_y \\
p
\end{bmatrix}
\begin{bmatrix}
\mathbf{C} \\
\mathbf{C}_o
\end{bmatrix}
\]

It should be interpreted as follows: the components together with the base vectors and form a 2-dimensional vector (i.e. a length and a direction) independent of the origin. Component together with base vector should be interpreted as a point. This can be illustrated by the following figure:

![Figure 21: Explanation of the extra dimension in homogenous coordinates.](image-url)
The third base vector of a vector basis should be interpreted as a point located at a height of 1 above the origin of the vector basis. In Figure 21 this means \( \vec{e}_3 \) is the vector pointing to point \( C^* \) (with in CS1 the components 0, 0 and 1 for the base vectors \( \vec{e}_1 \), \( \vec{e}_2 \) and \( \vec{e}_3 \)). \( \vec{e}_3 \) would be the vector pointing to \( O^* \) (with in CS0 the components 0, 0 and 1 for the base vectors \( \vec{e}_1 \), \( \vec{e}_2 \) and \( \vec{e}_3 \)). It is important to notice that vectors \( \vec{e}_3 \) and \( \vec{e}_3 \) are different in each CS, but in every CS they point towards \( C^* \) respectively \( O^* \).

A vector in these homogenous coordinates (Eqn. 153) should therefore be looked at as: “the vector described by \( " x \vec{e}_1 \) and \( " y \vec{e}_2 \), relative to the point described by \( p\vec{e}_3 \)” This means that when \( p \) is equal to 1, the homogenous vector describes a point rather than a vector (point interpretation). A \( p \) of 0 means the vector describes a vector, independent of the origin of the system (vector interpretation).

If the vector basis for CS1 would be expressed as a function of the base vectors of CS0, the vector interpretation must be used for the transformation of \( \vec{e}_1 \) and \( \vec{e}_2 \). The point interpretation should be used for the transformation of \( \vec{e}_3 \) (which, as said before, indicates a point rather than a direction and length). CS1 expressed in CS0 will be:

\[
\begin{align*}
\text{Eqn. 154} \quad \vec{e}_1 ^T &= \begin{bmatrix}
\varphi \vec{e}_1 & + S \varphi \vec{e}_2 \\
\varphi \vec{e}_2 & - S \varphi \vec{e}_1 \\
0 C_1 & + 0 C_2 & + 0 C_3 
\end{bmatrix} \\
\text{Eqn. 154} \quad \vec{e}_2 ^T &= \begin{bmatrix}
\varphi \vec{e}_1 & + S \varphi \vec{e}_2 \\
\varphi \vec{e}_2 & - S \varphi \vec{e}_1 \\
0 C_1 & + 0 C_2 & + 0 C_3 
\end{bmatrix} \\
\text{Eqn. 154} \quad \vec{e}_3 ^T &= \begin{bmatrix}
\varphi \vec{e}_1 & + S \varphi \vec{e}_2 \\
\varphi \vec{e}_2 & - S \varphi \vec{e}_1 \\
0 C_1 & + 0 C_2 & + 0 C_3 
\end{bmatrix}
\end{align*}
\]

In Eqn. 154 the first two components express the direction (and length, which is 1) of \( \vec{e}_1 \) and \( \vec{e}_2 \), expressed in \( \vec{e}_1 \) and \( \vec{e}_2 \). The third component is the vector \( \overrightarrow{OC^*} \) (instead of \( \overrightarrow{CC^*} \) as it is in CS1), and effectively expresses the \( C^* \) in \( \vec{e}_1 \) and \( \vec{e}_2 \).

One problem arising from the use of homogeneous coordinates is that the third base vector is only orthonormal with other base vectors when all are expressed in the original CS:

\[
\text{Eqn. 155} \quad " \vec{e}_k \cdot \vec{e}_m = \begin{cases}
0 & \text{if } k \neq 3 \text{ and } m = n \\
1 & \text{if } k = 3 \text{ and } m = n \\
\in \mathbb{R} & \text{if } m \neq n
\end{cases}
\]

This means that in the tensor notation the vector the tensor is working on must be expressed in the same vector basis as the rightmost vector basis in the used matrix representation for that tensor. So with both \( \vec{e}_1 ^T \cdot a \) and \( \vec{e}_2 ^T \cdot a \) describing the same point:

\[
\text{Eqn. 156} \quad P = \vec{e}_1 ^T \rightarrow \vec{e}_2 ^T \rightarrow P \vec{e}_1 \\
\Rightarrow \begin{cases}
P \cdot \left( \vec{e}_1 ^T \cdot a \right) \text{ will give the right result} \\
P \cdot \left( \vec{e}_2 ^T \cdot a \right) \text{ will give the wrong result}
\end{cases}
\]

Without homogenous coordinates, this was not required.

Note: It is not clear at this moment if this requirement makes the term “tensor” invalid for this case. The calculations described below do work properly however.

Another implication of the use of homogenous coordinate is that the CS transformation tensors (from CS1 to CS0, and vice versa) are not the same as the unity tensor (as it was without using homogenous coordinates).

In the following, the space will be 3-dimensional again making the homogenous matrices and vectors 4-dimensional.
The tensor to be used to find the components in CS0, when they are known in CS1 to CS0 is given by:

\[
\text{Eqn. 157 } P = \text{Trans}(p_1, p_2) \text{Rot}(\vec{e}_3, \varphi) = \begin{bmatrix}
c\varphi & -s\varphi & 0 & p_1 \\
s\varphi & c\varphi & 0 & p_2 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}
\]

Etn 158

\[
P^\top = \begin{bmatrix} 0 & 0 & 0 & -p_2 \cdot c\varphi - p_1 \cdot s\varphi \\
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}
\]

The tensor to be used to find the components in CS1, when they are known in CS0 is given by:

\[
\text{Eqn. 159 } P = \text{Rot}(\vec{e}_3, -\varphi) \text{Trans}(-p_1, -p_2) = \begin{bmatrix}
c\varphi & s\varphi & 0 & -p_1 \cdot c\varphi - p_2 \cdot s\varphi \\
s\varphi & c\varphi & 0 & p_1 \cdot s\varphi - p_2 \cdot c\varphi \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
P^{-1} = \begin{bmatrix} 0 & 0 & 0 & -p_1 \cdot c\varphi - p_2 \cdot s\varphi \\
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}
\]

\[
\text{Eqn. 160 } P^{-1} = \text{Rot}(\vec{e}_3, -\varphi) \text{Trans}(-p_1, -p_2) = \begin{bmatrix}
c\varphi & s\varphi & 0 & -p_1 \cdot c\varphi - p_2 \cdot s\varphi \\
s\varphi & c\varphi & 0 & p_1 \cdot s\varphi - p_2 \cdot c\varphi \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

If the vector interpretation is intended for the transformation, the 4th component should be taken equal to 0 (thereby discarding the influence of the translation of the origin). If point interpretation is intended, the 4th component should be made equal to 1.

For correct calculation of the time derivatives, a transformation tensor must be found to express CS1 in CS0. This transformation tensor will be:

\[
\text{Eqn. 161 } T = \text{Trans}(p_1, p_2) \text{Rot}(\vec{e}_3, \varphi) = \begin{bmatrix}
c\varphi & s\varphi & 0 & -p_1 \cdot c\varphi - p_2 \cdot s\varphi \\
s\varphi & c\varphi & 0 & p_1 \cdot s\varphi - p_2 \cdot c\varphi \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
\text{Eqn. 162 } T^{-1} = \begin{bmatrix} 0 & 0 & 0 & -p_1 \cdot c\varphi - p_2 \cdot s\varphi \\
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}
\]

\[
\text{Eqn. 163 } \vec{e}^T = T \cdot \vec{e}^T
\]

CS1 can now be expressed in CS0 (see Eqn. 154):

Again, like the 1-DOF case, other useful tensors can be derived from the previous ones:

\[
\text{Eqn. 164 } T^{-1} = \text{Rot}(\vec{e}_3, -\varphi) \text{Trans}(-p_1, -p_2) = \begin{bmatrix}
c\varphi & s\varphi & 0 & -p_1 \cdot c\varphi - p_2 \cdot s\varphi \\
s\varphi & c\varphi & 0 & p_1 \cdot s\varphi - p_2 \cdot c\varphi \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
\text{Eqn. 165 } T^{-1} = \begin{bmatrix} 0 & 0 & 0 & -p_1 \cdot c\varphi - p_2 \cdot s\varphi \\
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}
\]

\[
\text{Eqn. 166 } T^{-1} = \begin{bmatrix}
c\varphi & s\varphi & 0 & -p_1 \cdot c\varphi - p_2 \cdot s\varphi \\
s\varphi & c\varphi & 0 & p_1 \cdot s\varphi - p_2 \cdot c\varphi \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
\text{Eqn. 167 } \vec{e}^T = T \cdot \vec{e}^T
\]

For the general time derivatives \( \varphi, p_1 \) and \( p_2 \) are all assumed to be some function of \( t \). The first time derivative of \( T \) will be:

\[
\text{Eqn. 168 } \frac{dT}{dt} = \begin{bmatrix}
-\dot{\varphi} \cdot s\varphi & -\dot{\varphi} \cdot c\varphi & 0 & \dot{p}_1 \\
\dot{\varphi} \cdot c\varphi & -\dot{\varphi} \cdot s\varphi & 0 & \dot{p}_2 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{bmatrix}
\]

\[
\text{Eqn. 169 } \frac{dT}{dt} = \begin{bmatrix} 0 & 0 & 0 & -\dot{p}_1 \cdot c\varphi - \dot{p}_2 \cdot s\varphi \\
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}
\]

\[
\text{Eqn. 170 } \frac{dT}{dt} = \begin{bmatrix} 0 & 0 & 0 & -\dot{p}_1 \cdot c\varphi - \dot{p}_2 \cdot s\varphi \\
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}
\]
The second time derivative of $T$ will be:

$$\ddot{T} = \begin{bmatrix} -c\dot{\phi}^2 - s\dot{\phi}\dot{\phi} & s\dot{\phi}\dot{\phi} - c\dot{\phi}^2 & 0 & \ddot{\phi}_1 \\ c\dot{\phi}^2 - s\dot{\phi}\dot{\phi} & -c\dot{\phi}^2 - s\dot{\phi}\dot{\phi} & 0 & \ddot{\phi}_2 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Eqn. 168

$$\ddot{T} = \begin{bmatrix} -c\dot{\phi}^2 - s\dot{\phi}\dot{\phi} & s\dot{\phi}\dot{\phi} - c\dot{\phi}^2 & 0 & \ddot{\phi}_1 \\ c\dot{\phi}^2 - s\dot{\phi}\dot{\phi} & -c\dot{\phi}^2 - s\dot{\phi}\dot{\phi} & 0 & \ddot{\phi}_2 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Eqn. 169

Obtaining the combined first- and second order time derivative tensors is a bit more complicated than without homogenous coordinates. The (unity) tensor that was added in Eqn. 74 and Eqn. 77 to make the calculation easier is not equal to $0^e_{\tau}[T^0_{e-1}]_e$ this time. The reason for this is explained in Eqn. 155. It is however taken to be equal to the tensor $P$. This means the point described by the vector the combined derivative tensor is working on will first be transformed to CS0 by $P$ (by using point interpretation, the 4th component of the vector must therefore be 1).

The tensor that was added in Eqn. 38 and Eqn. 39 to gain a result relative to CS1 is $P^{-1}$.

As explained before, the vector interpretation must be used while transforming the speed and acceleration vectors, not the point interpretation. Due to the zeroes in the last row in the matrices in Eqn. 166 and Eqn. 168, this will happen automatically (by making the 4th component equal to zero).

With homogenous coordinates, the combined first order time derivative tensors, similar to Eqn. 74 will be:

$$\dot{\mathbf{e}} = \begin{bmatrix} -\dot{\phi} \cdot s\phi & -\dot{\phi} \cdot c\phi & 0 & \dot{\phi}_1 \\ \dot{\phi} \cdot c\phi & -\dot{\phi} \cdot s\phi & 0 & \dot{\phi}_2 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{P}$$

Eqn. 170

$$\mathbf{F} = \dot{T} \cdot T^{-1} \cdot P = \begin{bmatrix} -c\dot{\phi}^2 - s\dot{\phi}\dot{\phi} & s\dot{\phi}\dot{\phi} - c\dot{\phi}^2 & 0 & \ddot{\phi}_1 \\ c\dot{\phi}^2 - s\dot{\phi}\dot{\phi} & -c\dot{\phi}^2 - s\dot{\phi}\dot{\phi} & 0 & \ddot{\phi}_2 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Eqn. 171

The combined second order time derivative tensors, similar to Eqn. 77 will be:

$$\ddot{\mathbf{e}} = \begin{bmatrix} -\dot{\phi}^2 & -\dot{\phi} \cdot \ddot{\phi} & 0 & c\phi \cdot \dot{\phi}_1 + s\phi \cdot \dot{\phi}_2 \\ c\dot{\phi} \cdot \ddot{\phi} & -c\dot{\phi}^2 - s\dot{\phi}\dot{\phi} & 0 & c\phi \cdot \dot{\phi}_2 - s\phi \cdot \dot{\phi}_1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{P}$$

Eqn. 172

$$\mathbf{S} = \ddot{T} \cdot T^{-1} \cdot P = \begin{bmatrix} -c\phi^2 - s\phi\phi \cdot \ddot{\phi} & s\phi \cdot \ddot{\phi} - c\phi^2 - c\phi \cdot \phi \cdot \ddot{\phi} & 0 & \ddot{\phi}_1 \\ c\phi^2 - s\phi\phi \cdot \ddot{\phi} & -c\phi^2 - s\phi\phi \cdot \ddot{\phi} & 0 & \ddot{\phi}_2 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Eqn. 173

And finally similar to Eqn. 68 to Eqn. 71, the first and second time derivatives of $\ddot{\mathbf{a}}$ can be calculated.

To get more general applicable equations $\ddots\mathbf{a}$ is not presumed to be constant.
The derivatives in vector notation become:

\[ \frac{\partial \vec{a}}{\partial t} = \frac{1}{2} \sum_{i,j} \left( \frac{\partial^2 T^{i}}{\partial \vec{e}_i \partial \vec{e}_j} \right) \vec{a}_j + \frac{1}{2} \sum_{i,j} \left( \frac{\partial^2 T^{j}}{\partial \vec{e}_i \partial \vec{e}_j} \right) \vec{a}_i \]

By using tensors (the same way as with Eqn. 80 and Eqn. 81), the former equations can be written more conveniently as:

\[ \vec{A} = F \cdot \left( \vec{e} \cdot A \right) + P \cdot \left( \vec{e} \cdot A \right) \]

It may seem strange that the addition of tensor \( P \), which is not equal to the unit tensor, does not change the result. However the derivatives of the component column of \( \vec{A} \) will have a zero as the 4th component due to the derivative of the constant \( I \). This means the vector interpretation must be used, effectively making the translational part of \( P \) useless; it can be made equal to zero without problems. Tensor \( P \) with zero translational part is however equal to the unity tensor. This is why in this case the tensor \( P \) may be added to this specific vector without changing the result.

In matrix notation, these results (expressed both in CS0 and CS1) will be:

\[ \begin{align*}
\vec{A} = 0 \cdot e^{T} \cdot A & = 0 \cdot e^{T} \cdot 1_5 \cdot e \cdot A + \sum_{i,j} P_{ij} \cdot e^{T} \cdot A \\
\vec{A} = 1 \cdot e^{T} \cdot A & = 1 \cdot e^{T} \cdot 1_5 \cdot e \cdot A + \sum_{i,j} P_{ij} \cdot e^{T} \cdot A \\
\vec{A} = 0 \cdot e^{T} \cdot A & = 0 \cdot e^{T} \cdot 1_5 \cdot e \cdot A + \sum_{i,j} P_{ij} \cdot e^{T} \cdot A \\
\vec{A} = 1 \cdot e^{T} \cdot A & = 1 \cdot e^{T} \cdot 1_5 \cdot e \cdot A + \sum_{i,j} P_{ij} \cdot e^{T} \cdot A \\
\vec{A} = 0 \cdot e^{T} \cdot A & = 0 \cdot e^{T} \cdot 1_5 \cdot e \cdot A + \sum_{i,j} P_{ij} \cdot e^{T} \cdot A \\
\vec{A} = 1 \cdot e^{T} \cdot A & = 1 \cdot e^{T} \cdot 1_5 \cdot e \cdot A + \sum_{i,j} P_{ij} \cdot e^{T} \cdot A \\
\end{align*} \]

Keep in mind that when \( \vec{A} \) is constant, for the final result of Eqn. 180 to Eqn. 183 only the first term will be non-zero.
### Comparison of various state estimation techniques

<table>
<thead>
<tr>
<th>Art.</th>
<th>Model</th>
<th>Observer</th>
<th>Sensors</th>
<th>Vehicle</th>
<th>Tests</th>
<th>Author</th>
<th>Purpose</th>
<th>Future</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>[9]</td>
<td>4 Wheel, 2DOF front steering, dynamic model</td>
<td>(extended) Luenberger</td>
<td>1,2,3,4,5,6,7,8</td>
<td>4 wheel car, front steering</td>
<td>Real-life, few reference measurements.</td>
<td>A. Daif, Karlsruhe Univ.</td>
<td>ABS development</td>
<td>RLS estim. must be extended for cornering. Design of control algorithm</td>
<td>Not everything is described, or finished in this article</td>
</tr>
<tr>
<td>[10]</td>
<td>Bicycle, 2DOF, front &amp; rear steering, Kinematic model</td>
<td>Kalman filter</td>
<td>7,8,9,10</td>
<td>4 wheel car, front &amp; rear steering</td>
<td>Real life, constantly changing steering angle</td>
<td>Y. Hirano, Toyota Motor Corp.</td>
<td>ARS development</td>
<td>Frequency domain loop shaping control</td>
<td>Also describes model following control. i.e. cornering stiffness values are required</td>
</tr>
<tr>
<td>[11]</td>
<td>Bicycle, 5DOF, front steering, dynamic model</td>
<td>Extended Kalman filter</td>
<td>1,2,3,5,7</td>
<td>Observer+measurement noise compared to 9DOF model</td>
<td></td>
<td>L.R. Ray, Christian Brothers Univ.</td>
<td>General advanced vehicle control, tire force modeling</td>
<td>No future developments given</td>
<td>Also describes simple PI slip est. feedback control. Estimates seem accurate</td>
</tr>
<tr>
<td>[12]</td>
<td>4 Wheel, 8DOF, front steering, dynamic model</td>
<td>Hybrid Extended Kalman filter</td>
<td>1,2,3,4,5,6,7,16</td>
<td>4 wheel car, front steering</td>
<td>Extensive real-life tests, different maneuvers</td>
<td>L.R. Ray, Christian Brothers Univ.</td>
<td>General advanced vehicle control, tire force modeling</td>
<td>More tests, with more maneuvers and different road types</td>
<td>Friction coefficient different from inspected appears to introduce slip estimate offset.</td>
</tr>
<tr>
<td>[13]</td>
<td>4 Wheel, 4DOF, front steering, kinematic model</td>
<td>Extended adaptive Kalman filter</td>
<td>1,2,11,12,13</td>
<td>Observer+measurement noise compared to 6DOF + Pacejka tire model</td>
<td></td>
<td>M.C. Best, Loughborough Univ.</td>
<td>General vehicle control, and chassis design improvements</td>
<td>Improved tire modeling, and various other improvements</td>
<td>Side slip estimate accuracy in this article is 20-30% RMS (relative to signal)</td>
</tr>
<tr>
<td>[14]</td>
<td>Bicycle, 2DOF, dynamic model</td>
<td>Combination of direct integration and observer</td>
<td>7,14,15,10</td>
<td>4 wheel car, front steering</td>
<td>Some real life tests.</td>
<td>Y. Fukada, Toyota Motor Corp.</td>
<td>Yaw-moment control, for improved vehicle handling</td>
<td>No Future development given. but this estimation is used in Toyota’s VSC since 1997</td>
<td>With a J-turn maneuver, the estimates show quite large deviations, but apparently this is good enough for use in VSC.</td>
</tr>
<tr>
<td>[15]</td>
<td>Bicycle, 2DOF, front steering, kinematic model</td>
<td>Adaptive Kalman filter</td>
<td>7,8,10</td>
<td>4 wheel car, front steering</td>
<td>Real life tests</td>
<td>M. Kaminaga, Nissan Motor Corp.</td>
<td>General vehicle control</td>
<td>No future developments given</td>
<td>Robustness is very good according to article</td>
</tr>
</tbody>
</table>

### Sensor Description

<table>
<thead>
<tr>
<th>Sensor</th>
<th>Description</th>
<th>Sensor</th>
<th>Description</th>
<th>Sensor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Longitudinal acceleration (in centre of gravity)</td>
<td>7</td>
<td>Yaw rate</td>
<td>13</td>
<td>Lateral acceleration (certain distance above rear axle)</td>
</tr>
<tr>
<td>2</td>
<td>Lateral acceleration (in centre of gravity)</td>
<td>8</td>
<td>Front steering angle</td>
<td>14</td>
<td>Front wheel side force</td>
</tr>
<tr>
<td>3</td>
<td>Front left wheel rotational speed</td>
<td>9</td>
<td>Rear steering angle</td>
<td>15</td>
<td>Rear wheel side force</td>
</tr>
<tr>
<td>4</td>
<td>Front right wheel rotational speed</td>
<td>10</td>
<td>Vehicle speed</td>
<td>16</td>
<td>Roll rate</td>
</tr>
<tr>
<td>5</td>
<td>Rear left wheel rotational speed</td>
<td>11</td>
<td>Lateral acceleration (in front axle)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Rear right wheel rotational speed</td>
<td>12</td>
<td>Lateral acceleration (in rear axle)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bibliography

Literature used in text

Literature found in literature survey
C. Canudas de Wit, A.D. Ndouidi-Likoho; Institut National Polytechnique de Grenoble
"Nonlinear control for a convoy-like vehicle"
In: Automatica, Vol. 36, no.3
March 2000, Pages 457-462

T. Fujioka, M. Omae; University of Tokyo
"Vehicle following control in lateral direction for platooning"
In: Vehicle system dynamics supplement 28
1998. Pages 422-437

S. Saraf, M. Tomizuka; University of California
"Slip angle estimation for vehicles on automated highways"
In: Proc. Of the American control conference, albuquerque, New Mexico
June 1997. Pages 1588-1592

S. J. Gehr, F. J. Stein; Research Institute Daimler Benz AG
"A trajectory-based approach for the lateral control of car following systems"
In: IEEE International conf. on systems, man, and cybernetics, 11-14 oct 1998, Vol 4
1998. Pages 3596-3601

"The development of vehicle stability control at Ford"
In: IEEE/ASME transactions on mechatronics, Vol 4 no.3
September 1999. Pages 223-234

A. Modjathedzadeh, R. A. Hess; California University
"A model of driver steering control behavior for use in assessing vehicle handling qualities"
In: Trans. of the ASME journal of dynamic systems, measurement and control, Vol 115 no.3
September 1993. Pages 456-464

M. S. Habib; ?
"A predictive control model of driver steering behavior"
In: Road vehicle automation II towards systems integration. Proceedings of the 2nd Intl. Conf. on road vehicle automation;
11-13 sept 1995 Bolton
1997, Pages 125-132

H. Takahashi; Nissan Motor Corp. Ltd.
"A study on designing intelligent vehicle using driving environment-driver-vehicle interaction model "
In: Proc. of the 5th Intl. Conf. on soft computing and information/intelligent systems. Methodologies for the conception, design and application of soft computing. 16-20 Oct 1998 Vol. 1
1998, Pages 221-226

K. W. Ahn, Won-Gul Hwang; Asia Motors / Chonam National University
"A study on handling characteristics of a medium bus during a single lane change maneuver"
In: Autumn Conference, 25 nov. 1994, Korea. Published by KSAE
1994

A. Honguchi, T. Suemori; Mazda Motor Corp.
"A study on the human perception of yaw motion using the driving simulator"
In: Intl. Symposium on advanced vehicle control, 14-17 sept 1992. Published by JSAE
1992

J. B. Ferrer, Daimler Chrysler Corp.
"Factors affecting perceptions of ride quality in automobiles"
In: Dynamic systems and control division. The ASME int'l. Mechanical engineering congress and exposition, Nashville 14-19 nov 1999, Vol. 67
1999. Pages 649-654

J. H. Hogema; TNO human research inst.
"Modeling motorway driving behavior"
In: Transportation research record. No. 1689
Nov 1999. Pages 25-32

M. Immscher, T. Jurgensohn, H. P. Willumeit;
"Driver models in vehicle development"
In: Vehicle system dynamics, Vol 33 supplemental issue.
1999. Pages 83-93