Online Vehicle Inertial Parameter Estimation

Testing Rozyn's Algorithm Under More Realistic Conditions

J.C. Dijkhuizen - 4225457

Master of Science Thesis

TU Delft Delft University of Technology

Department of Cognitive Robotics - Vehicle Engineering

Online Vehicle Inertial Parameter Estimation

Testing Rozyn's Algorithm Under More Realistic Conditions

Master of Science Thesis

For the degree of Master of Science in Vehicle Engineering at Delft University of Technology

J.C. Dijkhuizen - 4225457

August 13, 2019

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Abstract

The inertial parameters of a vehicle, which include the mass, centre of gravity position and the moments of inertia, influences the dynamics of the vehicle. Currently, the modelling of the vehicle is done by assuming fixed, conservative, values for the inertial parameters. Knowing the exact values may increase the performance, safety and comfort of the vehicle. A literature review has been conducted [1], where different methods for online inertial parameter estimation have been graded based on the amount of parameters it is able to estimate, the sensors used and the accuracy of the methods. Rozyn's method seems best for online inertial parameter estimation. Rozyn proposes a method which can estimate the inertial parameters from vertical acceleration data using a state variable method, modal analysis and a simple vehicle model. Rozyn's method can be summarised in four steps:

1. Extract the free decay response from acceleration data.
2. Construct the state transition matrix.
3. Construct the system characteristic matrix.
4. Estimate the inertial parameters using the constructed characteristic matrix and simplified vehicle model.

The main shortcoming of Rozyn's method is the road profile which is used for the simulation, which is described in the ISO 8608 norm. The ISO 8608 description is a stationary Gaussian process. This means that the road profile random variables are normally distributed. Furthermore, the properties of the road profile (mean and variance) does not change over time. In practice however, road profiles never follow a stationary Gaussian process, but are much more random, with more variance between different sections. Another, more realistic, road profile description is proposed by Bogsjö where the road profile follows a non-stationary Laplace distribution. Another shortcoming of Rozyn's paper is that it only shows results for only one condition. For the simulation, the vehicle is driving 100 km/h and a measurement period of 1,000 seconds is used. Unknown is the influence of the velocity of the vehicle on the results. It is to be expected that the accuracy decreases for shorter measurement periods, but by how much is also unknown.

In this thesis, Rozyn's algorithm is explained and implemented using a half car vehicle model. Rozyn's algorithm is validated using the ISO 8608 road profile description on similar conditions. The algorithm is then tested using the ISO 8608 road profile description where the velocity of the vehicle is varied between 30 and 100 km/h and the measurement periods between 30 and 120 seconds. This is done 100 times for each condition. This results in 100 estimates of the inertial parameters of each condition. From these results, the average and standard deviation between the estimates can be calculated. This is also done for the alternative Laplace road description. The resulting standard deviations are plotted in surface plots, as function of the varying velocity and measurement period.

The results show that the standard deviation between the different estimated parameters when using the Laplace description for the road profile are up to 5 times higher compared to the ISO 8608 road profile description. The results also show that the performance of the algorithm is heavily dependent on the measurement time. A measurement time of at less then 60 seconds is not recommended, due to the large deviation in the estimated parameters. For the mass and centre of gravity position, the performance is independent of the velocity of the vehicle. However, the pitch moment of inertia shows a slight dependency on the velocity, with lower deviations between the different estimates for higher velocities.

The algorithm can still be used on non-stationary road profiles. However, more and longer measurements are needed for the algorithm to return with an accurate estimation of the inertial parameters. Even then, some errors in the estimated parameters in the order of 10% are present.

Table of Contents

Preface

Before you lies my thesis about Online Vehicle Inertial Parameter Estimation. It is a subject that interests me as a future Vehicle Engineer. The inertial parameters of a vehicle are never constant, and I was interested which online methods exists for estimation of these parameters. This started with a literature study where I read many papers regarding this subject. This turned out to be an exhausting process, but it resulted in a clear conclusion: That it is very difficult to estimate the inertial parameters using an online method. Although Rozyn proposes a method which seems best for this task, it was not easy to implement it in Python, since the equations and assumptions used were not well documented. But, eventually, after some hard work I got a working algorithm, which was able to estimate the inertial parameters.

I would like to thank my supervisor Prof.dr.ir. Martijn Wisse for his guidance and support during the process. Also, I would like to thank my friends from the UNO Club, Robert Cornet, Alexander van Doeveren, Mounir El Hassnaoui, Gowtham Nagalingam and Quinn Vroom. I always looked forward to playing a few games of UNO during the lunch break. Thanks to them, it was never boring. I also want to thank my family and the rest of my friends for their support.

I hope you enjoy reading my thesis.

Joël Dijkhuizen

Delft, University of Technology
August 13, 2019

Chapter 1

Introduction

In the last decades, many new active safety systems, such as ABS (Anti-lock Braking System) and ESP (Electronic Stability Program) have been introduced. These systems increase the safety on the road and decrease the number of fatalities. Knowing the exact dynamics of a vehicle is very important for the safety and comfort of the vehicle. Currently, the determination of the vehicle dynamics is done by extensive modelling of the vehicle. However, determining the dynamics is a very time consuming and expensive procedure since it requires full insight in the properties of the vehicle. All of the systems controlling the car are sensitive to the vehicle inertial parameters, such as vehicle weight, position of the center of gravity and moments of inertia [2]. But what if these properties are unknown? Determining the parameters of the vehicle online might reduce the time and cost needed to determine the dynamics of the vehicle. Furthermore, these vehicle inertial properties can change over time, due to change in load or fuel usage [3]. Knowing the exact value of the parameter can increase the effectiveness of a mid-level controller, such as ABS [4].

A literature review has been conducted about this subject [1]. The online inertial parameter estimation can be estimated using different dynamics of the vehicle, namely the longitudinal, the lateral and the suspension dynamics. For the longitudinal dynamics, it is necessary to estimate all the forces acting in the longitudinal plane of the vehicle, which include the aerodynamic drag, the rolling resistance, the engine force and the force due to a road grade. This influences the estimation accuracy of the inertial parameters of the vehicle [5]. For the lateral dynamics, the tyre dynamics are used. This complicates the problem, since the tyres road interaction changes over time, based on the road surface, weather, temperature, wear of the tyres, etc. In the literature, these variables are assumed constant, which influences the estimation of the inertial parameters [4]. The suspension dynamics are also used in literature to estimate the inertial parameters using suspension displacement sensors [6] or using accelerometers in the body of the vehicle [4]. The literature has been graded based on the number of parameters which can be estimated using the methods, the sensors used and the accuracy of the proposed methods. Here, Rozyn scores the best with an algorithm that is able to estimate all the necessary inertial parameters using cheap and easy to use sensors with an high accuracy of the estimated inertial parameters.

The method proposed by Rozyn et al [4] is able to obtain the vehicle's unknown inertial parameters, which include the mass, position of centre of gravity and the moments of inertia. This is done by measuring the sprung mass response when the vehicle is excited by an unknown and unmeasured road profile. From these measurements the free decay response of the vehicle is extracted. This is followed by modal analysis to estimate the vehicle's sprung mass natural frequencies, damping ratios and mode shapes. This information can be used to create an estimation of the vehicle's characteristic matrix. This matrix is then compared to the vehicle's characteristic matrix obtained from a simplified vehicle model followed by a least squares analysis to obtain the vehicle's inertial parameters.

The road profile that is used by Rozyn is defined in the ISO 8608 standard, however it is not suitable for the description for longer road profiles [7]. A more realistic road profile description is provided by Bogsjö [8]. Furthermore, Rozyn only gives result for the algorithm using a measurement period of 1,000 seconds and a velocity of 100 km/h. Unknown is the influence of these parameters on the performance of the algorithm. This leads to the following research question:

How does the method proposed by Rozyn, for online vehicle inertial parameter estimation, perform under more realistic conditions?

This is tested by implementing the two road profile description, and testing Rozyn's algorithm on both road profiles for measurement periods between 10 and 120 seconds and also by varying the velocity between 30 and 100 km/h. This is done by estimating the parameters on above mentioned conditions 100 times for each condition using 100 different, randomly created, road profiles. This results in 100 estimates of the inertial parameters each condition. The results will show the standard deviation of the estimates plotted against the velocity and measurement period.

Chapter 2

Rozyn's Method

Rozyn et al proposes a method where the inertial parameters are estimated, based on vertical acceleration data of the vehicle. These sensors are place in the body of the vehicle, on the corners. The algorithm extracts the free-decay response of the vehicle from this acceleration data using the autocorrelation function. From this free-decay response, the state transition matrix is estimated. The system characteristic matrix can be estimated, by using modal analysis. This characteristic matrix is then compared to a simple vehicle model, where a error minimisation scheme is used to estimate the inertial parameters.

Rozyn's description of the algorithm is very brief and not all the equations necessary are mentioned. Two papers by Zhang, [9] and [10], are used to implement the algorithm.

2-1 Extracting Free-Decay Response

The input of the algorithm is a free-decay response of the body of the vehicle. Free-decay, also called free vibration response, is the response of the vehicle when the system is given an initial condition and is allowed to vibrate freely. If the vehicle is excited by a random road profile, the acceleration data contains the accelerations due to excitation of this road profile. The free-decay response is hidden in this data and can be extracted using an autocorrelation function. The autocorrelation function calculates the correlation between one time point of the data and the other, which is equivalent to the free decay response, in the case of a system that is excited by stationary Gaussian white noise [11], [12]. The equation to calculate the autocorrelation function can be seen in equation 2-1.

$$R_{yy}(\tau) = \int_0^\infty y(t) \cdot y(t+\tau)dt \tag{2-1}$$

One requirement for the autocorrelation function is that the process, on which it is applied, is stationary [13]. The road surface can be seen as a wave. The propagation of the wave is the velocity of the vehicle. The condition that the wave must be stationary means that the free-decay response of the vehicle can only be determined accurately if the vehicle is driving at a constant velocity. Also the road surface should be stationary, which is not always the case in real life. These assumptions leads to inaccuracies of the extracted free decay responses if the road surface is not a stationary and Gaussian distributed process.

2-2 Estimating State Transition Matrix

The state transition matrix contains the characteristics of the system in continuous and discrete time [10]. Since the measurements are performed at fixed time resolutions, the system is represented in discrete time as follows:

$$X(k+1) = A_1 \cdot X(k) \tag{2-2}$$

With A_1 as the state transition matrix. In this equation, the discrete state vectors are sampled at time t = kT, where T is the sampling interval. The modal parameters of the system can be obtained by solving the corresponding eigenvalue problem. However, the measured signals will always contain some errors. To identify these errors, the size of the state transition matrix is increased by adding pseudo measurements.

$$X(k) = \begin{bmatrix} Y^T(k) & Y^T(k+1) & \cdots & Y^T(k+p) \end{bmatrix}^T \tag{2-3}$$

Where Y are the sensor measurements and $p = \gamma \cdot p_0$. In this equation, γ is a tuning parameter based on the signal to noise ratio, in the case of a higher signal to noise ratio, γ is selected as $4 \sim 6$. Since no noise is present in the simulation, γ is chosen as 2. This results in the best estimates of the inertial parameters. p_0 is a parameter based on the system and can be calculated as follows:

$$p_0 = \frac{2n}{m} \tag{2-4}$$

Where n is the degrees of freedom of the system and m the measured outputs. In this case, where a half car model is used for the simulation (see Chapter 3), the degrees of freedom of the system is four, namely the vertical position of the the wheels of the vehicle and the vertical position and pitch angle of the body. The number of measurement sensors is two, the acceleration of the front of the body and the acceleration of the rear. This makes p equal to:

$$p = \gamma \cdot \frac{2n}{m} = 6 \cdot \frac{2 \cdot 4}{2} = 24 \tag{2-5}$$

The transition matrix A can be obtained by using the following state matrices:

$$\begin{aligned} \Phi &= \begin{bmatrix} X(1) & X(2) & \cdots & X(M) \end{bmatrix} \\ \overline{\Phi} &= \begin{bmatrix} X(2) & X(3) & \cdots & X(M+1) \end{bmatrix} \end{aligned} \tag{2-6}$$

Here, M is the number of data points in the measurements. However, this means that it uses M + p + 1 data points to calculate the state transition matrix, while only N data points are available. This means that the variable M should be calculated as follows:

$$M = N - p - 1 \tag{2-7}$$

The two state matrices must satisfy the following equation:

$$\overline{\Phi} = A_1\Phi + \widetilde{\Phi} \tag{2-8}$$

The state transition matrix A_1 can then be determined by using least squares, as follows:

$$A_1 = (\overline{\Phi}\Phi^T)(\Phi\Phi^T)^{-1} \tag{2-9}$$

2-3 Estimating System Characteristic Matrix

In continuous time, the system can be described in the form of the state equation:

$$\dot{X} = A_0 \cdot X \tag{2-10}$$

Here, A_0 is the system characteristic matrix which relates the mass, stiffness and damping matrices as follows:

$$A_0 = \begin{bmatrix} 0 & I \\ -M^{-1}K & -M^{-1}C \end{bmatrix} \tag{2-11}$$

The state characteristic matrix can be calculated by solving the eigenvalue problem of the state transition matrix. This yields the the eigenvalues of the system in the complex, discrete, Z-plane. Since state transition matrix A_1 has the size of $2 \cdot (p+1)$ by $2 \cdot (p+1)$, this will result in $2 \cdot (p+1)$ eigenvalues and corresponding eigenvectors. The system characteristic matrix A_0 is, however, defined in the continuous S-plane as in equation 2-11. This means that the eigenvalues must be converted to the S-plane. The relation between the Z-plane and the S-plane is as follows: [10]

$$Z = e^{S \cdot T} \tag{2-12}$$

Where Z and S are defined as follows:

$$Z = a + jb \tag{2-13}$$

$$S = \alpha + j\beta = -\zeta\omega + j\omega(1-\zeta^2)^{1/2} \tag{2-14}$$

Where the real and imaginary part of the eigenvalues in the S-plane are expressed as:

$$\alpha = \frac{ln(a^2+b^2)}{2 \cdot T} \tag{2-15}$$

$$\beta = \frac{tan^{-1}\left(\frac{b}{a}\right)}{T} \tag{2-16}$$

The damped and undamped natural frequencies and the damping ratios can be calculated as follows:

$$\omega_n = \sqrt{\alpha^2 + \beta^2} \tag{2-17}$$

$$\omega_d = \beta \tag{2-18}$$

$$\zeta = -\frac{\alpha}{\omega_n} \tag{2-19}$$

The system will have two natural modes, the front of the body of the vehicle and the rear. The estimation however has $2 \cdot (p+1)$ modes. This means the noise modes will have to be found and disregarded. This is done by looking at several known characteristics of the system. A car will be underdamped, this means it oscillates with an amplitude that gradually decreases to zero. An underdamped system will have eigenvalues that appear in complex conjugates [14]. This means that all non-imaginary eigenvalues can be removed from the list. Furthermore, the frequencies must lie within a certain range. It is known, for instance that the bounce frequency will lie within the 1-3 Hz range [15] and are underdamped with a damping loss factor between 0.1 and 0.5. Every frequency and damping loss factor outside this range can be assumed to be a noise mode and can thus be disregarded. The corresponding eigenvectors are generated by normalizing the first m, two in the case of the half-car model, elements of the eigenvector.

The estimated system characteristic matrix can then be written as follows [9]:

$$A_0 = \begin{bmatrix} \psi & \psi^* \\ \psi\lambda & \psi^*\lambda^* \end{bmatrix} \begin{bmatrix} \lambda & 0 \\ 0 & \lambda^* \end{bmatrix} \begin{bmatrix} \psi & \psi^* \\ \psi\lambda & \psi^*\lambda^* \end{bmatrix}^{-1} \tag{2-20}$$

Where λ denotes the eigenvalues and ψ the eigenvectors, the complex conjugate is indicated with a *.

2-4 Estimating Inertial Parameters

Now the system characteristic matrix is constructed from the measurements, it can be compared to the system characteristic matrix with some known parameters. Assumed is that the stiffness and damping parameter are known. With this assumption, a characteristic matrix is constructed from a simplified vehicle model, see Figure 2-1, where the unsprung masses and the springs and dampers have been combined into one spring/damper with an equivalent stiffness/damping coefficient which can be calculated as follows:

$$k_{eq} = \frac{k_s \cdot k_t}{k_s + k_t} \tag{2-21}$$

According to Rozyn, this equation is not accurate enough due to the presence of non-proportional damping. Rozyn uses an equation for the equivalent stiffness which depends on the frequency in which it oscillates and also the damping coefficients, as seen in equation 2-22. However, since no non-proportional damping is present in the simulation model, equation 2-21 will not add any errors to the estimation of the parameters. If the algorithm is implemented in a real vehicle non-proportional damping characteristics might be present. In this case equation 2-21 might result in errors and the equivalent stiffness as calculated in equation 2-22 will be more suited to this case.

$$k_{eq} = \frac{k_s k_t^2 + k^2 s k_t + \omega(k_s c_t^2 + k_t c_s^2)}{(k_s + k_t)^2 + \omega^2 (c_s + c_t)^2} \tag{2-22}$$

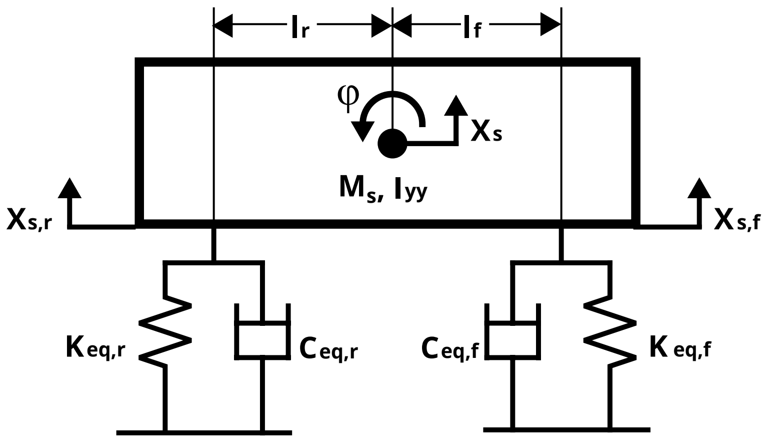

Figure 2-1: Equivalent half car model with as generalised coordinates x_s and φ.

The corresponding equations to the equivalent half car model can be written as follows:

$$m_s \cdot \ddot{x}_s = -k_{eq,f} \cdot x_{s,f} - c_{eq,f} \cdot \dot{x}_{s,f} - k_{eq,r} \cdot x_{s,r} - c_{eq,r} \cdot \dot{x}_{s,r} \tag{2-23}$$

$$I_{yy} \cdot \ddot{\varphi} = -l_f \cdot (k_{eq,f} \cdot x_{s,f} + c_{eq,f} \cdot \dot{x}_{s,f}) + l_r \cdot (k_{eq,r} \cdot x_{s,r} + c_{eq,r} \cdot \dot{x}_{s,r}) \tag{2-24}$$

Where $x_{s,f}$ and $x_{s,r}$ can be written as follows, according to the linearised equations:

$$x_s = \frac{l_r \cdot x_{s,f} + l_f \cdot x_{s,r}}{L} \tag{2-25}$$

$$\varphi = \frac{x_{s,f} - x_{s,r}}{L} \tag{2-26}$$

For these equations, small angles for the pitch angles are assumed to linearize the equation. This can be done with no problem, since the pitch angle of the vehicle driving over a road is very small.

The corresponding matrices of the system characteristic matrix are as follows:

$$A_0 = \begin{bmatrix} 0 & I \\ -M^{-1}K & -M^{-1}C \end{bmatrix} \tag{2-27}$$

$$M = \begin{bmatrix} m_s \cdot \frac{l_r}{L} & m_s \cdot \frac{l_f}{L} \\ \frac{I_{yy}}{L} & -\frac{I_{yy}}{L} \end{bmatrix} \tag{2-28}$$

$$K = \begin{bmatrix} k_{eq,f} + k_{eq,r} & l_f \cdot k_{eq,f} - l_r \cdot k_{eq,r} \\ l_f \cdot k_{eq,f} - l_r \cdot k_{eq,r} & l_f^2 \cdot k_{eq,f} + l_r^2 \cdot k_{eq,r} \end{bmatrix} \tag{2-29}$$

$$C = \begin{bmatrix} c_{eq,f} + c_{eq,r} & l_f \cdot c_{eq,f} - l_r \cdot c_{eq,r} \\ l_f \cdot c_{eq,f} - l_r \cdot c_{eq,r} & l_f^2 \cdot c_{eq,f} + l_r^2 \cdot c_{eq,r} \end{bmatrix} \tag{2-30}$$

Since both the measured system characteristics matrix from equation 2-20 and vehicle system characteristic matrix (equation 2-27) are in the same form, the unknown parameters (m_s, I_{yy}, l_f and l_r) can be estimated using a least squares algorithm.

In this chapter, Rozyn's algorithm has been explained. It estimates the inertial parameters by extracting the free-decay response from the acceleration data. The extracted free decay is used to build the state transition matrix. By solving the eigenvalue problem of this matrix and some filtering, the system characteristic matrix can be constructed. This matrix is then compared to a system characteristic matrix constructed using a simplified vehicle model to estimate the inertial parameters.

Chapter 3

Vehicle model

To obtain the acceleration data, which is used as input of the algorithm, a vehicle model is necessary. A half car model with four degrees of freedom is defined which is excited by a random road profile. Using a half car model means that the following inertial parameters can be estimated:

- Body mass
- Pitch moment of inertia
- Longitudinal position centre of gravity

It is also possible to use a more complex vehicle model, such as a 12 degree of freedom vehicle model which can estimate more parameters, such as the height of the centre of gravity. However, due to time constraints, a half car model is chosen. This model can still estimate important parameters and is able to show what happens when more realistic conditions are applied on Rozyn's algorithm.

The body of the vehicle is connected to the wheels with springs and dampers. The wheels are connected to the road with tyres, modelled as springs and dampers. A figure of the used half car model can be seen in Figure 3-1.

The following coordinates have been chosen as general coordinates:

- Vertical movement of the front wheels of the vehicle ($\mathrm{x}_{t,f}$)
- vertical movement of the rear wheels of the vehicle ($\mathrm{x}_{t,r}$)
- vertical movement of the body of the vehicle (x_s)
- pitch angle of the body of the vehicle (φ)

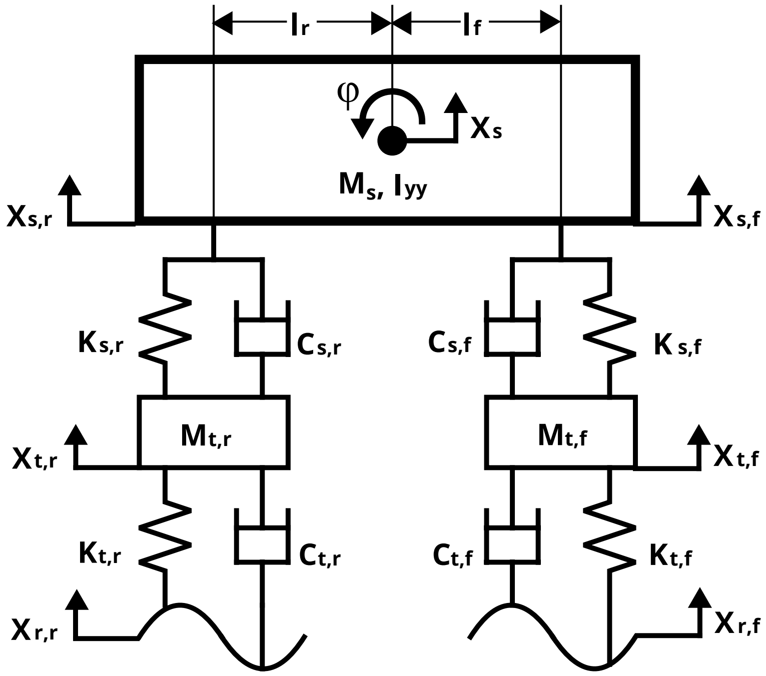

Figure 3-1: Half car model with as generalised coordinates $x_{t,f}$, $x_{t,r}$, x_s and φ.

The resulting acceleration of the vehicle body will be used to estimate the vehicle mass, pitch inertia and longitudinal position of the centre of gravity.

The generalized coordinates of this systems are:

$$q = \begin{bmatrix} x_{t,f} & x_{t,r} & x_s & \varphi \end{bmatrix}^T \tag{3-1}$$

And the state X is as follows:

$$X = \begin{bmatrix} q \\ \dot{q} \end{bmatrix} \tag{3-2}$$

The equations of motion of this system are given by the following equations:

$$\mathbf{M}\ddot{q} + \mathbf{C}\dot{q} + \mathbf{K}q = \mathbf{B}u \tag{3-3}$$

Where u is the input of the system: $u = \begin{bmatrix} x_{r,f} & x_{r,r} & \dot{x}_{r,f} & \dot{x}_{r,r} \end{bmatrix}^T$

The mass, damping and stiffness matrices M, C, K and B are as follows:

$$M = \begin{bmatrix} m_{t,f} & 0 & 0 & 0 \\ 0 & m_{t,r} & 0 & 0 \\ 0 & 0 & m_s & 0 \\ 0 & 0 & 0 & I_{yy} \end{bmatrix} \tag{3-4}$$

$$K = \begin{bmatrix} k_{t,f} + k_{s,f} & 0 & -k_{s,f} & -l_f \cdot k_{s,f} \\ 0 & k_{t,r} + k_{s,r} & -k_{s,r} & l_r \cdot k_{s,r} \\ -k_{s,f} & -k_{s,r} & k_{s,f} + k_{s,r} & l_f \cdot k_{s,f} - l_r \cdot k_{s,r} \\ -l_f \cdot k_{s,f} & l_r \cdot k_{s,r} & l_f \cdot k_{s,f} - l_r \cdot k_{s,r} & l_f^2 \cdot k_{s,f} + l_r^2 \cdot k_{s,r} \end{bmatrix} \tag{3-5}$$

$$C = \begin{bmatrix} c_{t,f} + c_{s,f} & 0 & -c_{s,f} & -l_f \cdot c_{s,f} \\ 0 & c_{t,r} + c_{s,r} & -c_{s,r} & l_f \cdot c_{s,r} \\ -c_{s,f} & -c_{s,r} & c_{s,f} + c_{s,r} & l_f \cdot c_{s,f} - l_r \cdot c_{s,r} \\ -l_f \cdot c_{s,f} & l_r \cdot c_{s,r} & l_f \cdot c_{s,f} - l_r \cdot c_{s,r} & l_f^2 \cdot c_{s,f} + l_r^2 \cdot c_{s,r} \end{bmatrix} \tag{3-6}$$

$$B = \begin{bmatrix} \frac{k_{t,f}}{m_{t,f}} & 0 & \frac{c_{t,f}}{m_{t,f}} & 0 \\ 0 & \frac{k_{t,r}}{m_{t,r}} & 0 & \frac{c_{t,r}}{m_{t,r}} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3-7}$$

Where $x_{s,f}$ and $x_{s,r}$ can be written as follows, according to the linearised equations:

$$x_{s,f} = x_s + l_f \cdot \varphi \tag{3-8}$$

$$x_{s,r} = x_s - l_f \cdot \varphi \tag{3-9}$$

In these equations, the subscripts s, t and r represents, respectively, the sprung mass (body), mass of the wheels (including tyres) and the road. The subscript f and r stands for the front and rear, respectively. The position of the centre of gravity with respect to, respectively the front and rear wheels are represented by l_f and l_r. For these equations, small angles for the pitch angles are assumed to linearise the equation. This can be done with no problem, since the pitch angle of the vehicle driving over a road is very small, as can be seen in Figure A-3.

The derivative of the state, $\dot{X}$, can then be calculated using the following equation:

$$\dot{X} = A_0 \cdot X \tag{3-10}$$

Where A_0 is:

$$A_0 = \begin{bmatrix} 0 & I \\ -M^{-1}K & -M^{-1}C \end{bmatrix} \tag{3-11}$$

This results in the accelerations and velocities of the generalised coordinates. The input of the estimation method are the accelerations of the front and the rear of the body, $\ddot{x}_{t,f}$ and $\ddot{x}_{t,r}$.

To check the implemented vehicle model, the motion of the body and the wheels of the vehicle have been plotted against the time for a free decay of the vehicle and using a random road profile. The pitch angle for the body has also been illustrated. The results show that the pitch angle of the body of the vehicle is very small, and the linearization in equation 3-8 and 3-9 can be used. The results of this can be seen in Appendix A-1.

In this chapter, the vehicle model, which is used to generate the acceleration data is explained. A half car vehicle model is chosen with 4 degrees of freedom, namely the vertical position of the front and rear wheels, the vertical position of the body and the pitch angle. Using a half car model means that the algorithm is able to estimate the following parameters: Mass, longitudinal position of centre of gravity.

Chapter 4

Problem Statement

The vehicle model, which is introduced in Chapter 3 needs to be excited by a road profile for the simulation of the acceleration data. Rozyn uses a road profile which is described in the ISO 8608 standard [16]. It is important to understand the road profile used by Rozyn to explain the shortcomings of using this road profile. The road profile used by Rozyn is explained in Section 4-1-1. The problem with this road profile description is that it can only be used for shorter road sections in the range of 100 meters [7]. Rozyn, however uses a measurement period of 1,000 seconds with a velocity of the vehicle of 100 km/h. This means that the ISO 8608 norm cannot be used to accurately describe the road profile over this road section. A more realistic road profile is described by Bjogsö and explained in Section 4-1-2. Also, the vehicle conditions used by Rozn, in terms of the measurement period and the velocity, are beneficial for the accurate estimation of the inertial parameters, as explained in Section 4-2.

4-1 Road Profile

4-1-1 ISO 8608

The road profile used by Rozyn is constructed using methods discussed in Cebon [17]. Cebon uses a method described by the ISO Standard: ISO 8608 [16]. This method reports the road profile as a Power Spectral Density (PSD). The PSD describes the power of the road elevation versus the wavenumber. The wavenumber is the spatial frequency of the wave, measured in cycles per unit distance. Using the PSD to describe the road profile, means that only two parameters are necessary to describe the road profile, the roughness of the road, $G_d(n_0)$, and the slope of the fitted PSD, which is the waviness of the road profile, w. In Figure 4-1 the fitted PSD is visible. The roughness $G_d(n_0)$ is displayed on the y-axis at the reference spatial frequency n_0. The exponent of the equation, w, is the slope of the fitted PSD.

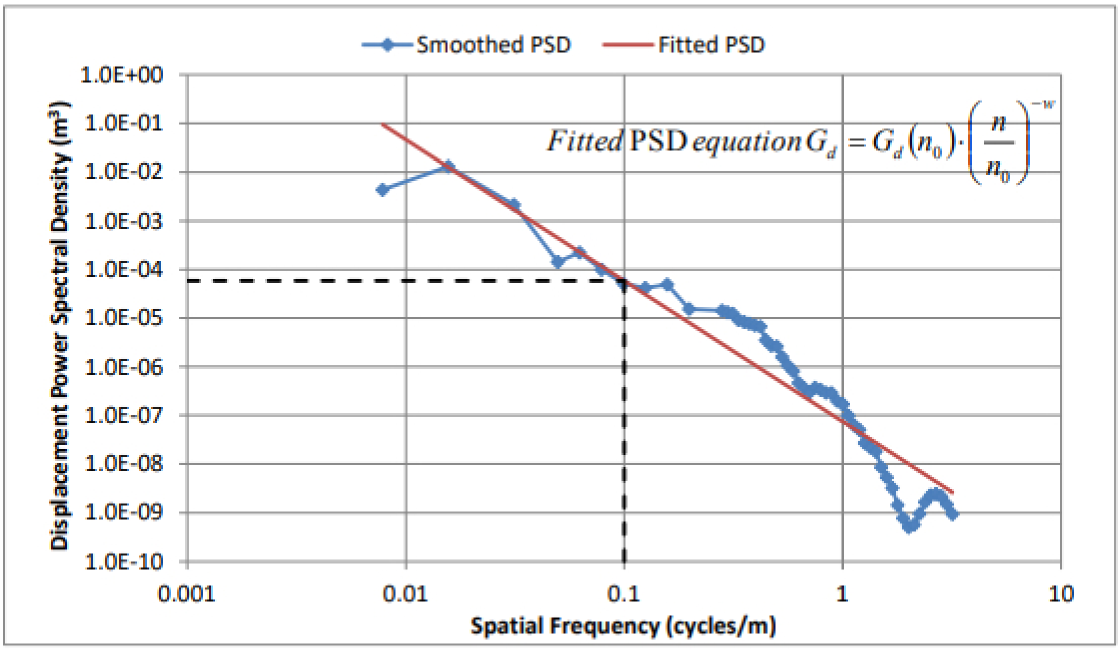

Figure 4-1: Smoothed and Fitted Power Spectral Density (PSD) according to ISO 8608. [18]

The equation for the fitted PSD is:

$$G_d(n) = G_d(n_0) \cdot \left(\frac{n}{n_0}\right)^{-w} \tag{4-1}$$

In this equation G_d represents the vertical displacements of the road profile as function of the frequency n, n_0 represents the reference spatial frequency [cycles/m], which is 0.1 for the ISO 8608 road description. The slope, w, is the waviness of the road.

The unevenness index, $G_d(n_0)$, is based on the quality of the road surface. The values can be seen in Table 4-1. In the ISO 8608 norm, eight classes are identified: from class A to class H. Class A represents the roads with a good quality and low roughness. Class H roads have a high degree of roughness. Rozyn does not include which road class is used for the road profile. In the simulation road class A has been used, since this represents an average quality asphalt road, according to Mucka et al [19]. The waviness is a measurement of the waves that the road profile follows. According to Mucka, the average waviness of a road profile is 2.

We can also describe the road profile as a function of distance. This is done by using a sinus approximation of the PSD. Agostinacchio [20] describes how a road profile can be generated, using the spectral density. According to Agostinacchio, the ISO 8608 norm provides that the roughness profile of the road surface can be defined as follows:

$$G_d(n) = \lim_{\Delta n \to 0} \frac{\psi_x^2}{\Delta n} \tag{4-2}$$

Where ψ_x^2 is the mean square value of the component of the signal for the spatial frequency n, within the frequency band Δn.

The mean square value can also be written as:

$$\psi_x^2 = \frac{A_i^2}{2} \tag{4-3}$$

Combining these two equations, the following results for the amplitude A_i

$$A_i = \sqrt{2\psi_x^2} = \sqrt{2\Delta n G_d(n_i)} \tag{4-4}$$

The profile of the road can be written as a simple harmonic function as follows:

$$h(x) = \sum_{i=0}^{N} A_i \cdot cos(2\pi \cdot n_i \cdot x_i + \varphi_i) \tag{4-5}$$

In this equation A_i represents the amplitude of the road vertical displacements, n_i is the spacial frequency in cycles/meter, x the abscissa variable from 0 to L (the length of the road). φ is the phase angle, with random angles, uniformly distributed between 0 and 2π.

Combining Equation 4-4 and 4-5 will result in the following equation, which describes the spot heights of the road surface:

$$h(x) = \sum_{i=0}^{N} \sqrt{2 \cdot \Delta n \cdot G_d(n_i)} \cdot cos(2\pi \cdot n_i \cdot x_i + \varphi_i) \tag{4-6}$$

The road profile can then be generated by combining Equation 4-1 and 4-6, which results in a final equation for the artificial road profile:

$$h(x) = \sum_{i=0}^{N} \sqrt{2 \cdot \Delta n \cdot G_d(n_0)} \cdot \left(\frac{n_0}{n}\right) \cdot cos(2\pi \cdot n_i \cdot x_i + \varphi_i) \tag{4-7}$$

Where $\Delta n = 1/L$ and $G_d(n_0)$ is a degree of roughness, as seen in Table 4-1.

Table 4-1: ISO 8608:2016 Road classification

Road class	$G_d(n_0) \cdot 10^{-6}$ m^3		
	Lower limit	Geometric mean	Upper limit
A	-	16	32
B	32	64	128
C	128	256	512

An example of a generated road profile can be seen in Figure 4-2.

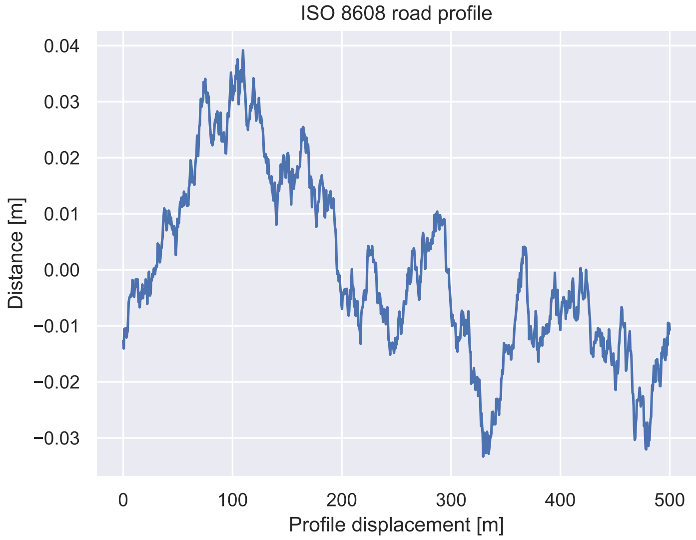

Figure 4-2: Road profile generated with the ISO 8608: 1995 description. The variability of the road is the same over the whole section.

The road surface used for the front and the rear wheels are the same, but only delayed for the rear wheels. The delay is based on the velocity of the vehicle and the wheelbase.

The problem with the ISO road profile description is that it is a stationary Gaussian process. When a system is excited by a stationary Gaussian process, the autocorrelation of the acceleration data will be equivalent to the free decay response [21]. However, stationary Gaussian processes cannot accurately describe the road profile of longer sections, since they contain sections with above average irregularity. Furthermore, the ISO 8608 description is the result of straight line fitting of the PSD [22]. This means that the PSD provided in the ISO 8608 description is not always a good representation of the PSD of the real road profiles. Therefore, more realistic road profile descriptions have been proposed, as explained in the following section.

4-1-2 Laplace

Although the ISO 8608 description of the road profile is widely used in literature ([19], [18], [23], etc.) it is only suitable for shorter sections of a few hundred meters [7]. Bogsjö et al [8] proposes another description for the road profile, which has a more realistic course, which is called the Laplace description. In this description, the road is divided into shorter sections with some variability between the different sections. The variance is continuous varied according to the Laplace distribution. This description gives a more accurate profile of the road for longer sections.

The Laplace description can be seen as a non-stationary Gaussian process with randomly varying variance of the irregularities. For the simulation purpose, one road profile of 3500 meter is compiled, with 35 sections of 100 meter. This is equal to driving 100 km/h for 120 seconds.

An example of a Laplace road profile, consisting of 5 sections of 100 meter, can be seen in Figure 4-3. The variance between the sections is clearly visible. The first section between 0 and 100 meters has a higher amplitude than the section between 100 and 200 meters. Also, if compared with the ISO 8608 road profile in 4-2 it can be seen that the ISO 8608 road profile is much more constant in amplitude over the distance.

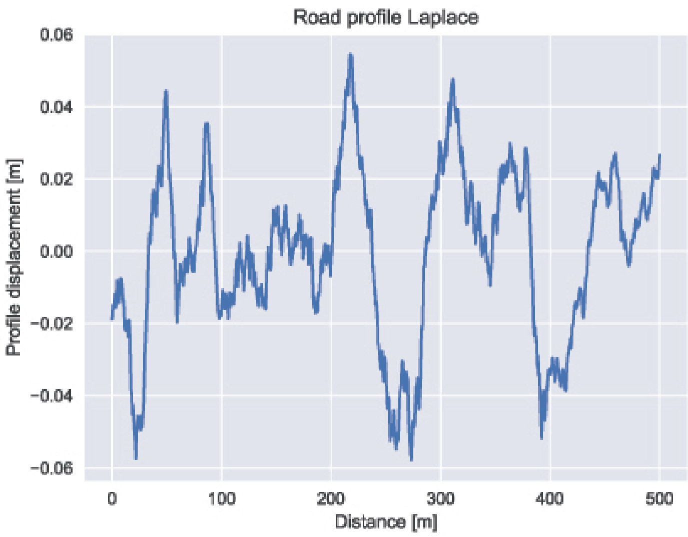

Figure 4-3: Road profile generated according to the Laplace description of Bogsjö. Visible is the different amplitudes between the different sections with a length of 100m each.

4-2 Vehicle Conditions

For the inertial parameter estimation, not only the road profile is an important factor on the results, but also the velocity. The wavelength of the oscillations of the road are constant, however the frequency changes depending on the velocity of the vehicle. Changing the velocity might influence the performance of the estimation of the inertial parameters. Rozyn only uses a velocity of 100 km/h. This is quite high, since it is often only reached after driving for some time. In this research, a range of velocities is selected to test the algorithm on its performance. The range selected is 30-100 km/h. 30 km/h is selected as lower end of the range, since that is a common velocity in residential areas. On the highway, 100 km/h is a normal velocity.

To identify the road influence on the oscillation of the body of the vehicle, a frequency analysis is performed. The road will have a constant wavelength, independent on the velocity of the vehicle. The propagation of the road however is dependent on the velocity. This means that the impact of the road onto the vehicle is dependent on the velocity of the vehicle.

The spatial frequency of the road in m^{-1} can be seen in Figure 4-4. The spatial frequency is the inverse of the wavelength of the road:

$$\zeta = \frac{1}{\lambda} \tag{4-8}$$

Where ζ is the spatial frequency in m^{-1} and λ the wavelength in m.

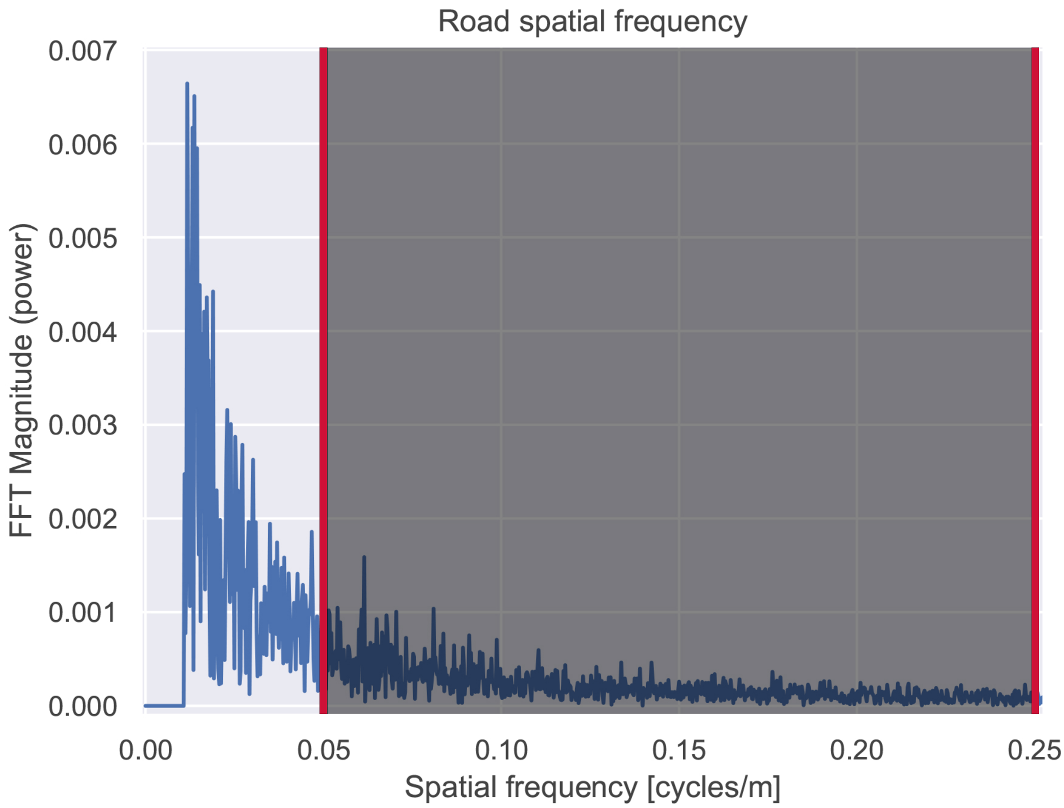

Figure 4-4: Fourier transform of the road. The area indicated is the oscillation range of the vehicle

The spatial frequency of the road will be constant, regardless of the velocity of the vehicle. The time frequency however is dependent on the velocity the vehicle is driving. This means that for different velocities of the vehicle, the road profile input will have a different frequency. If the frequency of the excitation of the road is the same as the natural frequency of the body of the vehicle it will add resonance to the body. If that is the case, it is more difficult to determine the modal parameters of the body of the vehicle. The relation between the frequency, the wavelength and the velocity is as follows:

$$\lambda = \frac{v}{f} \tag{4-9}$$

With the current test conditions, the body of the vehicle will have a natural frequency between 1.5 and 2 Hz. For a velocity of 30 km/h, this will result in a wavelength of the body between 4 and 6 meters. For a velocity of 100 km/h, the wavelengths will be between 14 and 20 meters. So, the bandwidth of the wavelengths of the oscillation of the vehicle will be between 4 and 20 meters, or 0.05 m^{-1} and 0.25 m^{-1} as spatial frequency. As can be seen in Figure 4-4, most of the frequencies lie in a lower range (< 0.05 m^{-1}). This means that most oscillations of the road profile will consist of longer waves and thus a lower spatial frequency and will not add any resonance with a high amplitude to the body of the vehicle.

Also the measurement time will influence the performance of the estimation. The longer the measurement time is, the more data points are available, with a more accurate estimation as result. However, in practice such long measurement periods will not be feasible. Furthermore, a constant velocity is desired during the measurement period. Changing the velocity will make the road profile less stationary, which will add extra errors in the estimation of the inertial parameters. Therefor, in this research a range of measurement periods is used, between 30 and 120 seconds. It is to be expected that a shorter measurement period results in a less accurate estimate of the inertial parameters, since less data points are available.

In this chapter, the shortcomings of Rozyn's algorithm has been explained. The road profile used by Rozyn is not a realistic description of a real life road profile, where there is much more variability over longer road sections which the ISO 8608 road profile fails to describe. Therefor, a more realistic road profile has been proposed, the Laplace description. Furthermore, the algorithm of Rozyn uses a measurement period of 1,000 seconds at a velocity of 100 km/h. During this period the velocity should remain constant for the algorithm to work optimal. This is not realistic, therfor, more realistic conditions are proposed, where the measurement time is varied between 10 and 120 seconds and the velocity between 30 and 100 km/h. The results of this can be seen in the next chapter, Chapter 5

Chapter 5

Results

In this chapter the results of the two different road profile, as introduced in Chapter 4-1 will be shown. But first, the algorithm needs to be validated for a correct implementation. To validate Rozyn's implemented algorithm, two tests have been conducted. In the first test, the vehicle is not excited by a road profile, instead the body of the vehicle has been given initial conditions to extract the free decay response of the vehicle. In the second test, the ISO 8608 road profile is used where the vehicle is driving 100 km/h and a measurement period of 100 seconds is used.

The following parameters have been used for the experiments:

Table 5-1: Vehicle parameters. The bold parameters indicate the parameters which are estimated using the algorithm.

Parameter	Value	Symbol	Unit
Tyre stiffness	$k_{t,f}$	200,000	N/m
Front suspension stiffness	$k_{s,f}$	45,000	N/m
Rear suspension stiffness	$k_{s,r}$	80,000	N/m
Tyre damping	$c_{t,f}$	50	N·s/m
Front suspension damping	$c_{s,f}$	2,800	N·s/m
Rear suspension damping	$c_{s,r}$	3,500	N·s/m
Front unsprung wheel mass	$m_{t,f}$	40	kg
Rear unsprung wheel mass	$m_{t,r}$	65	kg
Wheelbase	WB	2.8	m
Body mass	$\mathbf{m_s}$	**1,000**	**kg**
Body pitch moment of inertia	$\mathbf{I_{yy}}$	**2,000**	**kg·m^2**
Centre of gravity position (measured from front axle)	$\mathbf{l_f}$	**1.2**	**m**

5-1 Free Decay Response

The input of the algorithm is a free decay response. When the vehicle model is excited by the road, the free decay response can be extracted using an autocorrelation function. To validate the algorithm, a free decay response have been generated by setting the road profile to zero and giving the vehicle an initial condition. The resulting free decay response of the front and rear of the vehicle can be seen in, respectively, Figure 5-1 and 5-2. Since the algorithm uses the free decay response of the vehicle as input, this should result in a perfect estimation of the inertial parameters.

Figure 5-1: Free decay response of the front of the vehicle.

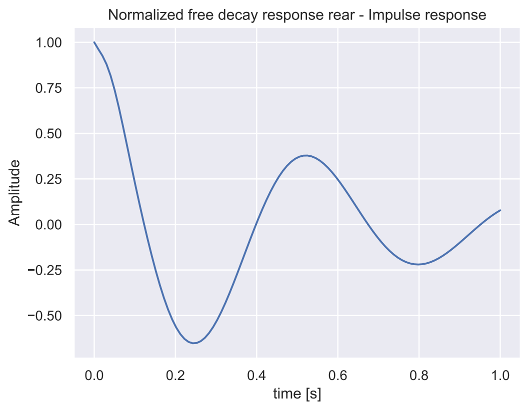

Figure 5-2: Free decay response of the rear of the vehicle.

The resulting estimates of the algorithm, with the free decay as input is as follows:

$$\text{Mass estimate} = 1{,}002 \text{ kg}$$
$$\text{Inertia estimate} = 2{,}009 \text{ kg}{\cdot}\text{m}^2$$
$$\text{Centre of gravity estimate} = 1.204 \text{ m}$$

As can be seen, the algorithm does indeed estimate the inertial parameters with a very high accuracy. These results can be used as baseline performance of the algorithm, and be compared with the results of the following experiments.

5-2 ISO 8608 Road Profile

In the second experiment, the algorithm is tested using the same parameters as used in Rozyn et al. This means that the vehicle is excited by the ISO 8608 road profile, while driving at a constant velocity of 100 km/h. The measurement period is, due to computational power, reduced to 100 seconds. However, as can be seen in the results, this does not contribute to large errors in the estimation of the parameters. The simulation creates 100 pseudo measurements, using 100 different random profiles, resulting in 100 different accelerations of the vehicle body. Each time, the autocorrelation function is used to extract the free decay response of the vehicle. The end result are 100 estimates for the inertial parameters. The average of these estimates is:

$$\begin{aligned}\text{Mass estimate} &: 984.7 \text{ kg}\\ \text{Inertia estimate} &: 2{,}038 \text{ kg}\cdot\text{m}^2\\ \text{Centre of gravity estimate} &: 1.201 \text{ m}\end{aligned}$$

The following figures have been generated for this experiment:

- Autocorrelation of the front acceleration, see Figure 5-3
- Autocorrelation of the rear acceleration, see Figure 5-4
- Comparison between autocorrelation and free decay for the front of the vehicle, see Figure 5-5
- Comparison between autocorrelation and free decay for the rear of the vehicle, see Figure 5-6
- Histogram for the mass estimates, see Figure 5-7
- Histogram for the inertia estimates, see Figure 5-8
- Histogram for the centre of gravity estimates, see Figure 5-9
- Boxplot of the errors of the estimates, see Figure 5-10

In Figure 5-3 and 5-4 the extracted free decay response of the vehicle body can be seen for the 100 different pseudo measurements. As can be seen, the difference between the different extracted free decay responses are very small. This indicates that the algorithm should be able to estimate the inertial parameters with a small deviation.

The extracted free decay response is also compared with the free decay response from Figure 5-1 and 5-2 and can be seen in Figure 5-5 and 5-6. As can be seen, there is a difference in amplitude between the real free decay response and the extracted free decay response. More important for the estimation of the inertial parameters is the period of the free decay, which is the same for the free decay and the autocorrelation function.

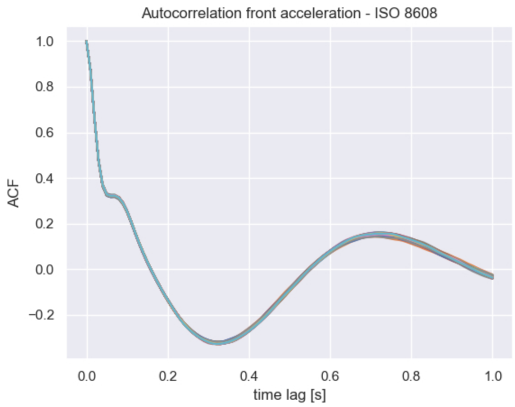

Figure 5-3: Autocorrelation of the front acceleration for 100 measurements using the ISO 8608 road profile. There is very little difference between the 100 different extracted free decay responses.

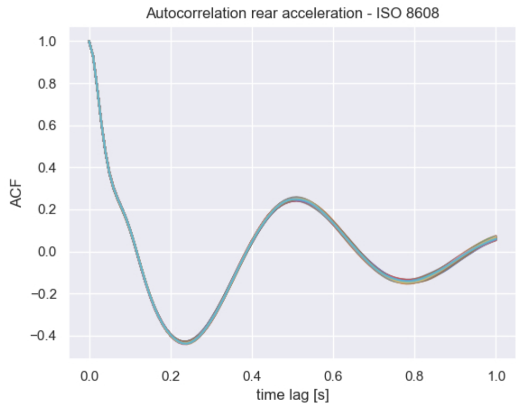

Figure 5-4: Autocorrelation of the rear acceleration for 100 measurements using the ISO 8608 road profile. There is very little difference between the 100 different free decay responses.

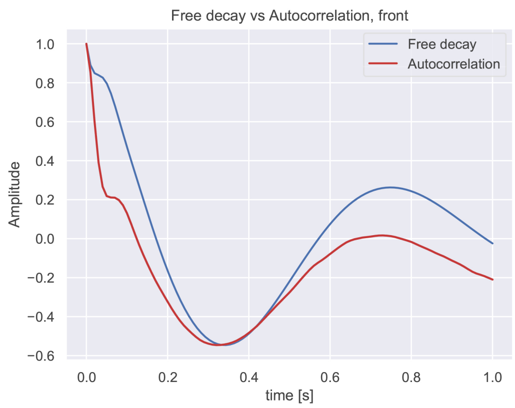

Figure 5-5: Free decay (blue) response compared to the extracted free decay response using the autocorrelation function (red) for the front of the vehicle. There is a difference in the amplitude, however the period is the same.

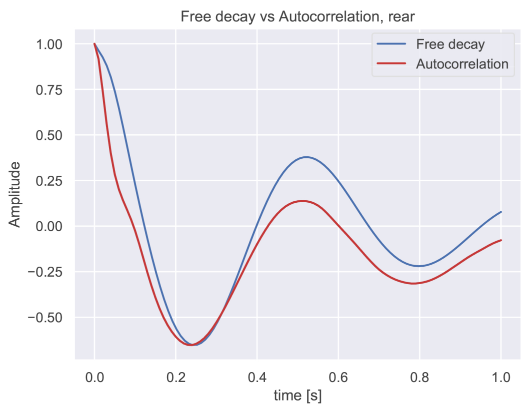

Figure 5-6: Free decay (blue) response compared to the extracted free decay response using the autocorrelation function (red) for the front of the vehicle. There is a difference in the amplitude, however the period is the same.

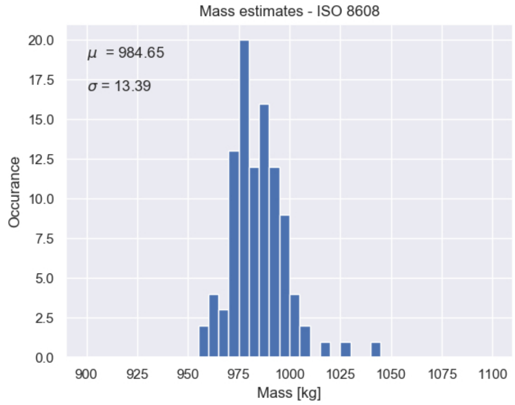

Figure 5-7: Histogram of the mass estimates with a good estimate of the mass parameter and a small deviation between the different estimates for the ISO 8608 road profile.

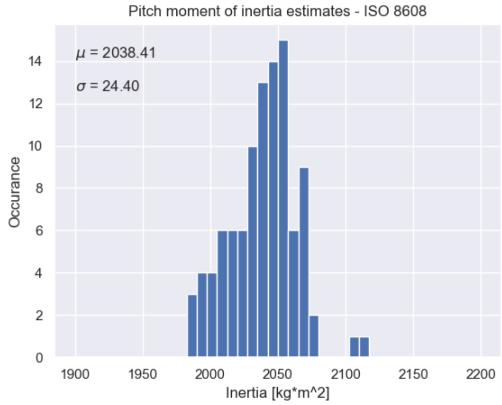

Figure 5-8: Histogram inertia estimates with a good estimate of the inertia parameter and a small deviation between the different estimates for the ISO 8608 road profile.

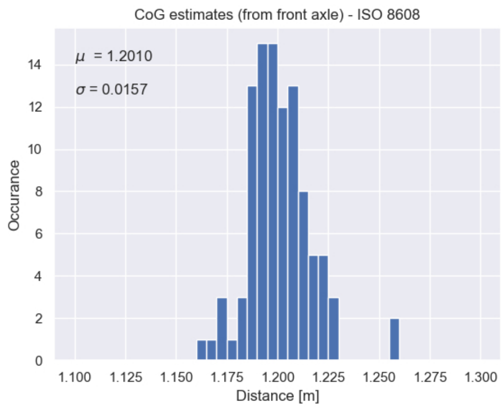

Figure 5-9: Histogram centre of gravity estimates with a very good estimate and a small deviation between the different estimates of the position of the centre of gravity for the ISO 8608 road profile.

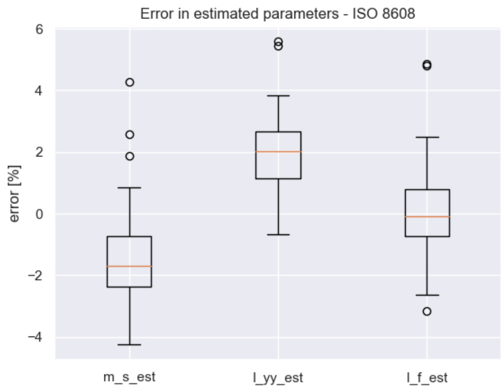

Figure 5-10: Boxplot of the errors of the estimates with very low deviation of the estimates of the inertial parameters for the ISO 8608 road profile.

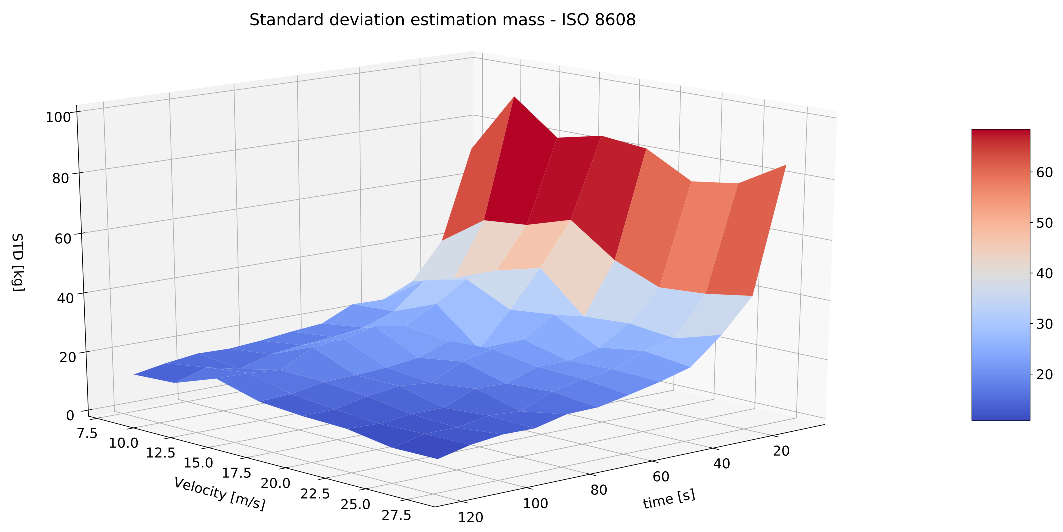

Figure 5-11: Standard deviation of the estimates of the vehicles mass for the ISO road profile. The standard deviation remains constant over varying velocity, where as it increases with decreasing measurement time.

The histogram in Figure 5-7 - 5-9 shows the estimates of the parameters each measurement. The results show that the inertial parameters can be estimated with a good average of the different measurements. Furthermore, the deviation between the different measurements is small, as can also be seen in the boxplot in Figure 5-10. All the estimates lie within a 6% error margin, the standard deviation σ between the different estimates is very small. This means that all the estimates are close to the average of the estimated inertial parameters. The standard deviation can also be used as performance indicator of the algorithm, since a small standard deviation indicates that less pseudo measurements are needed to tell something about the real inertial parameters. The results from the second experiment, with the ISO 8608 description as road profile are comparable to Rozyn's results.

The experiments have been repeated for different velocities, between 30 and 100 km/h and measurement periods: between 10 and 120 seconds. Each time, the inertial parameters have been estimated using 100 different random road profiles. The standard deviation of the different estimates for each condition can be seen in Figure 5-11 - 5-13. It is clearly visible that the standard deviation of the estimates remains constant over a varying velocity. The standard deviation however increases for decreasing measurement periods.

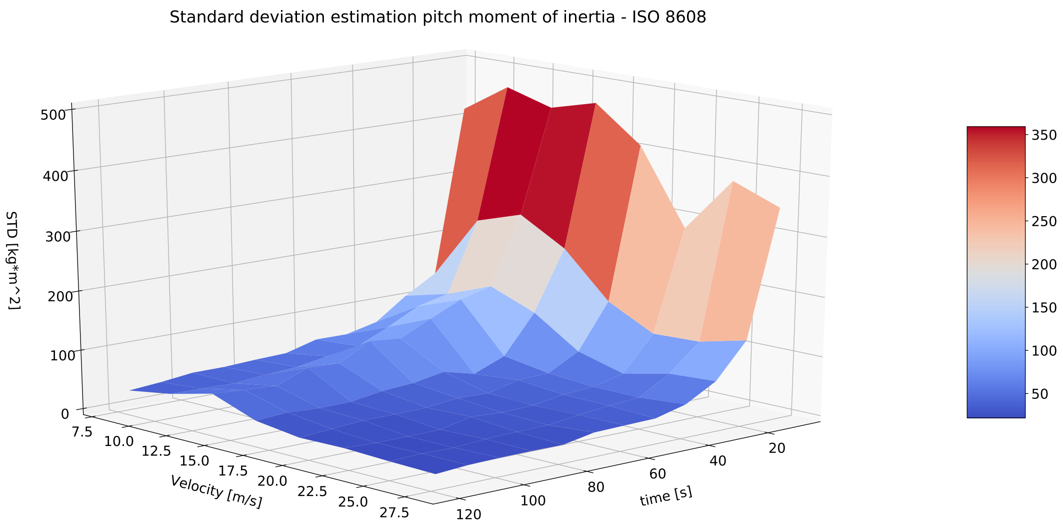

Figure 5-12: Standard deviation of the estimates of the vehicles pitch moment of inertia for the ISO road profile. The standard deviation remains constant over varying velocity, where as it increases with decreasing measurement time.

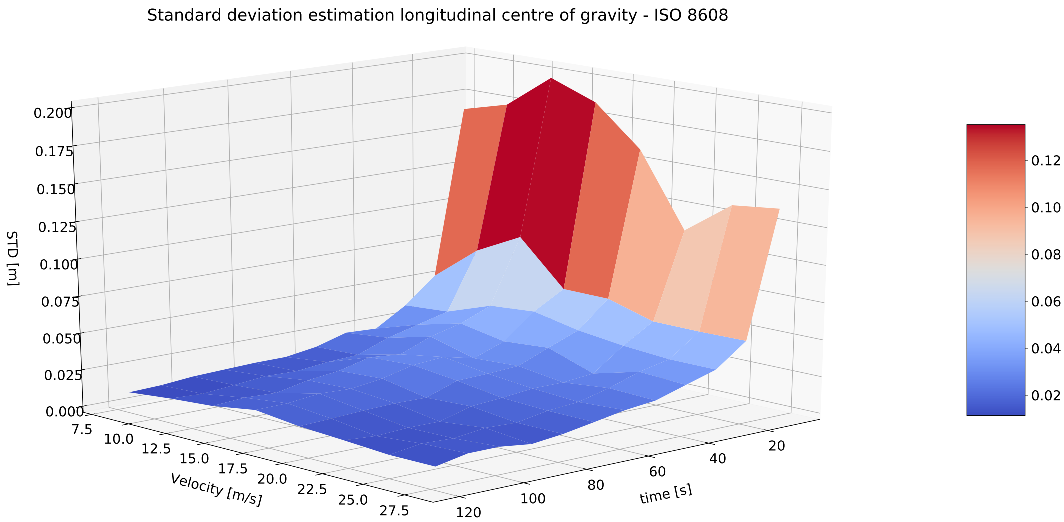

Figure 5-13: Standard deviation of the estimates of the vehicles longitudinal position of centre of gravity for the ISO road profile. The standard deviation shows a slight dependency over the velocity, where as it increases with decreasing measurement time.

5-3 Laplace Road Profile

In the following experiment we use the algorithm to estimate the parameters of the vehicle, while the vehicle is excited by the Laplace description of the road, as proposed by Bogsjö. This is done with the same parameters, thus a velocity of 100 km/h and a measurement time of 100 seconds. The estimation of the inertial parameters is as follows:

$$\text{Mass estimate} = 950 \text{ kg}$$
$$\text{Inertia estimate} = 2{,}088 \text{ kg}\cdot\text{m}^2$$
$$\text{Centre of gravity estimate} = 1.199 \text{ m}$$

The results of the first experiment can be seen in Figures 5-14 - 5-19. The average of the estimates are still fairly accurate (+- 5%), however the standard deviation of the measurements are a factor 5 higher, compared to the ISO 8608 road. This means that more measurements are needed for an accurate estimate of the inertial parameters.

These experiments have been repeated for measurement periods between 10 and 120 seconds and velocities between 30 and 100 km/h. The standard deviation of these estimation of the inertial parameters have been plotted against the velocity and measurement period and can be seen in Figure 5-20 - 5-22.

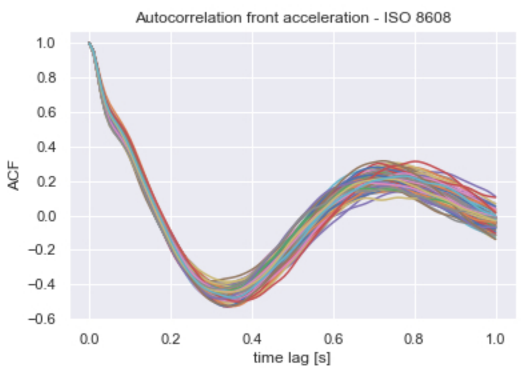

Figure 5-14: Autocorrelation front acceleration of the vehicle for 100 estimates, using the Laplace road profile. The free decay response shows more deviation in the extracted free decay response between the different measurements in comparison to the ISO 8608 road profile in Figure 5-3.

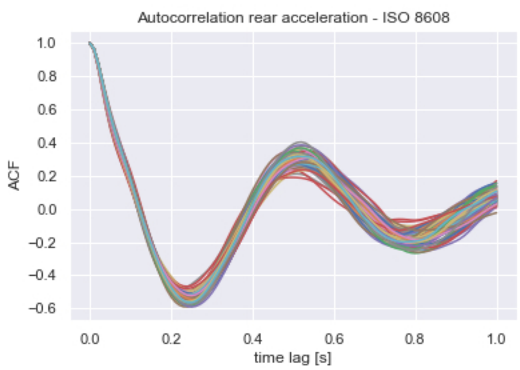

Figure 5-15: Autocorrelation rear acceleration of the vehicle for 100 estimates, using the Laplace road profile. The free decay response shows more deviation in the extracted free decay response between the different measurements in comparison to the ISO 8608 road profile in Figure 5-4.

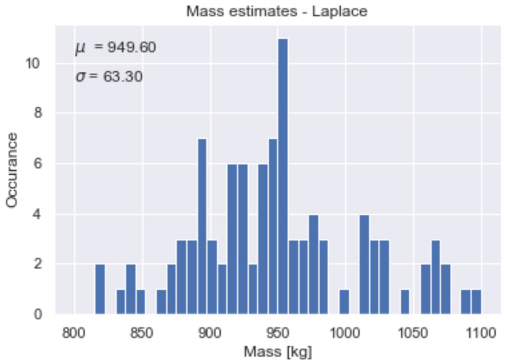

Figure 5-16: Histogram mass estimates for the Laplace road profile with higher deviation between the different estimates compared to the ISO 8608 road profile.

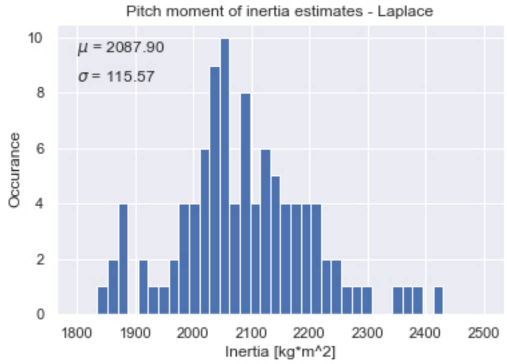

Figure 5-17: Histogram inertia estimates for the Laplace road profile with higher deviation between the different estimates compared to the ISO 8608 road profile.

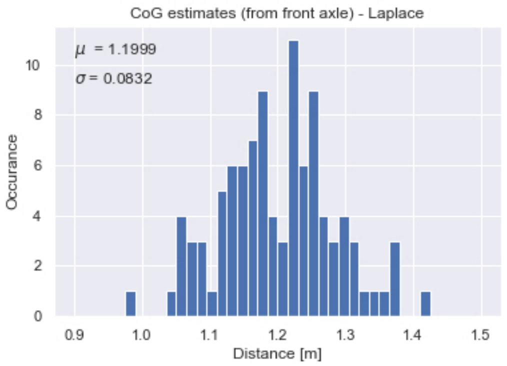

Figure 5-18: Histogram centre of gravity estimates for the Laplace road profile with higher deviation between the different estimates compared to the ISO 8608 road profile.

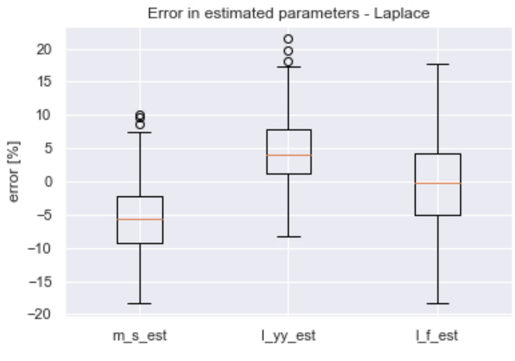

Figure 5-19: Boxplot of the errors of the estimates for the Laplace road profile with higher deviation between the different estimates compared to the ISO 8608 road profile.

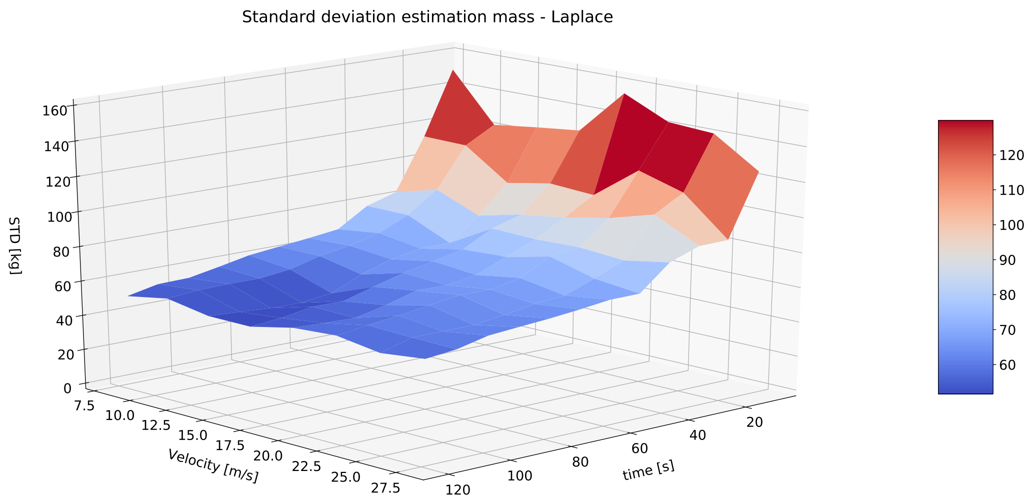

Figure 5-20: Standard deviation of the body mass estimates for the Laplace road profile. The standard deviation is lower for shorter measurement periods and does not change much for different velocities. Clearly visible is the higher deviation between the different estimates in comparison to the ISO 8608 road profile in Figure 5-11.

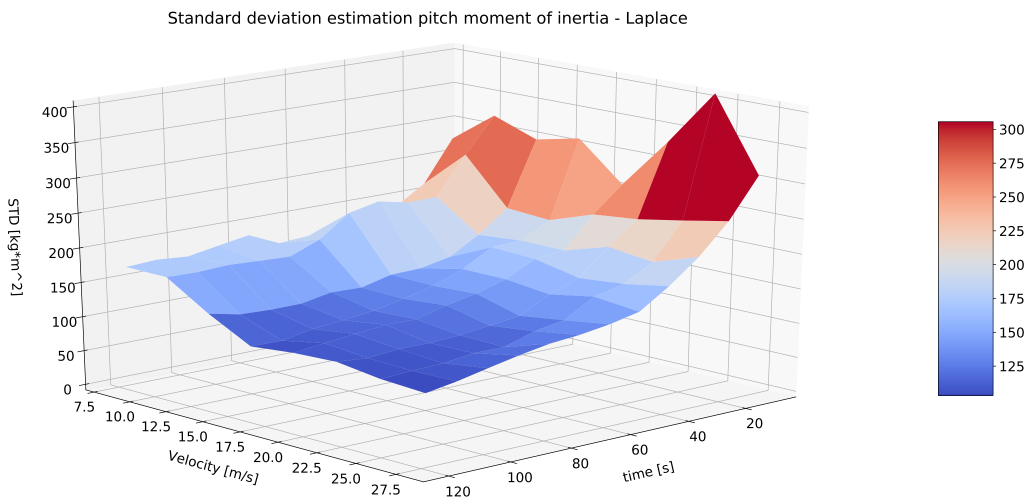

Figure 5-21: Standard deviation of the pitch moment of inertia estimates for the Laplace road profile. The standard deviation is lower for shorter measurement periods and shows a slight dependency on the velocity, where higher velocities results in lower deviations between the different estimates. Clearly visible is the higher deviation between the different estimates in comparison to the ISO 8608 road profile in Figure 5-12.

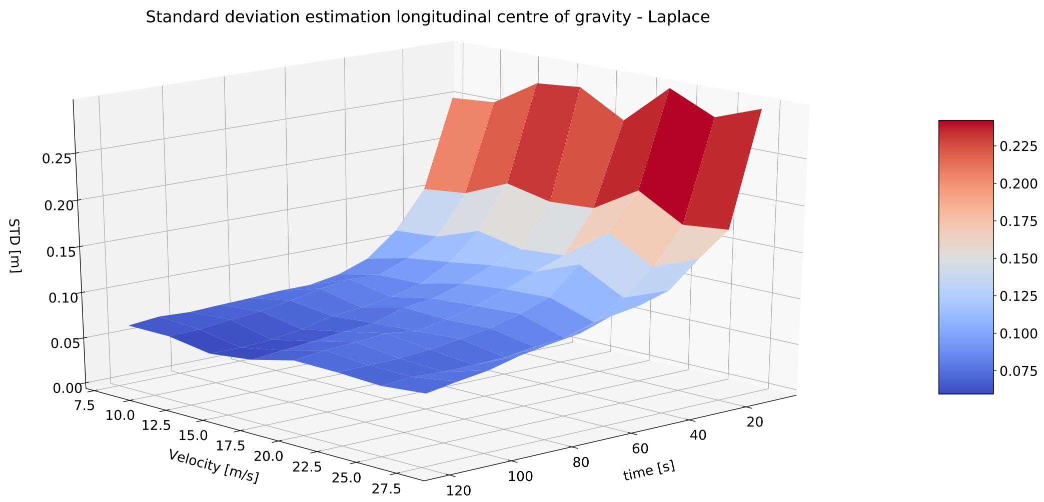

Figure 5-22: Standard deviation of the longitudinal position centre of gravity estimates for the Laplace road profile. The standard deviation is lower for shorter measurement periods. Clearly visible is the higher deviation between the different estimates in comparison to the ISO 8608 road profile in Figure 5-13.

It is clearly visible that the standard deviation of the estimates remains constant over a varying velocity. The standard deviation however increases for decreasing measurement periods.

Chapter 6

Conclusion & Discussion

6-1 Contributions

In this thesis, Rozyn's algorithm has been explained and implemented in Python. The input of the algorithm are vertical accelerations of the body of a vehicle. For this, a vehicle model has been introduced and implemented in Python. The vehicle model has been given an initial condition which results in a free decay of the vehicle. This is used to validate the implemented algorithm. Also, the road description, which is used by Rozyn, as described in the ISO 8608 standard, is implemented and used under the same conditions as Rozyn to verify that the algorithm has been implemented correctly.

The experiments have been repeated under different conditions. The velocity is varied between 30 and 100 km/h and the measurement period between 10 and 120 seconds. For each condition, 100 random road profiles have been used to generate 100 accelerations. From these accelerations, 100 different estimates of the inertial parameters have been made. The standard deviation between these estimates tells something about the accuracy of the measurement. A surface graph has been made where the standard deviation has been plotted against the velocity and measurement period. This is done for both the ISO 8608 and the Laplace road profile. From the results can be concluded that Rozyn uses a unrealistic framework for his algorithm, with a road profile which is not realistic and unrealistic vehicle conditions, such as a high velocity and a long measurement period.

6-2 Conclusion & Recommendation

In this thesis, a method for online inertial parameter estimation has been implemented and tested on multiple conditions. The algorithm is proposed by Rozyn et al and is able to estimate the mass of the body of the vehicle, the position of the centre of gravity and the inertial moments with high accuracy, according to Rozyn. The input of the algorithm is a free decay response which is extracted from vertical acceleration data using an autocorrelation function. This acceleration data is generated by exciting a simulated vehicle model with a road profile where the accelerometers are placed on the corners of the vehicle.

The road profile used by Rozyn is described in the ISO 8608 standard. It is a random road profile, constructed as a stationary and homogeneous Gaussian process. In real life, however, a road profile is never stationary and homogeneous but changes over time and distance. This is why the ISO 8608 standard is not suitable for generating longer road profiles, according to Johannesson. A more realistic road profile is proposed by Bjogsö. He calls it the Laplace description, where the road profile is constructed using smaller road sections with a random variability, according to the Laplace distribution, between those sections.

In this thesis, the algorithm for inertial parameter estimation as proposed by Rozyn is implemented in Python. Since the input of the algorithm is a free decay response, the implemented algorithm is first validated using a free decay response of the vehicle. This is done by setting the road profile to zero and giving the vehicle an initial condition. This results in a free decay response, where the movement of the vehicle decays to zero. The results show that the algorithm estimates the inertial parameters with a very high accuracy, within a 0.5 % error margin.

The method has also been validated by estimation of the inertial parameters, under the same conditions that Rozyn uses, with a random road profile, as defined in the ISO 8608 standard. Using this random road profile, the algorithm is able to estimate the parameters with high accuracy and small deviation between the estimated parameters, where all the measurements fall into a 6% error range. The average of the estimated inertial parameters has a deviation of 2 % compared to the real inertial parameters.

The problem with Rozyn's algorithm is that he uses an unrealistic framework with a high velocity and measurement period of 1000 seconds at a constant velocity of 100 km/h. This is why, in this thesis, a range of velocities, between 30 and 100 km/h and measurement periods, between 10 and 120 seconds, have been proposed. The algorithm has been used to estimate the inertial parameters, using the ISO 8608 road profile. Here, 100 pseudo measurements for each velocity and time parameter on 100 random road profiles are used. This results in 100 estimates of the inertial parameters. These results are used to calculate both the average of the estimates and the standard deviation of the estimates. The standard deviation indicates the reliability of the algorithm. A low standard deviation means that less estimates are needed for insight in the inertial parameters. The standard deviation is plotted against the measurement period and the velocity in a 3D surface plot. The results show that the standard deviation is independent on the velocity but increases with shorter measurement periods. The standard deviation between the different estimates is small for measurement periods longer than 60 seconds.

The algorithm has also been tested on the Laplace road description, proposed by Bjogsö, with velocities between 30 and 100 km/h and measurement periods between 10 and 120 seconds. Results show that using the Laplace description for the road profile results in a much larger standard deviation of the estimated inertial parameters in comparison to Rozyn's algorithm, under the same conditions of the measurement time and velocity. Rozyn uses the ISO 8608 description for the road profile which is constructed according to a stationary Gaussian process. This means that the free decay response of the vehicle can be extracted more accurately by the autocorrelation function. The Laplace description is constructed using a Laplace description, which can be seen as a non-stationary Gaussian process with a random variability between different sections of the road profile. This means that the autocorrelation function cannot extract the free decay response accurately enough. This can be seen in the results, where for the same conditions for the velocity and measurement period, the standard deviation of the estimation increases by a factor 5. Reducing the measurement period from 120 seconds will increase this standard deviation of the estimated inertial parameters even more.

The results of other experiments, where the measurement time and velocity is varied, shows that for the estimates of the inertial parameters the velocity has no or little influence on the standard deviation of the estimates. The standard deviation of the estimates does increase with decreasing measurement periods.

Recommended is to estimate the parameters using a measurement period longer than 60 seconds. The deviation in the estimated parameters increase heavily for measurement periods below 60 seconds.

The algorithm can still be used on non-stationary road profiles. However, more and longer measurements are needed for the algorithm to return with an accurate estimation of the inertial parameters. Even then, some errors in the estimated parameters in the order of 10% are present.

Appendix A

Appendix

A-1 Vehicle Model

In Figure A-1 and A-2 the response of the front and rear, respectively, can be seen after the body (green line) and the wheels (blue line) of the vehicle have been given an initial condition. The road surface (red line) is zero. It clearly shows the free decay of the vehicle until steady state. The vehicle behaves as expected.

In Figure A-3 the pitch angle during the free decay can be seen. It shows that the pitch angle is indeed very small. This is also the case if the vehicle is excited by a road profile. This indicates that the linearized equations in 3-8 and 3-9 can indeed be used.

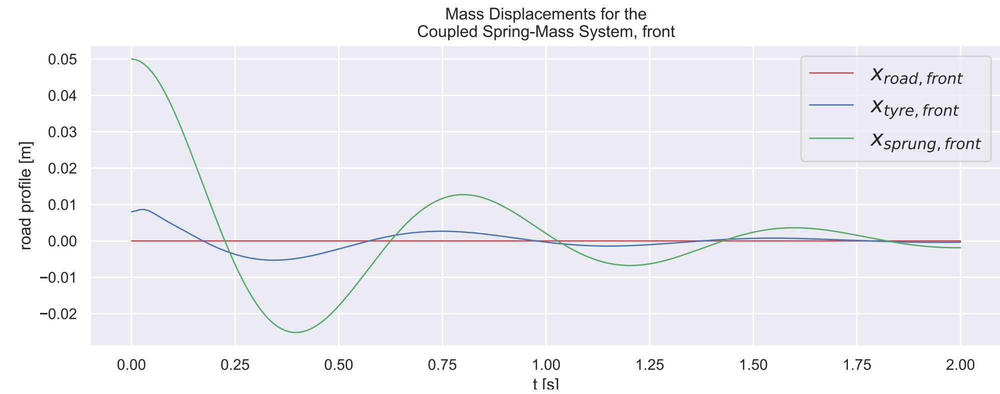

Figure A-1: Front vehicle response upon free decay

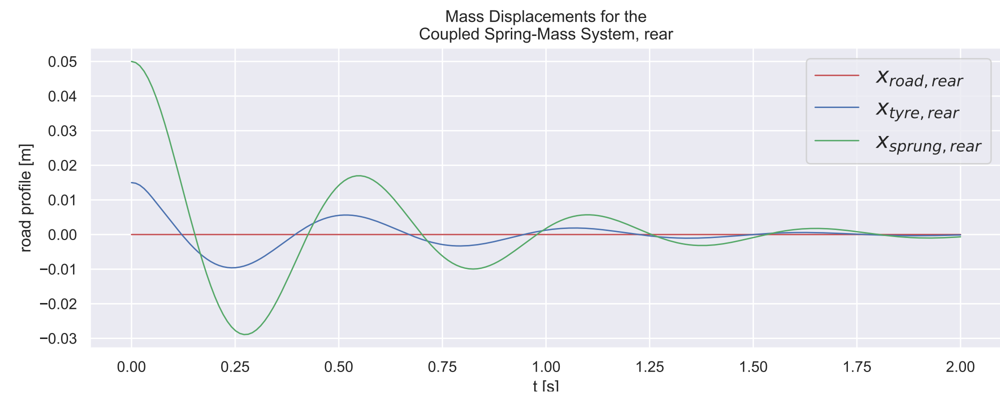

Figure A-2: Rear vehicle response upon free decay

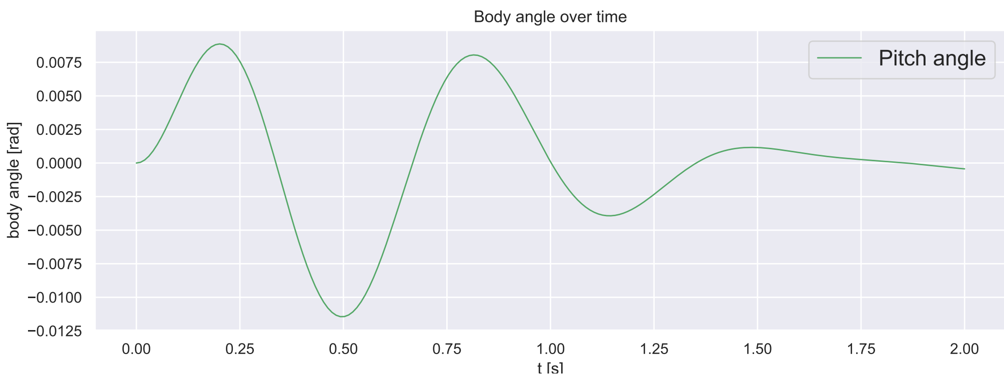

Figure A-3: Pitch angle during free decay

Figure A-4: Front vehicle response ISO 8608 road profile

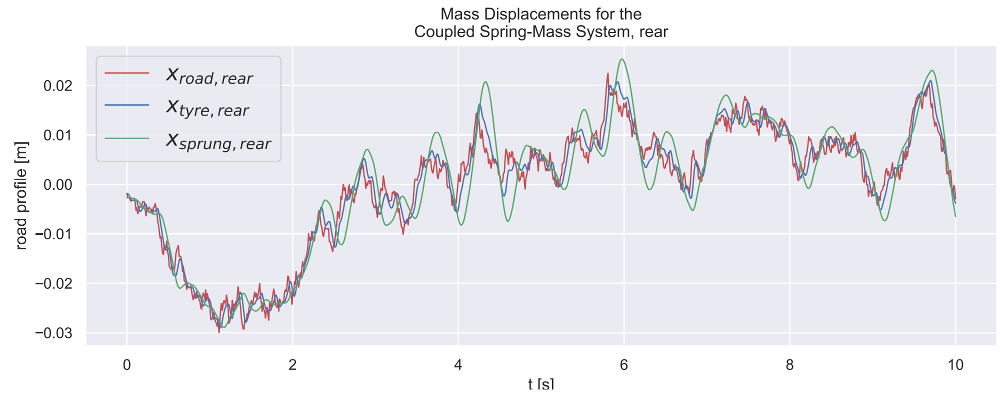

Figure A-5: Rear vehicle response ISO 8608 road profile

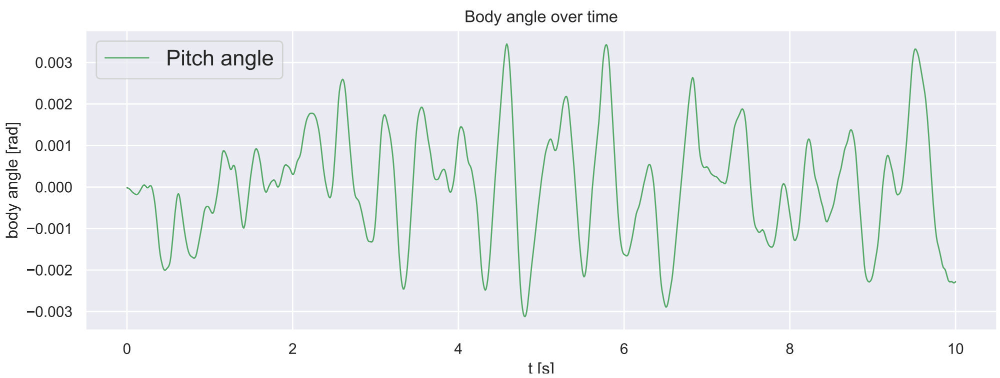

Figure A-6: Pitch angle ISO 8608 road profile

A-2 Programming Code

A-2-1 Main File

```
"""
@author: JoÃńl Dijkhuizen
"""
import matplotlib.pyplot as plt
from matplotlib.font_manager import FontProperties
import numpy as np
from scipy.integrate import odeint
from EOM1 import parameters, road, solution, dstate, state_transition,
    estimation, parameter
from scipy.optimize import fsolve
import time
from statsmodels.tsa.stattools import acf, ccf
import seaborn as sns
from scipy.signal import resample
from scipy.fftpack import fft

sns.set()
plt.close("all")
start = time.time()

## Import paramters
[m, k, c, I, l_s, l, var] = parameters()
m_t_f, m_t_r, m_s = m # Masses
k_t_f, k_t_r, k_s_f, k_s_r = k # Stiffnesses
c_t_f, c_t_r, c_s_f, c_s_r = c # Damping coefficients
I_yy = I # Inertia moment
l_f, l_r = l # Position CoG with respect to the unsprung masses
l_s_f, l_s_r = l_s # Length of the springs
vel, stoptime, L, T_s, N, WB, road_profile = var
p = [m_t_f, m_t_r, m_s, k_t_f, k_t_r, k_s_f, k_s_r, c_t_f, c_t_r, c_s_f,
    c_s_r, I_yy, l_f, l_r] # Pack up the paramters

## Time vector
t = np.linspace(0, stoptime, N)

mp = 1 # Number of pseudo measurments

m_s_est = np.zeros(mp)
I_yy_est = np.zeros(mp)
l_f_est = np.zeros(mp)

ddx_s_f = np.zeros((101, mp))
ddx_s_r = np.zeros((101, mp))

for z in range(0, mp):
    print("z = ", z+1, "/", mp)
    ## Import road profile
    x_r = [x_r_f, dx_r_f, x_r_r, dx_r_r, L] = road(t, var, z)
```

```
    ## Initial conditions
    if road_profile in range(1,2): # Random road
        x_t_f0 = x_r_f[0]
        dx_t_f0 = dx_r_f[0]
        x_t_r0 = x_r_r[0]
        dx_t_r0 = dx_r_f[0]
        x_s0 = (x_r_f[0] + x_r_r[0])/2
        dx_s0 = (dx_r_f[0] + dx_r_r[0])/2
        phi0 = (x_r_f[0] + x_r_r[0])/WB
        dphi0 = (dx_r_f[0] + dx_r_r[0])/WB

    if road_profile == 0: # Free decay
        x_t_f0 = -0.015
        dx_t_f0 = 0
        x_t_r0 = -0.015
        dx_t_r0 = 0
        x_s0 = -0.03
        dx_s0 = 0
        phi0 = 0.00
        dphi0 = 0

    state = np.zeros((N, 8)) # define size of the state
    state_0 = np.array([x_t_f0, x_t_r0, x_s0, phi0, dx_t_f0, dx_t_r0,
        dx_s0, dphi0]) # initial condition state
    state[0, :] = state_0 # initial condition state

    ## Integration
    for i in range(0, N - 1):
        tt = [0, T_s] #integrate over 1 timestep
        x_r_f_ = x_r_f[i] # corresponding road profile
        dx_r_f_ = dx_r_f[i]
        x_r_r_ = x_r_r[i]
        dx_r_r_ = dx_r_r[i]
        x_ra = [x_r_f_, dx_r_f_, x_r_r_, dx_r_r_, L]
        state_temp = odeint(solution, state[i, :], tt, args=(p, x_ra)) #
            Integrate state dot using solution function, state0 as initial
             values and paramters p
        state[i+1, :] = state_temp[1, :] # new state

    state = np.transpose(state)

    dstate_temp = dstate(state, p, t, x_r)
    ddx_s = np.ravel(dstate_temp[6, :])
    ddphi = np.ravel(dstate_temp[7, :])

    T_s_sample = 0.01 #Sampling frequency 100 Hz
    N_sample = round(stoptime/T_s_sample)

    ddx_s_f__ = ddx_s + l_f*ddphi
    ddx_s_r__ = ddx_s - l_r*ddphi

    if road_profile == 2:
```

```
            ddx_s_f_ = resample(ddx_s_f__, N_sample) # Sample accelerations on sampling frequency
            ddx_s_r_ = resample(ddx_s_r__, N_sample)
        else:
            ddx_s_f_ = ddx_s_f__
            ddx_s_r_ = ddx_s_r__

        if road_profile in range(1,2): # take autocorrelation of signal for random road input
            lag = round(1/T_s_sample)
            ddx_s_f[:,z] = acf(ddx_s_f_, nlags = lag)
            ddx_s_r[:,z] = acf(ddx_s_r_, nlags = lag)
            ACC_t = np.linspace(0, lag*T_s_sample, lag + 1)
            ddx = [ddx_s_f[:,z], ddx_s_r[:,z]]

            plt.figure(1)
            plt.plot(ACC_t, ddx_s_f[:,z])
            plt.xlabel('time lag [s]')
            plt.ylabel('ACF')
            plt.title('Autocorrelation front acceleration - ISO 8608')
            plt.savefig('Autocor_front')

            plt.figure(2)
            plt.plot(ACC_t, ddx_s_r[:,z])
            plt.xlabel('time lag [s]')
            plt.ylabel('ACF')
            plt.title('Autocorrelation rear acceleration - ISO 8608')
            plt.savefig('Autocor_rear')

        if road_profile == 0:  # Free decay input
            ddx_s_f = ddx_s_f__
            ddx_s_r = ddx_s_r__

            ddx_s_f = ddx_s_f__/np.max(ddx_s_f__)
            ddx_s_r = ddx_s_r__/np.max(ddx_s_r__)
            ddx = [ddx_s_f, ddx_s_r]

            plt.figure(1)
            plt.plot(t, ddx_s_f)
            plt.xlabel('time [s]')
            plt.ylabel('Amplitude')
            plt.title('Normalized free decay response front - Impulse response')
            plt.savefig('Acceleration free decay front')

            plt.figure(2)
            plt.plot(t, ddx_s_r)
            plt.xlabel('time [s]')
            plt.ylabel('Amplitude')
            plt.title('Normalized free decay response rear - Impulse response')
            plt.savefig('Acceleration free decay rear')

```

```
    ## State transition
    A_meas, omega_nS_hz, omega, index, omega_nS_hz_noise = state_transition(ddx, N, T_s_sample, t)

    if omega != [0,0]: # Estimate parameters if identification is succesfull
        A_est, cost, k_eq_f, k_eq_r = estimation(A_meas, k, c, l, omega)
        x0 = [800, 1750, 1] #initial estimate mass, inertia, l_f
        par_est = fsolve(parameter, x0, args=(A_meas, k_eq_f, k_eq_r, WB))

        [m_s_est[z], I_yy_est[z], l_f_est[z]] = par_est

    else: # Estimation is zero
        [m_s_est[z], I_yy_est[z], l_f_est[z]] = [0, 0, 0]
        z += 1

np.save('mass_estimate', m_s_est)
np.save('inertia_estimate', I_yy_est)
np.save('cog_estimate', l_f_est)

l_r_est = WB - l_f_est
dif_m = np.abs(m_s_est - m_s)/m_s*100
dif_I_yy = np.abs(I_yy_est - I_yy)/I_yy*100
dif_l_f = np.abs(l_f_est - l_f)/l_f*100

m_s_estimate = np.average(m_s_est[m_s_est != 0])
print('Mass estimate =', m_s_estimate, 'kg')
I_yy_estimate = np.average(I_yy_est[I_yy_est != 0])
print('Inertia estimate =', I_yy_estimate, 'kg*m^2')
l_f_estimate = np.average(l_f_est[l_f_est != 0])
print('COG estimate =', l_f_estimate, 'm')

m_bin = np.linspace(int(round(np.min(m_s_est)-49, -2)), int(round(np.max(m_s_est)+49, -2)), 41)
plt.figure(3)
plt.hist(m_s_est, bins = m_bin)
plt.text(int(round(np.min(m_s_est)-49, -2)), 0.9*plt.ylim()[1], '$\mu$ = {:.2f}'.format(m_s_estimate))
plt.text(int(round(np.min(m_s_est)-49, -2)), 0.8*plt.ylim()[1], '$\sigma$ = {:.2f}'.format(m_s_est.std()))
plt.title('Mass estimates - Laplace')
plt.xlabel('Mass [kg]')
plt.ylabel('Occurance')
plt.savefig('hist_mass')

I_bin = np.linspace(int(round(np.min(I_yy_est)-49, -2)), int(round(np.max(I_yy_est)+49, -2)), 41)
plt.figure(4)
plt.hist(I_yy_est, bins = I_bin)
plt.text(int(round(np.min(I_yy_est)-49, -2)), 0.9*plt.ylim()[1], '$\mu$ = {:.2f}'.format(I_yy_estimate))
```

```
plt.text(int(round(np.min(I_yy_est)-49, -2)), 0.8*plt.ylim()[1], '$\sigma$ = {:.2f}'.format(I_yy_est.std()))
plt.title('Pitch moment of inertia estimates - Laplace')
plt.xlabel('Inertia [kg*m^2]')
plt.ylabel('Occurance')
plt.savefig('hist_inertia')

l_f_bin = np.linspace(round(np.min(l_f_est)-0.049, 1), round(np.max(l_f_est)+0.049, 1), 41)
plt.figure(5)
plt.hist(l_f_est, bins = l_f_bin)
plt.text(round(np.min(l_f_est)-0.049, 1), 0.9*plt.ylim()[1], '$\mu$  = {:.4f}'.format(l_f_estimate))
plt.text(round(np.min(l_f_est)-0.049, 1), 0.8*plt.ylim()[1], '$\sigma$ = {:.4f}'.format(l_f_est.std()))
plt.title('CoG estimates (from front axle) - Laplace')
plt.xlabel('Distance [m]')
plt.ylabel('Occurance')
plt.savefig('hist_cog')

m_s_error = (m_s_est - m_s)/m_s*100
I_yy_error = (I_yy_est - I_yy)/I_yy*100
l_f_error = (l_f_est - l_f)/l_f*100
error = [m_s_error, I_yy_error, l_f_error]

label = ['m_s_est', 'I_yy_est', 'l_f_est']
plt.figure(6)
plt.title('Error in estimated parameters - Laplace')
plt.ylabel('error [%]')
plt.boxplot(error, labels = label)
plt.savefig('Boxplot')

Results = [[stoptime], [vel], [m_s_estimate], [m_s_est.std()], [I_yy_estimate], [I_yy_est.std()], [l_f_estimate], [l_f_est.std()]]

import os
mydir = os.getcwd() # Get directory
updir = os.path.dirname(mydir) # Go up one directory
mydir_new = os.chdir(updir) # Change current directory to updir
np.savetxt('Result_test.txt', Results)

dx = 0.05

plt.figure(8)
yf = fft(x_r_f)
tf = np.linspace(0.0, 1.0/(2.0*dx), N//2)
plt.plot(tf, 2.0/N * np.abs(yf[0:N//2]))
plt.title('Road spatial frequency')
plt.xlabel('Spatial frequency [cycles/m]')
plt.ylabel('FFT Magnitude (power)')


# Plot results
```

```
x_t_f = state[0,:]
x_t_r = state[1,:]
x_s = state[2,:]
phi = state[3,:]

x_s_f = x_s + l_f*phi
x_s_r = x_s - l_r*phi

# Plots
plt.figure(10, figsize=(12, 4)) # Plot road profile and body displacement front
plt.xlabel('t [s]')
plt.ylabel('road profile [m]')
plt.grid(True)
lw = 1
plt.plot(t, x_r_f, 'r', linewidth=lw) #plot road displacement
plt.plot(t, x_t_f, 'b', linewidth=lw) #plot wheel displacement
plt.plot(t, x_s_f, 'g', linewidth=lw) #plot body displacement
plt.legend((r'$x_{road, front}$', r'$x_{tyre, front}$',r'$x_{sprung, front}$'), prop=FontProperties(size=16))
plt.title('Mass Displacements for the\nCoupled Spring-Mass System, front')
plt.savefig('two_springs_f.png', dpi=1000)

plt.figure(11, figsize=(12, 4)) # Plot road profile and body displacement rear
plt.xlabel('t [s]')
plt.ylabel('road profile [m]')
plt.grid(True)
plt.plot(t, x_r_r, 'r', linewidth=lw) #plot road displacement
plt.plot(t, x_t_r, 'b', linewidth=lw) #plot wheel displacement
plt.plot(t, x_s_r, 'g', linewidth=lw) #plot body displacement
plt.legend((r'$x_{road, rear}$', r'$x_{tyre, rear}$',r'$x_{sprung, rear}$'), prop=FontProperties(size=16))
plt.title('Mass Displacements for the\nCoupled Spring-Mass System, rear')
plt.savefig('two_springs_r.png', dpi=1000)

plt.figure(12, figsize=(12, 4)) # Plot body angle phi
plt.xlabel('t [s]')
plt.ylabel('body angle [rad]')
plt.grid(True)
plt.plot(t, phi, 'g', linewidth=lw) #plot x_s
plt.legend((['Pitch angle']), prop=FontProperties(size=16))
plt.title('Body angle over time')
plt.savefig('two_springs.png', dpi=1000)

end = time.time()

print("time elapsed", end - start, "s")
```

A-2-2 Function File

```python
"""
@author: JoÃńl Dijkhuizen
"""
import matplotlib.pyplot as plt
import numpy as np
from sympy import Symbol, Matrix
from Norm import normalize
from scipy.io import loadmat
import math

plt.close("all")

def parameters():
    # Parameter values

    # Masses:
    m_t_f = 40 # Front unsprung mass [kg]
    m_t_r = 65 # Rear unsprung mass [kg]
    m_s = 1000 # Sprung mass [kg]
    m = m_t_f, m_t_r, m_s

    # Spring constants
    k_t_f = 200000 # Tyre stiffness front [N/m]
    k_t_r = k_t_f # Tyre stiffness rear [N/m]
    k_s_f = 45000 # Front suspension stiffness [N/m]
    k_s_r = 80000 # Rear suspension stiffness [N/m]
    k = k_t_f, k_t_r, k_s_f, k_s_r

    c_t_f = 50 # Tyre damping front [N*s/m]
    c_t_r = c_t_f # Tyre damping rear [N*s/m]
    c_s_f = 2800 # Front suspension damping [N*s/m]
    c_s_r = 3500 # Rear suspension damping [N*s/m]
    c = c_t_f, c_t_r, c_s_f, c_s_r

    I_yy = 2000 # Chassis pitch mass moment of inertia [Kg*m^2]
    I = I_yy

    l_s_f = 0.3 #height sprung mass front
    l_s_r = 0.3 #height sprung mass rear
    l_s = l_s_f, l_s_r

    l_f = 1.2 # Distance from front axle to CoG [m]
    l_r = 1.6 # Distance from rear axle to CoG [m]
    l = l_f, l_r

    WB = l_f + l_r #Vehicle wheelbase [m]

    vel = 100/3.6 # velocity
    stoptime = 100 # runtime
    print("Velocity = ", vel*3.6, "km/h")
    print("Measurement time = ", round(stoptime,2), "sec.")
```

```
    L = vel*stoptime # Covered distance
    dx = 0.05 # Spatial frequency

    road_profile = 1 # 0 for free decay, 1 for ISO, 2 for Laplace road
        profile

    if road_profile == 0:
        print("Free decay")
        stoptime = 1
        L = vel*stoptime
        T_s = 0.01
        N = round(stoptime/T_s)
    if road_profile == 1:
        print("ISO Road Profile")
        T_s = 0.01
        N = round(stoptime/T_s)
    if road_profile == 2:
        print("Laplace Road Profile")
        N = round(L/dx)
        T_s = stoptime/N

    var = vel, stoptime, L, T_s, N, WB, road_profile

    return m, k, c, I, l_s, l, var


def road(t_road, var, z):
    vel, stoptime, L, T_s, N, WB, road_profile = var

    dt = WB/vel # Time difference between front and rear [s]
    dindex = int(dt/T_s) # index difference between front and rear wheel
    dL = dindex * T_s * vel # Extra distance covered
    tt = np.linspace(0, stoptime + dindex*T_s, N + dindex)

    if road_profile == 0: # Free decay
        x_r = np.zeros(N + dindex)
        L = 0

    if road_profile == 1: # ISO road profile
        k = 3 # Road roughness ISO 8608
        dn = 1/L
        B = L/N
        n0 = 0.1
        n = np.linspace(dn, (N + dindex)*dn, N + dindex)
        angle = 2*np.pi*np.random.uniform(0, 1, size= N + dindex)
        ampx = np.sqrt(dn)*(2**k)*(1e-3)*(n0/n)

        x_r = np.zeros(N + dindex)

        x = np.linspace(0, L + dL - B, N + dindex)
        tt = np.linspace(0, stoptime + dindex*T_s, N + dindex)

```

```
        x_road = np.linspace(0, L, N)

        for i in range(0, N + dindex):
            x_r[i] = np.dot(ampx, np.sin(2*math.pi*n*x[i] + angle))

    if road_profile == 2: # Laplace road profile
        data = loadmat('road_surface.mat')
        x_r = np.array(data['zLAR'])
        x_r = x_r[:,z]

    x_r_f = np.ravel(x_r[dindex : N + dindex]) # Front road profile
    x_r_r = np.ravel(x_r[0 : N]) # Rear road profile

    dx_r_f = np.zeros(N)
    dx_r_r = np.zeros(N)

    for i in range(1, N - 1):
        dx_r_f[i] = (x_r_f[i+1] - x_r_f[i-1])/(2*T_s)
        dx_r_r[i] = (x_r_r[i+1] - x_r_r[i-1])/(2*T_s)

    plt.figure(7)
    plt.plot(tt, vel*np.ones(N + dindex))
    plt.title('Velocity profile')
    plt.xlabel('Time [s]')
    plt.ylabel('Velocity [m/s]')
    plt.savefig('Velocity_profile')

    return x_r_f, dx_r_f, x_r_r, dx_r_r, L


def solution(state, tt, p, x_r):
    x_t_f, x_t_r, x_s, phi, dx_t_f, dx_t_r, dx_s, dphi = state
    m_t_f, m_t_r, m_s, k_t_f, k_t_r, k_s_f, k_s_r, c_t_f, c_t_r, c_s_f,
        c_s_r, I_yy, l_f, l_r = p
    x_r_f, dx_r_f, x_r_r, dx_r_r, L = x_r

    x_s_f = x_s + l_f * phi
    dx_s_f = dx_s + l_f * dphi
    x_s_r = x_s - l_r * phi
    dx_s_r = dx_s - l_r * dphi


    dstate_0 = [dx_t_f,
                dx_t_r,
                dx_s,
                dphi,
                (-k_t_f*(x_t_f - x_r_f) - c_t_f*(dx_t_f - dx_r_f) + k_s_f
                    *(x_s_f - x_t_f) + c_s_f*(dx_s_f - dx_t_f))/m_t_f,
                (-k_t_r*(x_t_r - x_r_r) - c_t_r*(dx_t_r - dx_r_r) + k_s_r
                    *(x_s_r - x_t_r) + c_s_r*(dx_s_r - dx_t_r))/m_t_r,
                (-k_s_f*(x_s_f - x_t_f) - c_s_f*(dx_s_f - dx_t_f) - k_s_r
                    *(x_s_r - x_t_r) - c_s_r*(dx_s_r - dx_t_r))/m_s,
```

```
                    (-l_f*(k_s_f*(x_s_f - x_t_f) + c_s_f*(dx_s_f - dx_t_f)) + l_r*(k_s_r*(x_s_r - x_t_r) + c_s_r*(dx_s_r - dx_t_r)))/I_yy]

    dstate_0 = np.transpose(dstate_0)

    return dstate_0


def dstate(state, p, t, x_r):
    m_t_f, m_t_r, m_s, k_t_f, k_t_r, k_s_f, k_s_r, c_t_f, c_t_r, c_s_f, c_s_r, I_yy, l_f, l_r = p
    x_r_f, dx_r_f, x_r_r, dx_r_r, L = x_r
    ## Equation of motion: M*ddx + C*dx + K*x = 0, x = [x_t_f, x_t_r, x_s, phi], u = [x_r_f, x_r_r]

    u = [x_r_f,
         x_r_r,
         dx_r_f,
         dx_r_r]
    u = np.matrix(u)

    M = [[ m_t_f, 0,      0,    0    ],
         [ 0,     m_t_r, 0 ,    0    ],
         [ 0,     0,      m_s, 0    ],
         [ 0,     0,      0,    I_yy ]]

    K = [[ k_t_f + k_s_f,  0,              -k_s_f,               -l_f*k_s_f                 ],
         [ 0,              k_t_r + k_s_r,  -k_s_r,                l_r*k_s_r                 ],
         [-k_s_f,          -k_s_r,          k_s_f + k_s_r,         l_f*k_s_f - l_r*k_s_r     ],
         [-l_f*k_s_f,       l_r*k_s_r,      l_f*k_s_f - l_r*k_s_r, l_f**2*k_s_f + l_r**2*k_s_r ]]

    C = [[ c_t_f + c_s_f,  0,              -c_s_f,               -l_f*c_s_f                 ],
         [ 0,              c_t_r + c_s_r,  -c_s_r,                l_r*c_s_r                 ],
         [-c_s_f,          -c_s_r,          c_s_f + c_s_r,         l_f*c_s_f - l_r*c_s_r     ],
         [-l_f*c_s_f,       l_r*c_s_r,      l_f*c_s_f - l_r*c_s_r, l_f**2*c_s_f + l_r**2*c_s_r ]]

    B = [[   0,             0,             0,             0            ],
         [   0,             0,             0,             0            ],
         [   0,             0,             0,             0            ],
         [   0,             0,             0,             0            ],
         [ -k_t_f/m_t_f,    0,            -c_t_f/m_t_f,   0            ],
         [   0,           -k_t_r/m_t_r,    0,           -c_t_r/m_t_r ],
         [   0,             0,             0,             0            ],
         [   0,             0,             0,             0            ]]
```

```
    AA_p1 = np.hstack([np.zeros(shape=(4,4)), np.identity(4)])
    AA_p2 = np.hstack([-(np.dot(np.linalg.inv(M),K)), -(np.dot(np.linalg.inv(M),C))])
    A_int = np.concatenate((AA_p1, AA_p2), axis = 0)

    dstate = np.dot(A_int, state) + np.dot(B, u)

    return dstate


def state_transition(ddx, N, T_s, t):
    ddx_s_f, ddx_s_r = ddx

    gamma = 2 # Tuning parameter
    DoF = 4 # Degrees of Freedom
    M = 2 # Number of measuement stations
    p_0 = 2*DoF//M
    p = gamma * p_0

    arr1 = np.zeros(shape=(M*p, np.size(ddx_s_f)-p))
    i = 0
    k = 0
    for k in range(0, p):
        for i in range(0, np.size(ddx_s_f)-p):
            arr1[M*k,i] = ddx_s_f[i+k]
            arr1[1+M*k,i] = ddx_s_r[i+k]

    arr2 = np.zeros(shape=(M*p, np.size(ddx_s_f)-p))
    i = 0
    k = 0
    for k in range(0, p):
        for i in range(0, np.size(ddx_s_f)-p):
            arr2[M*k,i] = ddx_s_f[i+k+1]
            arr2[M*k+1,i] = ddx_s_r[i+k+1]

    A1 = np.dot(arr2, np.transpose(arr1))
    A2 = np.linalg.inv(np.dot(arr1, np.transpose(arr1)))
    A_Z = np.dot(A1, A2)

    eigenvalues_Z, eigenvectors_Z = np.linalg.eig(A_Z) #Eigenvalues and eigenvectors in Z-plane

    a = np.real(eigenvalues_Z)
    b = np.imag(eigenvalues_Z)

    alpha = np.log(a**2 + b**2)/(2*T_s)
    beta = np.arctan(b/a)/T_s

    eigenvalues_S_noise = alpha + beta*1j #Eigenvalues in S -plane
    eigenvectors_S_noise = eigenvectors_Z

    omega_nS = np.sqrt(alpha**2 + beta**2)
```

```
    omega_nS_hz_noise = omega_nS/(2*np.pi)

    damping_noise = -alpha/omega_nS

    # Cleaning
    eigenvalues_S = eigenvalues_S_noise[np.where(np.imag(eigenvalues_S_noise))[0]] # Remove all non complex eigenvalues

    eigenvectors_S = eigenvectors_S_noise[:,np.where(np.imag(eigenvalues_S_noise))[0]] # Remove all eigenvectors belonging to non complex eigenvalues

    omega_nS_hz = omega_nS_hz_noise[np.where(np.imag(eigenvalues_S_noise))[0]] # Remove all frequencies beloning to the non complex eigenvalues

    #Choose the correct omega's
    index = np.array(np.where((omega_nS_hz >= 1) & (omega_nS_hz <= 3))) # Select the indeces where omega is in range 1-3 Hz

    if np.size(index[0]) == 4:
        index_r = np.where(omega_nS_hz == np.max(omega_nS_hz[index][0]))
        index_f = np.where(omega_nS_hz == np.min(omega_nS_hz[index][0]))

        ind = [index_f[0][0], index_r[0][0]]
        ind_conj = [index_f[0][1], index_r[0][1]]

        omega_f = omega_nS_hz[ind[0]]*2*np.pi
        omega_r = omega_nS_hz[ind[1]]*2*np.pi

        omega = [omega_f, omega_r]

        damping_f = damping_noise[ind[0]]
        damping_r = damping_noise[ind[1]]
        damping = [damping_f, damping_r]

        labda = np.diagflat(eigenvalues_S[ind])
        labda_conj = np.diagflat(eigenvalues_S[ind_conj])

        psi = eigenvectors_S[:, ind]
        psi = psi[[0, 1],:]
        psi_norm = normalize(psi)

        psi_conj = eigenvectors_S[:,ind_conj]
        psi_conj = psi_conj[[0,1],:]
        psi_conj_norm = normalize(psi_conj)

        labda_diag1 = np.hstack([labda, np.zeros(shape=(2, 2))])
        labda_diag2 = np.hstack([np.zeros(shape=(2, 2)), labda_conj])
        labda_diag = np.concatenate((labda_diag1, labda_diag2), axis = 0)

        A_meas1 = np.hstack([psi_norm, psi_conj_norm])
```

```
        A_meas2 = np.hstack([np.dot(psi_norm, labda), np.dot(
            psi_conj_norm, labda_conj)])
        A_meas_nor = np.concatenate((A_meas1, A_meas2), axis = 0)
        A_meas_inv = np.linalg.inv(A_meas_nor)
        A_meas = np.dot(np.dot(A_meas_nor, labda_diag), A_meas_inv)

    else:
        A_meas = np.zeros((4,4))
        omega_nS_hz = 0
        omega = [0, 0]
        index = 0

    return A_meas, omega_nS_hz, omega, index, omega_nS_hz_noise


def estimation(A_meas, k, c, l, omega):
    k_t_f, k_t_r, k_s_f, k_s_r = k
    c_t_f, c_t_r, c_s_f, c_s_r = c
    omega_f, omega_r = omega
    l_f, l_r = l

    WB = l_f + l_r

    # Unknown parameters
    m_s = Symbol('m_s')
    I_yy = Symbol('I_yy')
    l_f = Symbol('l_f')
    l_r = (WB-l_f)

    # Equivalent stiffensses
    #k_eq_f = (k_s_f*k_t_f**2 + k_s_f**2*k_t_f + (omega_f**2)*(k_s_f*
        c_t_f**2 + k_t_f*c_s_f**2))/((k_s_f+k_t_f)**2 + (omega_f**2)*((
        c_s_f + c_t_f)**2))
    #k_eq_r = (k_s_r*k_t_r**2 + k_s_r**2*k_t_r + (omega_r**2)*(k_s_r*
        c_t_r**2 + k_t_r*c_s_r**2))/((k_s_r+k_t_r)**2 + (omega_r**2)*((
        c_s_r + c_t_r)**2))
    k_eq_f = (k_s_f*k_t_f)/(k_s_f+ k_t_f)
    k_eq_r = (k_s_r*k_t_r)/(k_s_r+ k_t_r)
    c_eq_f = (c_s_f*c_t_f)/(c_s_f + c_t_f)
    c_eq_r = (c_s_r*c_t_r)/(c_s_r + c_t_r)

    # Matrices
    M = [[ (l_r/WB)*m_s,  (l_f/WB)*m_s ],
         [ I_yy/WB,       -I_yy/WB       ]]

    M = Matrix(M)
    M_inv = M.inv()

    K_eq = [[ k_eq_f,       k_eq_r      ],
            [ l_f*k_eq_f, -l_r*k_eq_r  ]]

    K_eq = np.matrix(K_eq)

```

```
    C_eq = [[ c_eq_f,       c_eq_r      ],
            [ l_f*c_eq_f, -l_r*c_eq_r ]]

    C_eq = np.matrix(C_eq)

    AA_p1 = np.hstack([  np.zeros(shape=(2,2)),  np.identity(2) ])
    AA_p2 = np.hstack([ -M_inv*K_eq,             -M_inv*C_eq ])
    A_est = np.concatenate((AA_p1, AA_p2), axis = 0)

    cost = (A_meas - A_est)**2

    return A_est, cost, k_eq_f, k_eq_r

def parameter(parameters, A_meas, k_eq_f, k_eq_r, L):
    m_s, I_yy, l_f = parameters

    A_meas1 = np.abs(A_meas[2,0])
    A_meas2 = np.abs(A_meas[2,1])
    A_meas3 = np.abs(A_meas[3,1])

    f1 = (-I_yy*k_eq_f/(L*(-I_yy*l_f*m_s/L**2 - I_yy*m_s*(L - l_f)/L**2))
         - k_eq_f*l_f**2*m_s/(L*(-I_yy*l_f*m_s/L**2 - I_yy*m_s*(L - l_f)/L
        **2)) - A_meas1)**2
    f2 = (-I_yy*k_eq_r/(L*(-I_yy*l_f*m_s/L**2 - I_yy*m_s*(L - l_f)/L**2))
         - k_eq_r*l_f*m_s*(-L + l_f)/(L*(-I_yy*l_f*m_s/L**2 - I_yy*m_s*(L
        - l_f)/L**2)) - A_meas2)**2
    f3 = (-I_yy*k_eq_r/(L*(-I_yy*l_f*m_s/L**2 - I_yy*m_s*(L - l_f)/L**2))
         + k_eq_r*m_s*(-L + l_f)*(L - l_f)/(L*(-I_yy*l_f*m_s/L**2 - I_yy*
        m_s*(L - l_f)/L**2)) - A_meas3)**2

    return f1, f2, f3
```

Bibliography

[1] J. Dijkhuizen, “Online inertial parameter estimation,” August 2018.

[2] S. Bruyne, H. Van der Auweraer, P. Diglio, and J. Anthonis, “Online estimation of vehicle inertial parameters for improving chassis control systems,” *IFAC Proceedings Volumes (IFAC-PapersOnline)*, vol. 18, January 2011.

[3] G. J. Heydinger, R. A. Bixel, N. J. Durisek, E. Yu, and D. A. Guenther, “Effects of loading on vehicle handling,” *SAE Transactions*, vol. 107, pp. 407–415, 1998.

[4] M. Rozyn and N. Zhang, “A method for estimation of vehicle inertial parameters,” *Vehicle System Dynamics*, vol. 48, no. 5, pp. 547–565, 2010.

[5] W. Chu, K. Cao, S. Li, Y. Luo, and K. Li, “Vehicle mass estimation based on high-frequency-information extraction,” *IFAC Proceedings Volumes*, vol. 46, no. 21, pp. 72 – 76, 2013. 7th IFAC Symposium on Advances in Automotive Control.

[6] B. HS, J. Ryu, and C. Gerdes, “Road grade and vehicle parameter estimation for longitudinal control using gps,” *IEEE Transactions on Intelligent Transportation Systems - TITS*, 01 2001.

[7] P. Johannesson and I. Rychlik, “Modelling of road profiles using roughness indicators,” *International Journal of Vehicle Design*, vol. 66, pp. 317–346, January 2014.

[8] K. Bogsjö, K. Podgorski, and I. Rychlik, “Models for road surface roughness,” *Vehicle System Dynamics*, vol. 50, pp. 725–747, May 2012.

[9] N. Zhang and S. Hayama, “Identification of structural system parameters from time domain data : Direct identification of mass, stiffness and damping parameters of a structure,” *JSME international journal. Ser. 3, Vibration, control engineering, engineering for industry*, vol. 34, no. 1, pp. 64–71, 1991.

[10] N. Zhang and S. Hayama, “Identification of structural system parameters from time domain data : Identification of global modal parameters of a structural system by the

improved state variable method," *JSME international journal. Ser. 3, Vibration, control engineering, engineering for industry*, vol. 33, no. 2, pp. 168–175, 1990.

[11] A. Kareem and K. Gurley, "Damping in structures: its evaluation and treatment of uncertainty," *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 59, no. 2, pp. 131 – 157, 1996. Meeting on Structural Damping International Wind Engineering Forum and Additional Papers.

[12] T. L. Kijewski and A. Kareem, "Estimation and modeling of damping and engineering auxiliary damping systems in civil engineering structures : An overview," *NatHaz Modeling Laboratory Report 9.*, 2001.

[13] R. G. Brown and P. Y. C. Hwang, *Introduction to random signals and applied kalman filtering: with MATLAB exercises and solutions; 3rd ed.* New York, NY: Wiley, 1997.

[14] G. Lallement and D. J. Inman, "A Tutorial on Complex Eigenvalues," *Proceedings of the 13th International Modal Analysis Conference*, vol. 2460, p. 490, 1995.

[15] R. Barbosa, "Vehicle vibration response subjected to longwave measured pavement irregularity," *Journal of Mechanical Engineering and Automation*, vol. 2, pp. 17–24, March 2012.

[16] ISO, "Iso 8608:2016 mechanical vibration - road surface profiles - reporting of measured data," standard, International Organization for Standardization, Geneva, CH, November 2016.

[17] D. Cebon, *Handbook of Vehicle-Road Interaction.* Swets & Zeitlinger, 2000. ISBN 90-265-1554-5.

[18] G. Loprencipe, P. Zoccali, and G. Cantisani, "Effects of vehicular speed on the assessment of pavement road roughness," *Applied Sciences*, vol. 9, p. 1783, 04 2019.

[19] P. Múčka, "Simulated road profiles according to iso 8608 in vibration analysis," *Journal of Testing and Evaluation*, vol. 46, p. 20160265, 01 2018.

[20] M. Agostinacchio, D. Ciampa, and S. Olita, "The vibrations induced by surface irregularities in road pavements – a matlab approach," *European Transport Research Review*, vol. 6, pp. 267–275, September 2014.

[21] J. Vandiver, A. B. Dunwoody, R. B. Campbell, and M. F. Cook, "A mathematical basis for the random decrement vibration signature analysis technique," *Journal of Mechanical Design - J MECH DESIGN*, vol. 104, 04 1982.

[22] G. Loprencipe and P. Zoccali, "Use of generated artificial road profiles in road roughness evaluation," *Journal of Modern Transportation*, vol. 25, pp. 24–33, Mar 2017.

[23] J. Tudón-Martínez, F. Soheib, O. Sename, J. Martinez Molina, R. Morales-Menendez, and L. Dugard, "Adaptive road profile estimation in semi-active car suspensions," *IEEE Transactions on Control Systems Technology*, vol. 23, 04 2015.