CHALMERS

Vehicle Motion and Torque Prediction for a Long Haul Truck

Master's Thesis in Systems, Control and Mechatronics

RASMUS HOFWIMMER

DANIEL LIDANDER

Department of Signals and Systems
Division of Automatic Control, Automation and Mechatronics
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2014

Report No. EX012/2014

Vehicle Motion and Torque Prediction for a Long Haul Truck

Master of Sience Thesis in Master's program Systems, Control and Mechatronics

Rasmus Hofwimmer

Daniel Lidander

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Signals and Systems
Division of Automatic Control, Automation and Mechatronics
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2014

Vehicle motion and torque prediction for a long haul truck

Report No. EX012/2014

Department of Signals and Systems
Division of Automatic Control, Automation and Mechatronics
Automatic Control Group
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden

Tel. +46-(0)31 772 1000

Cover: The image is taken from the website `http://images.volvotrucks.com/#1402652697976_12`

Department of Signals and Systems
Gothenburg, Sweden 2014

Vehicle Motion and Torque Prediction for a Long Haul Truck
Master's Thesis in the Master's program in Systems, Control and Mechatronics
RASMUS HOFWIMMER
DANIEL LIDANDER
Department of Signals and Systems
Division of Automatic Control, Automation and Mechatronics
Automatic Control Group
Chalmers University of Technology

Abstract

Global Positioning System (GPS) technology and the detailed knowledge of the road attributes enable to predict the future speed and the torque applied at the wheels of a vehicle, which offer new opportunities for advanced energy management for reduction of the fuel consumption. This thesis concerns predictions of the vehicle motion and the torque applied at the wheels for a long haul heavy truck. The goal is to develop a generic structure of a real-time prediction algorithm.

The proposed prediction algorithm is based on a space-wise model of a closed loop system constituted of a driver and a truck. Predictions are generated by space-wise integrating the closed loop system over a given horizon. The driver model describes both the driver and the powertrain dynamics and is modeled by a proportional and integral controller. The following two approaches for identifying the parameters in the driver model have been investigated: an open loop identification using least squares and a closed loop identification using Moving Horizon Estimation (MHE). The truck model describes the dynamics from the torque at the wheels to the vehicle motion and the parameters are identified with MHE.

The prediction algorithm, without the identification routines, has been successfully implemented in a simulation environment where the model integrator runs in the microsecond range. The algorithm is able to predict the vehicle speed and the torque applied at the wheels when the driver, in the simulation environment, uses a cruise controller.

The MHE scheme for identifying the truck model parameters successfully estimates the aerodynamic drag coefficient, the rolling friction coefficient and the average wind speed. The open loop identification method for identifying the driver model parameters proved to be very difficult, since the dynamics of the driver and the powertrain had a much higher degree of complexity than the driver model itself. Therefore, this method is not favorable in this context. The closed loop identification has not yet been fully investigated. The main difficulty is the fact that the integrated error state is not available for measurement, which is needed for making the parameters observable.

Keywords: Long Haul Truck, Prediction, Moving Horizon Estimation, Parameter Identification

Acknowledgements

We would like to give our deepest gratitude to our supervisor Assistant Professor Sébastien Gros who supported us with great guidance and excellent ideas throughout the project. We also want to thank our supervisors Olof Lindgärde and Stefan Börjesson at Volvo Group Trucks Technology for valuable discussions and giving us insight in the automotive industry. Finally, we want to thank Mario Zanon for his great advise and helpful thoughts.

Gothenburg, June 2014

Rasmus Hofwimmer & Daniel Lidander

Contents

Acronyms

EKF Extended Kalman Filter

EU European Union

GPS Global Positioning System

GSP Global Simulation Platform

KKT Karush Kuhn Tucker

LCT Legacy Code Tool

MHE Moving Horizon Estimation

ODEs Ordinary Differential Equations

QP Quadratic Program

RTI Real-Time Iteration

SQP Sequential Quadratic Programming

Nomenclature

A_f	Vehicle front area.
C_d	Aerodynamic drag coefficient.
C_{roll}	Rolling resistance coefficient.
g	Gravitational acceleration.
J_{wheel}	Inertia of the wheel.
m	Mass of the truck.
ρ_a	Air density.
r_{wheel}	Radius of the wheel.
s	The position of the truck.
v	The speed of the truck.
W	Wind speed.
ω	Angular velocity of the wheel.

1

Introduction

In the total cost of owning a commercial vehicle the fuel consumption constitutes a significant part. A small decrease of the fuel consumption can therefore generate substantial cost savings in the long term. This is particularly appreciable for long haul heavy trucks due to their extensive fuel consumption, [1]. Ongoing technological research investigates advanced control methods which are intended to improve the fuel consumption in commercial vehicles.

GPS technology and the detailed knowledge of the road attributes enable to predict the future speed and the torque applied at the wheels of a vehicle. The predictions offer new opportunities for advanced energy management and for further reductions of the fuel consumption.

1.1 Background

The work presented in this thesis was conducted at Volvo Group Trucks Technology (Volvo GTT) and is a part of an European Union (EU) project called Convenient, [2]. The aim of Convenient is to reduce the fuel consumption for long-haul heavy trucks by developing a suit of innovative energy-saving solutions. A central part of the project is to incorporate predictive control strategies that utilize GPS data describing the topography and attributes of the road ahead.

Volvo GTT is currently making efforts in developing advanced energy control strategies that rely on predictions of the vehicle speed and the torque applied at the wheels. Many difficulties in developing an efficient prediction algorithm have been identified in previous research projects. One key difficulty is the fact that the future speed and torque are highly dependent on the driver's behavior. Additionally, it may vary depending on weather conditions, vehicle load, traffic situations or if the driver uses a cruise controller. Another difficulty is to quantify the quality of the predictive information since it always comes with a degree of uncertainty. For instance, when approaching a traffic light it is

not possible to know a long time in advance if it will be green or red which results in high uncertainty.

1.2 Objective

The objective of this master's thesis is to develop an algorithm that predicts the vehicle motion and the torque applied at the wheels for a long-haul heavy truck. The aim is to develop a generic structure of a real-time prediction algorithm. A prototype of the prediction algorithm should be implemented in the Global Simulation Platform (GSP) described in Section 1.4.1.

1.3 Limitations

In order to attain a reasonable scope of this thesis, the following limitations have been made:

- The prediction algorithm is only intended for long haul drives, i.e. having constant and high speeds for long periods of time.
- The prediction algorithm will only account for speed limits and road inclinations. For example, road attributes such as traffic lights, roundabouts and curves will not be considered.
- The prediction algorithm will not consider surrounding traffic.

1.4 Development tools

The work in this thesis has been conducted using MATLAB/Simulink and C. This section present the tools that have been used in the development of the prediction algorithm.

1.4.1 Global simulation platform

The GSP is a simulation tool in MATLAB/Simulink developed at Volvo GTT, which is often used for evaluating fuel consumption and performance for trucks. The components of the GSP are illustrated in Figure 1.1 and include:

- Road/Environment data which is GPS-like data that consists of speed limits, road altitudes and other road events for specific drive cycles. The data is position based which means that each attribute has a corresponding position. In this thesis it will hereinafter be referred to as *road map data* and will only include speed limits and road altitudes. The road map data is illustrated in Appendix A.

- Two driver model modes:
  - A driver model with a look-ahead functionality.
  - A cruise controller model.
- An advanced truck model describing the combustion engine, power transmission and chassis dynamics.

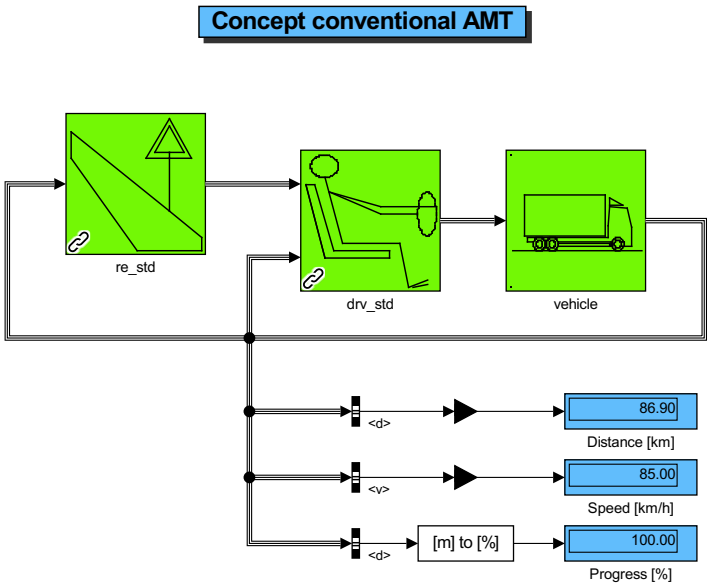

Figure 1.1.: Illustrates the components in the GSP environment.

1.4.2 CVX optimization software

CVX is a modeling language implemented in MATLAB and solves disciplined convex programs. According to [3] it supports a number of standard problem types such as, linear and quadratic problems, semidefinite programs and second order cone programs. CVX can also solve more complex convex problems including e.g., the non-differentiable l_1-norm. The CVX tool provides a convenient way of modeling and solving constrained norm minimization problems which is used in this thesis.

1.5 Method

This section describes the approach of designing the prediction algorithm. It will briefly explain the ideas behind the work presented in this thesis. Some of the details will be intentionally left out to simplify the reading at this stage.

First consider the block diagram in Figure 1.2 which depicts a closed loop system when driving a truck. The system is divided into three parts: a driver, a powertrain and a truck. The driver uses what he currently sees on the road together with the vehicle speed

to deliver a pedal position. The pedal position is then transformed to applied torque at the wheels via the powertrain.

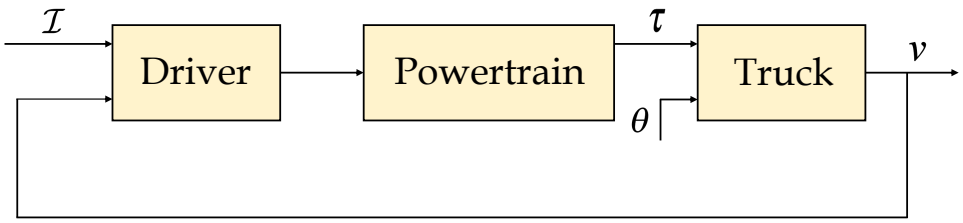

Figure 1.2.: Illustrates the physical closed loop system when driving a truck. $\mathcal{I}$ corresponds to what the driver currently sees on the road, τ is the torque at the wheels, θ is the current road slope and v is the vehicle speed.

The approach to design the prediction algorithm is to derive a model representation of the physical system. In order to simplify the modeling, the powertrain is included in the driver model. This means that it is assumed that the driver outputs the torque applied at the wheels instead of the pedal position, which is indeed a very rough assumption. The model representation, which is illustrated in Figure 1.3, is used to generate predictions by simulating the system.

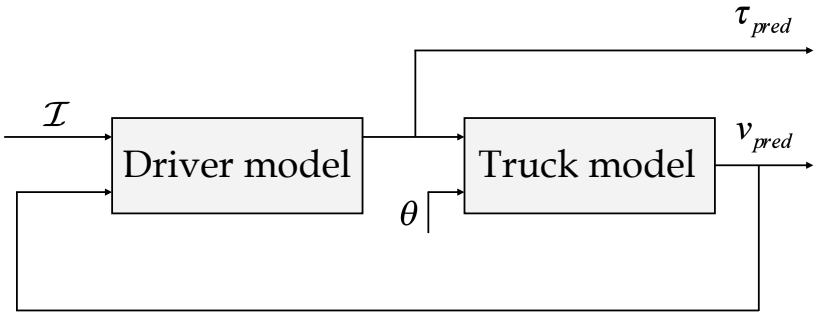

Figure 1.3.: Model used for predicting the speed and the torque applied at the wheels.

To preform predictions, it is necessary to have access to the parameters in the driver and the truck model. For this purpose, two online parameter identification routines will be developed for the driver and the truck model respectively. The motivation to use online techniques is to be able to account for possible variations in the parameters.

The structure of the prediction algorithm is illustrated in Figure 1.4. The parameters in the driver and the truck model are identified by performing corresponding identification routine. The parameters are then forwarded to the model integrator which utilizes the parameters together with the processed road map data, including the road attributes, to generate predictions. Since the road map data is based on position, the predictions will be performed in space which will be further explained in Chapter 5. It is important to

remark that the parameter identification routines are time based, which is emphasized in Table 1.1. The motivation to use time based identification routines is that the sensor measurements are usually sampled in time.

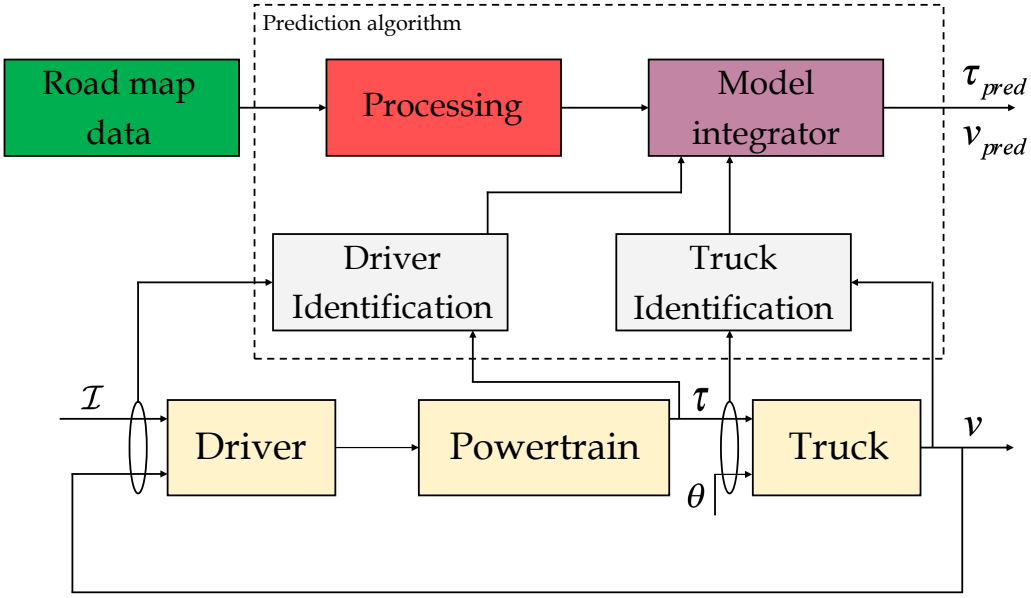

Figure 1.4.: Illustrates the prediction algorithm.

Table 1.1.: Shows which components that are time based or space based.

Time based	Space based
Truck model identification	Road map data
Driver model identification	Model integrator

1.6 Thesis outline

In the second chapter a truck model which describes the dynamics from the torque at the wheels to the vehicle motion is derived. The discretization method that is used throughout the thesis is also presented.

The third chapter concerns the identification of the truck model parameters. The identification is conducted using MHE which will be described thoroughly.

The fourth chapter involves the driver modeling and identification. The driver behavior has been modeled with a proportional and integral controller (PI-controller). The following identification techniques to obtain the parameters in the driver model have been investigated:

- Open loop identification using a least squares approach
- Closed loop identification using MHE

The nature of these identification techniques have enforced the use of two slightly different driver models which will be further explained in the chapter.

The fifth chapter involves the prediction algorithm. The time based dynamic models presented in previous chapters will here be transformed into space. The chapter will reveal how the road map data is used together with the space based models to generate predictions. An implementation of the prediction algorithm in the GSP will also be presented. The implementation does however not include any parameter identification routines. In the end of the chapter an analysis of the uncertainty of the predictions is carried out.

In the sixth chapter, the results are presented. The results are mainly based on measurements extracted from the GSP. In the seventh chapter a discussion of the work presented in the thesis is conducted. In the eighth chapter, the work is concluded. The ninth chapter presents potential improvements and other ideas that can be further investigated in a future work.

2

Mathematical modeling

This chapter presents a mathematical model of a truck which will be used in the prediction algorithm. The model describes the dynamics from the torque at the wheels to the vehicle motion. The chapter will also present a discretization method that is used throughout the thesis.

2.1 Truck model

The model is derived with the assumption that the truck can be represented as a moving mass on an inclined plane, which is illustrated in Figure 2.1.

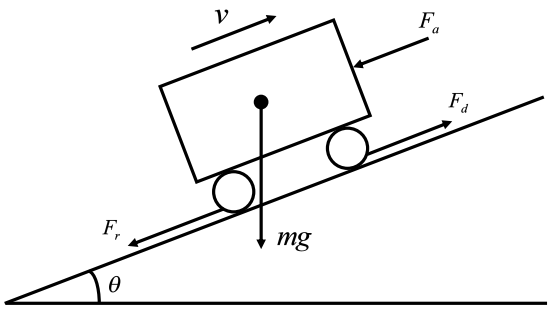

Figure 2.1.: Illustrates the forces the truck is exposed to.

Applying Newton's law yields

$$F_d - F_r - F_a - mg\sin\theta = m\ddot{s}, \tag{2.1}$$

where $F_r = C_{roll}mg\cos\theta$ is the rolling friction, $F_a = \frac{1}{2}C_d\rho_a A_f(v-W)^2$ is the aerodynamic drag and F_d is the driving force.

Assume that the driving wheels can be represented as one wheel and consider the free body diagram in Figure 2.2.

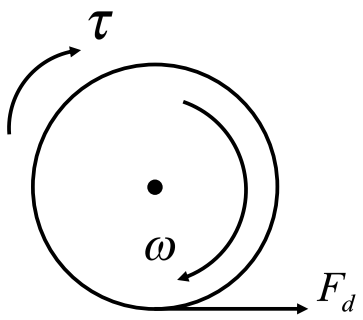

Figure 2.2.: Free body diagram of the wheel.

Summing up the torques at the wheel yields

$$\tau - F_d r_{wheel} - J_{wheel}\dot{\omega} = 0. \tag{2.2}$$

Assuming that $\dot{\omega} = \frac{\ddot{s}}{r_{wheel}}$ and reformulating the expression gives

$$J_{wheel}\frac{\ddot{s}}{r_{wheel}} = \tau - F_d r_{wheel}. \tag{2.3}$$

By combining (2.1) and (2.3), the following nonlinear state space model is obtained:

$$\begin{aligned} \dot{s} &= v \\ \dot{v} &= \frac{1}{m + \frac{J_{wheel}}{r_{wheel}^2}}\left(\frac{\tau}{r_{wheel}} - C_{roll} mg\cos\theta - \frac{1}{2}C_d\rho_a A_f (v - W)^2 - mg\sin\theta\right) \end{aligned} \tag{2.4}$$

2.2 Discretization method

Throughout this thesis a first-order forward Euler approximation will be used as discretization method, namely:

$$x_{k+1} = \underbrace{x_k + \Delta t f(x_k)}_{=F(x_k)} \tag{2.5}$$

Note that much better discretization methods have been proposed, for example the Runge-Kutta method which is described in [4], but for sake of simplicity the forward Euler method is used.

3

Truck parameter identification

This chapter involves the identification of the truck model parameters. The identification will be performed using MHE, [5]. The concept of MHE will be throughly described in the following subsections.

There are several parameters available in the model and only a subset of these are estimated. In this thesis, the following parameters are estimated: C_{roll}, C_d, A_f and W. Since C_d and A_f are in the same term in (2.4), they can not be uniquely determined. Instead, the compound constant ($C_d A_f$) is estimated.

3.1 Motivation to use MHE

The goal with this section is to motivate the usage of MHE. There exist multiple different nonlinear estimators e.g. Bayesian estimation, model inversion, MHE and Extended Kalman Filter (EKF). According to [6], EKF has attracted the most interest due to its relative simplicity and demonstrated efficacy in managing nonlinear systems. However, the EKF gives best results when the nonlinear system is almost linear. For highly complex problems, the EKF is at most an ad hoc solution with many difficulties from a practical point of view. This is indeed a good reason to motivate the use of MHE. The MHE scheme is an efficient online optimization based strategy for parameter estimation which has the ability to include constraints and alternative norms in the optimization problem.

3.2 Moving horizon estimation

The MHE uses a finite time window that assembles past and current measurements. The window is sliding forward in time and is referred to as a moving horizon. This is illustrated in Figure 3.1, where only the past N measurements are considered.

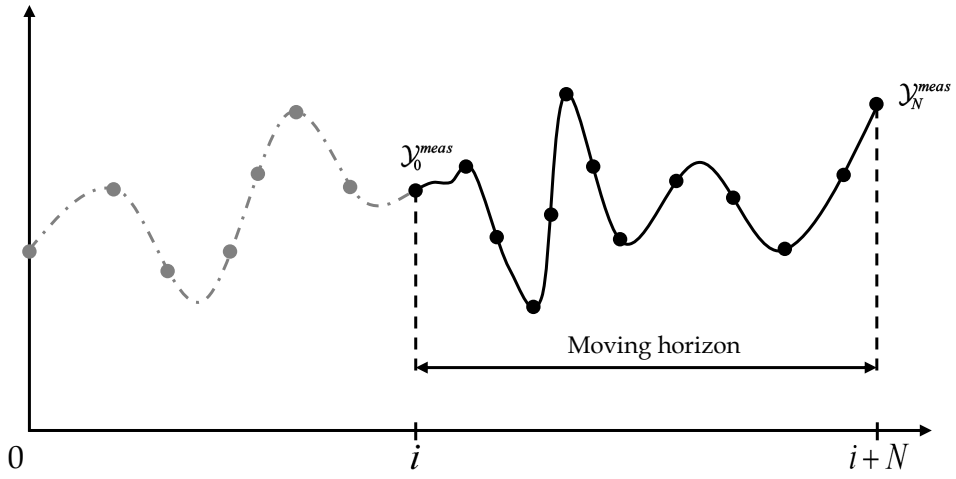

Figure 3.1.: Illustrates the moving horizon in MHE where $i + N$ corresponds to current time.

Assume that vectors with previous input and output measurements $\{\mathcal{Y}_0^{meas}, ..., \mathcal{Y}_N^{meas}\}$ are available at each time instant. We want to find a parameter trajectory $\{\rho_0, ..., \rho_{N-1}\}$ that solves an optimization problem aimed at minimizing the error between the model and the measurements, i.e.

$$\begin{aligned} \min_{x,u,\theta,\rho} \quad & A(\rho_0 - \bar{\rho}_0, x_0 - \bar{x}_0) + \sum_{k=0}^{N-1} l(\mathcal{Y}_k^{meas} - \mathcal{Y}_k(x_k, u_k)) + \bar{l}(\mathcal{Y}_N^{meas} - \mathcal{Y}_N(x_N)) \\ \text{s.t.} \quad & x_{k+1} = F(x_k, u_k, \rho_k) \quad k = 0, ..., N-1, \end{aligned} \tag{3.1}$$

where A is the arrival cost which summarizes the previous information of the parameters and the initial states, l is the stage cost which measures the model error, $\bar{l}$ is the terminal cost which measures the model error on the current state. For the estimation to be physically meaningful, the system model (F) is used to form the equality constraints. The optimization problem can be stacked into a more compact form

$$\begin{aligned} \min_{z} \quad & \phi(z) \\ \text{s.t.} \quad & g(z) = 0, \end{aligned} \tag{3.2}$$

where

$$z = \begin{bmatrix} x_0 & u_0 & \rho_0 & .. & x_{N-1} & u_{N-1} & \rho_{N-1} & x_N \end{bmatrix}, \tag{3.3}$$

$$\phi(z) = A(\rho_0, \bar{\rho}_0, x_0, \bar{x}_0) + \sum_{k=0}^{N-1} l(\mathcal{Y}_k^{meas} - \mathcal{Y}_k(x_k, u_k)) + \bar{l}(\mathcal{Y}_N^{meas} - \mathcal{Y}_N(x_N)), \tag{3.4}$$

$$g(z) = \begin{bmatrix} x_1 - F(x_0, u_0, \rho_0) \\ x_2 - F(x_1, u_1, \rho_1) \\ \vdots \\ x_N - F(x_{N-1}, u_{N-1}, \rho_{N-1}) \end{bmatrix}. \tag{3.5}$$

This problem can be solved using Sequential Quadratic Programming (SQP), [7], which is described in the sequent section.

3.3 Sequential quadratic programming

The SQP solves nonlinear programs on the form (3.2). The algorithm solves a sequence of optimization problems, each of which minimizes a quadratic cost function subject to linear constraints. The quadratic cost function is obtained by using the Gauss-Newton Hessian approximation on (3.4) at an initial guess z. The linear constraints are obtained by simply linearizing the constraint g at z. The Quadratic Program (QP) is as follows

$$\begin{aligned} \min_{\Delta z} \quad & \frac{1}{2}\Delta z^T H \Delta z + f^T \Delta z \\ \text{s.t.} \quad & g(z) + \bigtriangledown g^T(z)\Delta z = 0 \end{aligned} \tag{3.6}$$

where $H = \nabla^2\phi(z)$ and $f = \nabla\phi(z)$. The result is used to update the initial guess as $z = z + \gamma\Delta z$, where $\gamma \in]0, 1]$ is the step size, and the procedure is repeated until convergence. To sum up, the SQP algorithm has the following steps:

1. Collect past measurement data and give an initial guess z
2. SQP
    a) Linearize g at z to get ∇g and evaluate f, H
    b) Solve QP (3.6)
    c) Find step size $\gamma \in]0, 1]$ (globalization)
    d) Update guess $z \leftarrow z + \gamma\Delta z$
    e) Back to a) until convergence
3. Back to 1)

3.3.1 Real time iteration

As mentioned above, the SQP algorithm solves a sequence of quadratic optimization problems. However, it is not always necessary to iterate the algorithm to convergence. Very often the first SQP step provides a sufficiently good approximation of the solution. Hence, in practice a Real-Time Iteration (RTI) approach is often sucessfully used,where a single full step ($\gamma = 1$) per sampling instant is performed, [8]. The RTI scheme can be summarized as follows:

1. Collect past measurement data and give an initial guess z
2. RTI
    a) Linearize g at z to get ∇g and evaluate f, H
    b) Solve QP (3.6)
    c) Update guess $z \leftarrow z + \Delta z$
3. Back to 1)

3.3.2 Solution to the quadratic subproblem

The solution to QPs are particularly simple when they are subject to equality constraints only. To find the solution to (3.6) one can use the Lagrangian function, namely

$$\mathcal{L}(\Delta z, \lambda) = \frac{1}{2}\Delta z^T H \Delta z + f^T \Delta z + \lambda^T (g(z) + \nabla g^T(z)), \tag{3.7}$$

where λ corresponds to the Lagrangian multipliers. From the Karush Kuhn Tucker (KKT) conditions [7], we know that Δz^* is globally optimal if $\exists \lambda^*$ such that

$$\nabla_{\Delta z} L(\Delta z^*, \lambda^*) = H\Delta z^* + f + \nabla g(z)\lambda^* = 0 \tag{3.8}$$

$$\nabla_{\lambda} L(\Delta z^*, \lambda^*) = \nabla g(z) + \nabla g(z)^T \Delta z^* = 0 \tag{3.9}$$

$$\nabla_{\Delta z \Delta z} L(\Delta z^*, \lambda^*) = H \succeq 0 \tag{3.10}$$

Equations (3.8) and (3.9) can be used to form the KKT system

$$\underbrace{\begin{bmatrix} H & \nabla g(z) \\ \nabla g^T(z) & 0 \end{bmatrix}}_{\text{KKT matrix}} \begin{bmatrix} \Delta z^* \\ \lambda^* \end{bmatrix} = - \begin{bmatrix} f \\ g(z) \end{bmatrix}. \tag{3.11}$$

Using the approximation $H \approx \nabla^2 \phi(z)$ and using $f = \nabla \phi(z)$, the solution to the KKT system boils down to

$$\begin{bmatrix} \Delta z^* \\ \lambda^* \end{bmatrix} = \begin{bmatrix} \nabla^2 \phi(z) & \nabla g(z) \\ \nabla g^T(z) & 0 \end{bmatrix}^{-1} \begin{bmatrix} -\nabla \phi(z) \\ -g(z) \end{bmatrix}, \tag{3.12}$$

which holds if the KKT matrix is invertible. The solution is therefore obtained via a single matrix factorization, [9]. The KKT matrix is highly structured and its sparsity pattern is illustrated in Figure 3.2. The structure can be exploited for implementing an efficient matrix factorization.

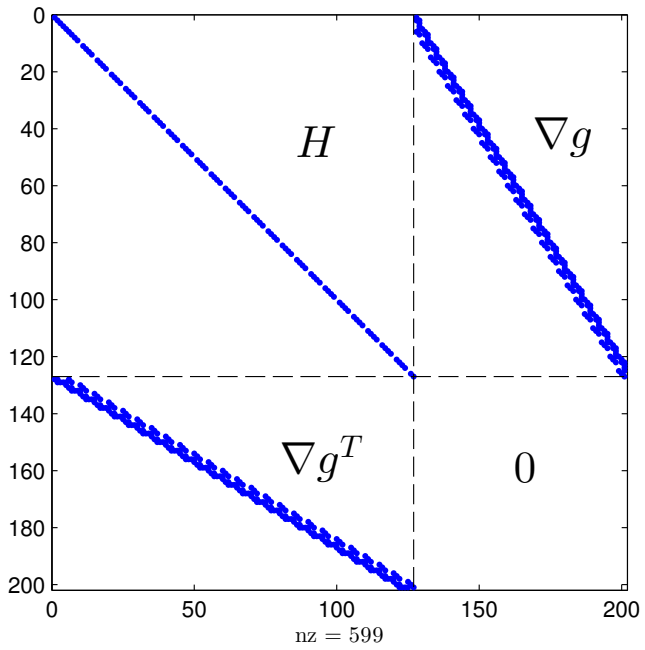

Figure 3.2.: Visualizes the sparsity pattern of the KKT matrix.

3.4 MHE problem for truck model parameter estimation

The goal of the estimator is to find the parameters that results in the best possible fit between the mathematical model and the physical system. As mentioned in Section 3.2, the objective function should measure the error between the model and the physical system. For this purpose the L_2-norm is used, namely

$$\phi(z) = \|\rho_0 - \bar{\rho}_0\|^2_{Q_A} + \|x_N - x_N^{meas}\|^2_{Q_x} + \sum_{k=0}^{N-1} \|\mathcal{Y}_k - \mathcal{Y}_k^{meas}\|^2_Q, \tag{3.13}$$

where

$$\mathcal{Y}_k = \begin{bmatrix} x_k \\ \tau_k \\ \theta_k \end{bmatrix}, \quad \mathcal{Y}_k^{meas} = \begin{bmatrix} x_k^{meas} \\ \tau_k^{meas} \\ \theta_k^{meas} \end{bmatrix}, \tag{3.14}$$

$$Q = \text{diag}(\begin{bmatrix} Q_x & Q_u & Q_\theta \end{bmatrix}). \tag{3.15}$$

Note that $\mathcal{Y}_k$ includes states and inputs from the model (2.4) at time k and $\mathcal{Y}_k^{meas}$ includes measured states and inputs at time k. The vector $\bar{\rho}_0$ is the parameter estimation from the previous MHE execution.

The equality constraints are constructed from the discrete time model derived in Chapter 2. To obtain a more sparse structure when linearizing the constraints, it is assumed that the parameters are constant over the horizon, i.e., $\rho_0 = ... = \rho_{N-1}$. This gives the following equality constraints:

$$g(z) = \begin{bmatrix} x_1 - F(x_0, \tau_0, \theta_0, \rho_0) \\ \rho_1 - \rho_0 \\ x_2 - F(x_1, \tau_1, \theta_1, \rho_1) \\ \rho_2 - \rho_1 \\ \vdots \\ x_N - F(x_{N-1}, \tau_{N-1}, \theta_{N-1}, \rho_{N-1}) \end{bmatrix} = 0 \tag{3.16}$$

Finally, the complete MHE problem for parameter estimation becomes

$$\begin{aligned} \min_z \ & \|\rho_0 - \bar{\rho}_0\|^2_{Q_A} + \|x_N - x_N^{meas}\|^2_{Q_x} + \sum_{k=0}^{N-1} \|\mathcal{Y}_k - \mathcal{Y}_k^{meas}\|^2_Q \\ \text{s.t.} \quad & g(z) = 0, \end{aligned} \tag{3.17}$$

where

$$z = \begin{bmatrix} x_0 & \tau_0 & \theta_0 & \rho_0 & .. & x_{N-1} & \tau_{N-1} & \theta_{N-1} & \rho_{N-1} & x_N \end{bmatrix}.$$

3.4.1 Quadratic subproblem

In order to solve the optimization problem (3.17) using the SQP/RTI algorithm, one needs to construct the quadratic subproblem as described in Section 3.3. For convenience, the

quadratic subproblem is repeated here:

$$\begin{aligned} \min_{\Delta z} \quad & \frac{1}{2}\Delta z^T H \Delta z + f^T \Delta z \\ \text{s.t.} \quad & g(z) + \bigtriangledown g^T(z)\Delta z = 0 \end{aligned} \tag{3.18}$$

The Hessian is obtained by the Gauss-Newton Hessian approximation and will thus have the structure

$$H = \nabla^2 \phi(z) = \begin{bmatrix} Q & & & & \\ & Q & & & \\ & & Q & & \\ & & & \ddots & \\ & & & & Q_x \end{bmatrix}. \tag{3.19}$$

We especially note that the gradient of the constraints has the structure

$$\nabla g^T = \begin{bmatrix} -A_0 & -B_0 & -C_0 & -D_0 & I & & & & & & \\ & & & -I & & I & & & & & \\ & & -A_1 & -B_1 & -C_1 & -D_1 & I & & & & \\ & & & & & -I & & & I & & \\ & & & & & & \ddots & & & \ddots & \\ & & & & & & & -A_{N-1} & -B_{N-1} & -C_{N-1} & -D_{N-1} & I \end{bmatrix}, \tag{3.20}$$

where $A_k = \frac{\partial F(x,u,\theta,\rho)}{\partial x}\Big|_{x_k,u_k,\theta_k,\rho_k}$, $B_k = \frac{\partial F(x,u,\theta,\rho)}{\partial u}\Big|_{x_k,u_k,\theta_k,\rho_k}$, $C_k = \frac{\partial F(x,u,\theta,\rho)}{\partial \theta}\Big|_{x_k,u_k,\theta_k,\rho_k}$, $D_k = \frac{\partial F(x,u,\theta,\rho)}{\partial \rho}\Big|_{x_k,u_k,\theta_k,\rho_k}$ are the sensitivities at stage k. The sparsity pattern of (3.20) is illustrated in Figure 3.3.

The sensitivities and the penalties for each stage are computed offline using automatic differentiation via the symbolic toolbox in MATLAB. The sensitivities are embedded in the model integrator which has been coded in C. It is also important to stress that the structure of (3.19) and (3.20) will always be the same. Hence, the stages of these matrices are generated offline and re-evaluated at each SQP iteration.

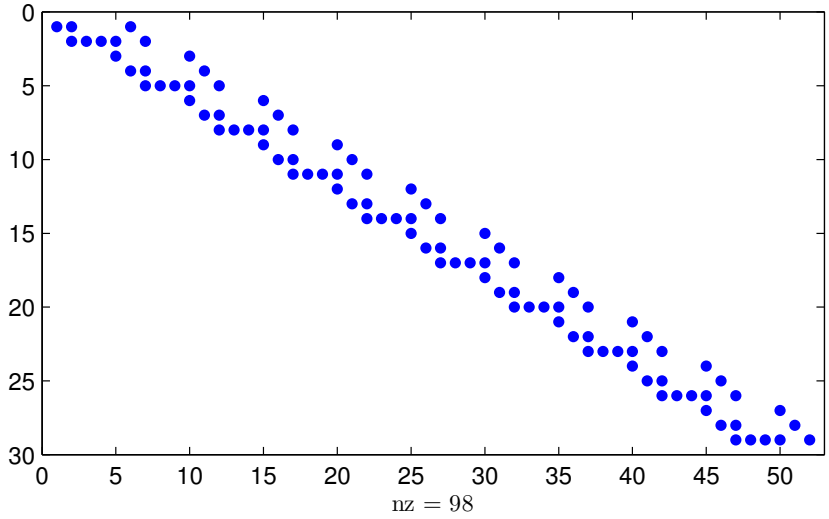

Figure 3.3.: Visualizes the sparsity pattern of the ∇g^T.

3.4.2 Covariance of the MHE

Lets assume that the SQP algorithm has reached the optimal solution, i.e. $\Delta z = 0$. Using the fact that the measurements only enter in the gradient, one can introduce a small pertubation in the measurements which gives

$$\Delta\nabla\phi(z) = Q_A\Delta\bar{\rho}_0 + Q_x\Delta x_N^{meas} + \sum_{k=0}^{N-1} Q\Delta\mathcal{Y}_k^{meas} = H\Delta\mathcal{Y}^{meas}, \tag{3.21}$$

where

$$\Delta\mathcal{Y}^{meas} = \left[\, \Delta{\mathcal{Y}^{meas}}_0^T \;\Delta\bar{\rho}_0\; \Delta{\mathcal{Y}^{meas}}_1^T \;0\; ...\; {\mathcal{Y}^{meas}}_2^T \;0\; ...\; ...\; \Delta{\mathcal{Y}^{meas}}_{N-1}^T \;0\; ...\; \Delta\hat{x}_N^T \,\right]^T. \tag{3.22}$$

The KKT system then becomes

$$\begin{bmatrix} H & \nabla g \\ \nabla g^T & 0 \end{bmatrix} \begin{bmatrix} \Delta z \\ \lambda \end{bmatrix} = \begin{bmatrix} H\Delta\mathcal{Y}^{meas} \\ 0 \end{bmatrix}. \tag{3.23}$$

We define $\mathcal{N}$ as the nullspace of ∇g^T, i.e:

$$\nabla g^T\mathcal{N} = 0 \tag{3.24}$$

From the second row in equation (3.23) we get

$$\nabla g^T\Delta z = 0. \tag{3.25}$$

Note that $\mathcal{N}$ is a basis for Δz, which implies that Δz can be expressed as a linear combination of $\mathcal{N}$, i.e:

$$\Delta z = \mathcal{N}\xi \tag{3.26}$$

Further, the first row in (3.23) gives

$$H\Delta z + \nabla g\lambda = H\Delta\mathcal{Y}^{meas}. \tag{3.27}$$

Then left multiplying with $\mathcal{N}$ and inserting (3.26) yields

$$\mathcal{N}^T H\mathcal{N}\xi + \underbrace{\mathcal{N}^T\nabla g\lambda}_{=0} = \mathcal{N}^T H\Delta\mathcal{Y}^{meas}. \tag{3.28}$$

By solving for ξ and multiplying with $\mathcal{N}$, the following expression is obtained:

$$\Delta z = \mathcal{N}(\mathcal{N}^T H\mathcal{N})^{-1}\mathcal{N}^T H\Delta\mathcal{Y}^{meas} \tag{3.29}$$

Lets assume that $\mathbb{E}\{(\Delta\mathcal{Y}^{meas})^T(\Delta\mathcal{Y}^{meas})\} = \mathbb{E}\{(\mathcal{Y}^{meas})^T(\mathcal{Y}^{meas})\}$, where

$$\mathcal{Y}^{meas} = \left[\, {\mathcal{Y}^{meas}}_0^T \;\bar{\rho}_0\; {\mathcal{Y}^{meas}}_1^T \;0\; ...\; {\mathcal{Y}^{meas}}_2^T \;0\; ...\; ...\; {\mathcal{Y}^{meas}}_{N-1}^T \;0\; ...\; \hat{x}_N^T \,\right]^T \tag{3.30}$$

corresponds to the real measurements. If the penalties are selected such that

$$H^{-1} = \mathbb{E}\{(\mathcal{Y}^{meas})^T(\mathcal{Y}^{meas})\}, \tag{3.31}$$

then the covariance of the deviation variables can be computed as

$$\begin{aligned}\mathbb{E}\{\Delta z^T \Delta z\} &= \mathbb{E}\{\mathcal{N}(\mathcal{N}^T H \mathcal{N})^{-1}\mathcal{N}^T H \Delta\hat{Y}\Delta\hat{Y}^T H\mathcal{N}(\mathcal{N}^T H\mathcal{N})^{-1}\mathcal{N}^T\} \\ &= \mathcal{N}(\mathcal{N}^T H\mathcal{N})^{-1}\mathcal{N}^T H \underbrace{\mathbb{E}\{(\Delta\mathcal{Y}^{meas})^T(\Delta\mathcal{Y}^{meas})\}}_{H^{-1}} H\mathcal{N}(\mathcal{N}^T H\mathcal{N})^{-1}\mathcal{N}^T \\ &= \mathcal{N}(\mathcal{N}^T H\mathcal{N})^{-1}\mathcal{N}^T H\mathcal{N}(\mathcal{N}^T H\mathcal{N})^{-1}\mathcal{N}^T \\ &= \mathcal{N}(\mathcal{N}^T H\mathcal{N})^{-1}\mathcal{N}^T.\end{aligned} \tag{3.32}$$

If the measurement noise is sufficiently small it is feasible to assume that

$$\mathbb{E}\{z^T z\} \approx \mathbb{E}\{\Delta z^T \Delta z\} = \mathcal{N}(\mathcal{N}^T H\mathcal{N})^{-1}\mathcal{N}^T. \tag{3.33}$$

This means that we now have a method for computing the covariance of the MHE. Recall that the MHE estimates the states, the inputs and the parameters. By assessing the covariance of the estimation it is possible to propagate it through the prediction algorithm to analyze the covariance of the predictions, which will be explored in Section 5.4.1.

3.4.3 Penalty parameter selection

The penalty parameter matrix Q is a diagonal matrix where each element determines how much the residual error in the different quantities affects the total cost.

$$Q = \begin{bmatrix} q_s & & & \\ & q_v & & \\ & & q_\tau & \\ & & & q_\theta \end{bmatrix}. \tag{3.34}$$

The Hessian is selected according to (3.31), which implies that the penalties for each stage are computed by

$$Q = (\mathbb{E}\{(\mathcal{Y}_k^{meas})^T(\mathcal{Y}_k^{meas})\})^{-1}, \tag{3.35}$$

and the arrival cost by

$$Q_A = (\mathbb{E}\{\bar{\rho}_0^T\bar{\rho}_0\})^{-1}. \tag{3.36}$$

An intuitive way of thinking about this type of selection is that the weights reflect how much the measurements are trusted. For example, if the standard deviation in the input signal measurements is very high then q_τ should be low. This means that the matching of the input signal will have a low priority in the optimization problem.

4

Driver parameter identification

This chapter concerns the driver modeling and identification. Before exploring the details, the following assumption has been made in order to simplify the driver modeling procedure.

Typically when a truck is accelerating, the torque at the wheels is saturated at different levels due to the gear shifts which is illustrated in Figure 4.1. To avoid modeling this behavior, the torque is transformed to power which is assumed to have a constant saturation limit.

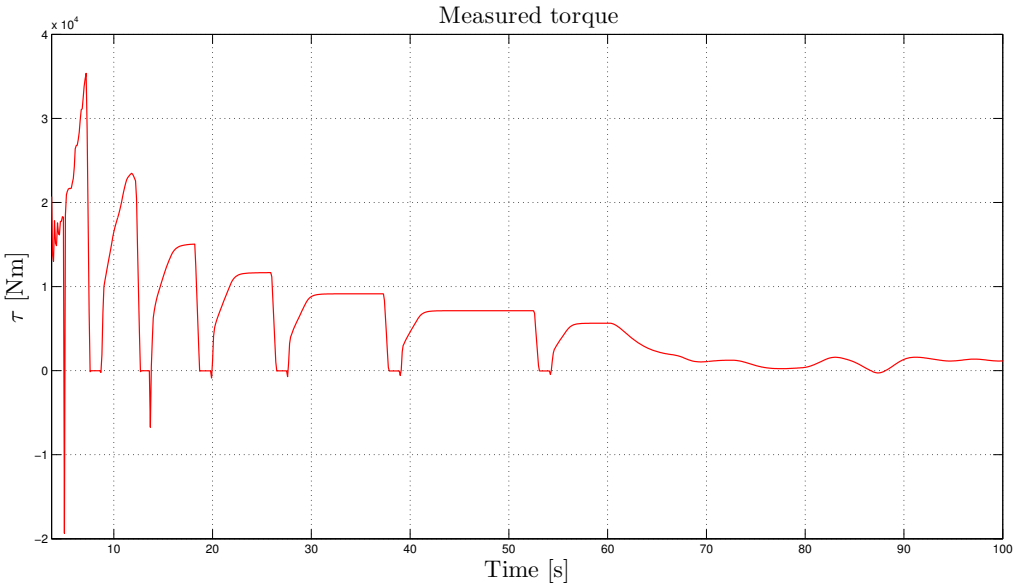

Figure 4.1.: Illustrates the torque at the wheels during an acceleration.

The torque can be expressed as

$$\tau = \frac{p}{\omega} = p\frac{r_{wheel}}{v}. \tag{4.1}$$

Inserting (4.1) into (2.4) leads to a truck model having power as input, namely

$$\begin{aligned} \dot{s} &= v \\ \dot{v} &= \frac{1}{m + \frac{J_{wheel}}{r_{wheel}^2}} \left(\frac{p}{v} - C_{roll} mg \cos\theta - \frac{1}{2} C_d \rho_a A_f (v - W)^2 - mg \sin\theta \right). \end{aligned} \tag{4.2}$$

By making this transformation, the driver model now delivers the power at the wheels instead of the torque at the wheels. The driver behavior is modeled by a PI-controller that maps the speed error to the power at the wheels. This will be further explained in the following subsections.

Recall from Figure 1.4 that the prediction algorithm should include a routine for identifying the parameters in the driver model. As a first approach, an open loop identification technique using least squares was considered. Experiments revealed that this technique was intrinsically hard to accomplish, which led to testing a closed loop identification technique using MHE instead. The difference between the concept of open loop and closed loop is illustrated in Figure 4.2.

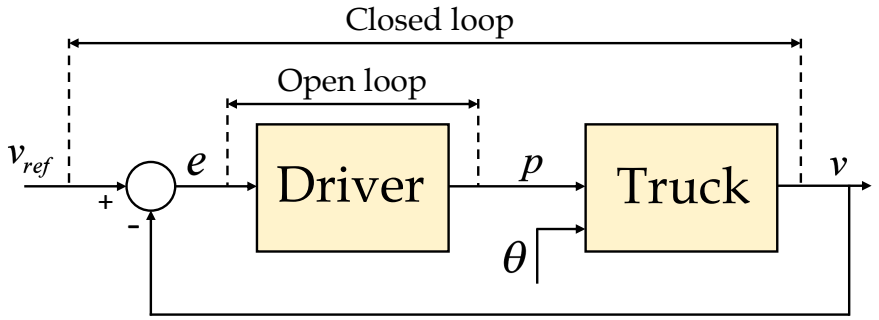

Figure 4.2.: Illustrates the concept of open loop identification and the closed loop identification. The open loop identification is based on the signals e and p and the closed loop identication is based on the signals v_{ref}, v and θ.

The fact that two different identification methods were used, enforced two slightly different driver models. To emphasize this, two models with corresponding identification techniques are considered:

- Driver model #1 - Open loop identification
- Driver model #2 - Closed loop identification

4.1 Driver model #1

The first driver model is a PI-controller with an ad hoc anti-windup solution that stops integrating when the saturated signal becomes less or equal to the unsaturated signal, see Figure 4.3. The driver model can be described with the equations

$$\begin{aligned} e(t) &= v_{ref}(t) - v(t) \\ p(t) &= K_p e(t) + K_i I(t) \\ p_{sat}(t) &= \text{sat}(p(t)), \end{aligned} \tag{4.3}$$

where the update of the integrator is given by

$$\dot{I}(t) = \begin{cases} 0, & p(t) > p_{max} \\ e(t), & -p_{max} < p(t) < p_{max} \\ 0, & p(t) < -p_{max}, \end{cases} \tag{4.4}$$

and saturation function is defined as

$$\text{sat}(x) = \begin{cases} p_{max}, & x > p_{max} \\ x, & -p_{max} < x < p_{max} \\ -p_{max}, & x < -p_{max}. \end{cases} \tag{4.5}$$

Note that $v_{ref}(t)$ corresponds to the current speed limits on the road. Hence, the driver is assumed to follow the speed limits. The parameters in the model will be identified with the open loop identification method described in Section 4.1.1.

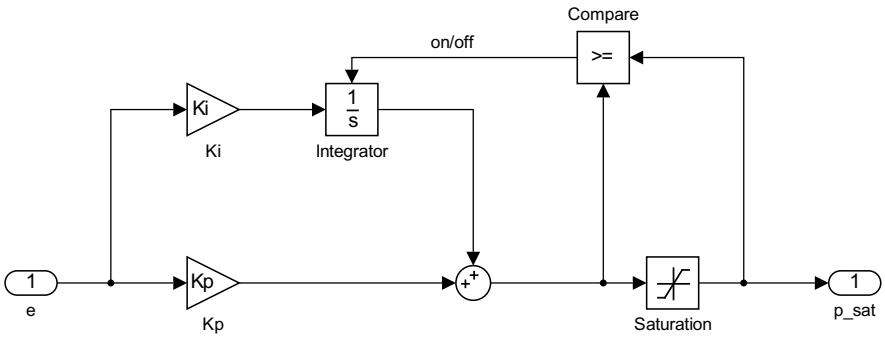

Figure 4.3.: A schematic of driver model #1.

4.1.1 Open loop parameter identification

In order to identify the parameters in driver model #1 in (4.3), an open loop identification routine is developed. The idea is to formulate an optimization problem that aims at minimizing the error between the modeled output power and the measured output power. The optimization problem is quite involved and will therefore be explained in several steps. First consider the following least squares problem:

$$\begin{aligned} \min_{K_p,K_i} \quad & \sum_k ||p_k - p_k^{meas}||^2 \\ \text{s.t.} \quad & p_k = K_p e_k^{meas} + K_i \int e_k^{meas} dt \end{aligned} \tag{4.6}$$

When the measured input and output data are fed into the optimization problem, the knowledge about the initial state of the integrator is unknown. In order to handle this problem, the initial state of the integrator is included as a decision variable. The revised least squares problem can now be expressed as:

$$\begin{aligned} \min_{K_p,K_i,I_0} \quad & \sum_k ||p_k - p_k^{meas}||^2 \\ \text{s.t.} \quad & p_k = K_p e_k^{meas} + K_i \big(I_0 + \int e_k^{meas}\big) dt \end{aligned} \tag{4.7}$$

Note that this problem lacks the anti-windup functionality described in 4.1, which needs to be implemented. The reason why this simple form of anti-windup is suitable here is to avoid having more decision variables and nonlinear dynamic constraints. The principle can be conceived by replacing the error in the integral term with

$$\bar{e}_k = \begin{cases} 0, & p_k^{meas} > p_{max} \\ e_k^{meas}, & p_k^{meas} < p_{max} \end{cases}, \tag{4.8}$$

which gives the optimization problem

$$\begin{aligned} \min_{K_p,K_i,I_0} \quad & \sum_k ||p_k - p_k^{meas}||^2 \\ \text{s.t.} \quad & p_k = K_p e_k^{meas} + K_i \big(I_0 + \int \bar{e}_k dt\big). \end{aligned} \tag{4.9}$$

Figure 4.4 shows an example window of measured power and speed error extracted from the GSP. The figure illustrates that the power contains outliers due to gear shifts. One can also assume that the truck changes dynamics when the power is negative, which means that the truck is braking instead of accelerating. These behaviors comprehensively complicate the fitting procedure. To cope with these issues the following modifications are introduced:

- Replacing the L_2-norm with a Huber penalty function to deal with outliers.
- Neglect fitting on negative power.

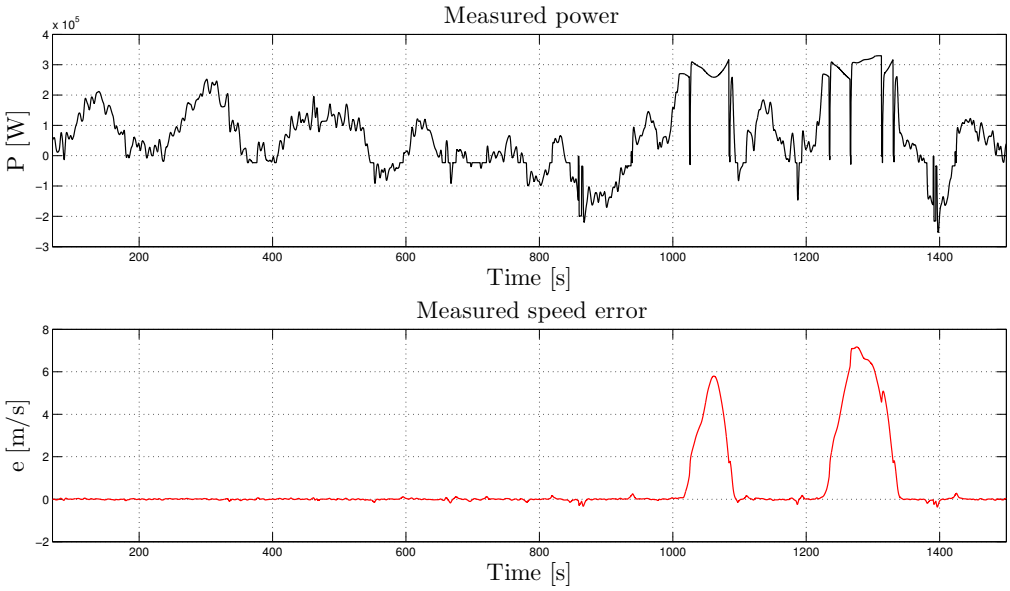

Figure 4.4.: Shows an example window of measured power and speed error extracted from the GSP.

The Huber penalty function in (4.12) has the property to penalize outliers less compared than the standard L_2-norm. By introducing a Huber penalty function in the objective function, the outliers will be of less importance in the optimization problem.

The negative power is not considered in the objective function by multiplying with the following logical expression:

$$\eta = \begin{cases} 1, & p_k^{meas} < p_{max} \quad \& \quad p_k^{meas} > 0 \\ 0, & \text{otherwise} \end{cases} \tag{4.10}$$

The final optimization problem which is used for identifying the driver model parameters now becomes

$$\begin{aligned} \min_{K_p, K_i, I_0} \quad & \sum_k \Phi\big(p_k - p_k^{meas}\big)\eta \\ \text{s.t.} \quad & p_k = K_p e_k^{meas} + K_i\big(I_0 + \int \hat{e}_k dt\big), \end{aligned} \tag{4.11}$$

where Φ corresponds to the Huber penalty function and is according to [3] defined by

$$\Phi(z) = \begin{cases} |z|^2, & |z| \leq 1 \\ 2|z| - 1, & |z| \geq 1 \end{cases} . \tag{4.12}$$

The optimization problem is solved using CVX, described in Section 1.4.2, and the results are shown in section 6.3.

4.2 Driver model #2

The second driver model is also a PI-controller but uses a back-calculation method for anti-windup, which is illustrated in Figure 4.5. The driver model can then be expressed analytically as

$$p_{sat}(t) = \text{sat}(p(t)), \tag{4.13}$$

where

$$p(t) = K_p e(t) + \int_0^t \Big(e(\tau) + \frac{1}{K_b}\big(p_{sat}(\tau) - p(\tau)\big)\Big) d\tau. \tag{4.14}$$

The model can be formulated as

$$\dot{I}(t) = K_i e(t) + \frac{1}{K_b}(p_{sat}(t) - p(t)), \tag{4.15}$$

where

$$\begin{aligned} p(t) &= K_p e(t) + I(t) \\ p_{sat}(t) &= \text{sat}(p(t)) \end{aligned} \tag{4.16}$$

and where the saturation function is defined in (4.5).

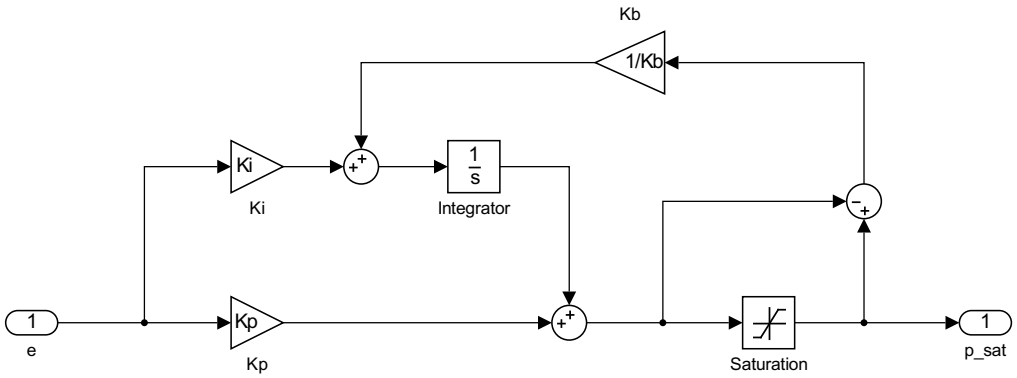

Figure 4.5.: A schematic of driver model #2.

4.2.1 Closed loop parameter identification

This section concerns the closed loop identification using MHE for identifying the driver model parameters. The MHE requires the model dynamics to be expressed with Ordinary Differential Equations (ODEs), therefore is driver model #2 suitable here. However, only some initial attempts have been made and the method is not yet functioning properly.

The closed loop approach is inherently hard since the closed loop system is nonlinear and non-differentiable at some operating points. The non-differentiability comes from the saturation function in the driver model. Since the MHE is solved using linearization, the saturation function needs to be approximated by a smooth function. The approximation is done by

$$p_{sat}(p) \approx (p + p_{max})\frac{\arctan(a(p + p_{max}))}{\pi} - (p - p_{max})\frac{\arctan(a(p - p_{max}))}{\pi}, \tag{4.17}$$

where a is a tuning parameter that determines the smoothness of the function. The saturation function is illustrated in Figure 4.6.

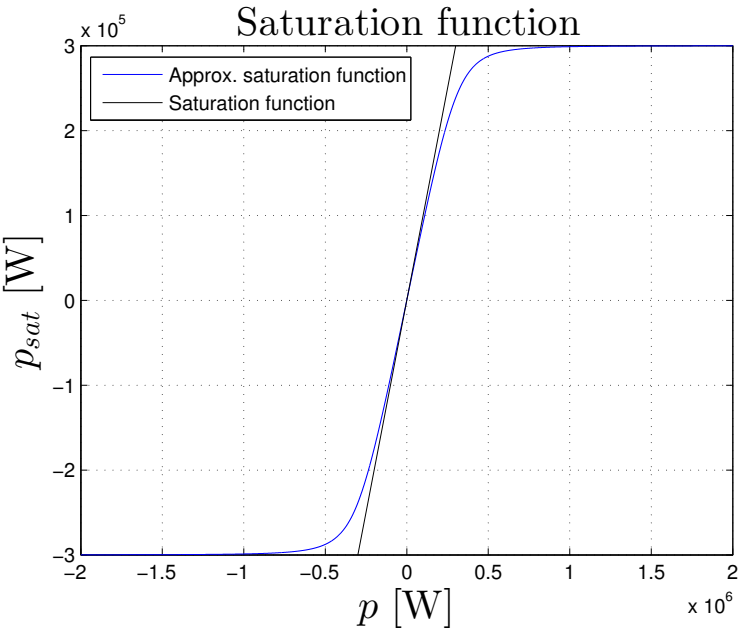

Figure 4.6.: Shows the approximated saturation function when $a = 5 \cdot 10^{-6}$.

By combining (4.2) with (4.15) and (4.16), the closed loop dynamic equations are

$$\begin{aligned}
\dot{s} &= v \\
\dot{v} &= \frac{1}{m + \frac{J_{wheel}}{r_{wheel}^2}}\left(\frac{p_{sat}(p)}{v} - C_{roll}mg\cos\theta - \frac{1}{2}C_d\rho_a A_f(v - W)^2 - mg\sin\theta\right), \\
\dot{I} &= K_i e + \frac{1}{K_b}(p_{sat}(p) - p)
\end{aligned} \tag{4.18}$$

where $p = K_p e + I$ and $e = v_{ref} - v$. The idea is to estimate the parameters in the driver model based on measurements of the inputs (v_{ref} and θ) and the states (s, v and I), see

Figure 4.2. However, the integrator state is not directly available for measurement but is virtually constructed from the other measurements and the arrival parameters from the previous MHE run. The following MHE formulation is used

$$\min_{z} \|\rho_0 - \bar{\rho}_0\|^2_{Q_{\rho_0}} + \|x_0 - \bar{x}_0\|^2_{Q_{x_0}} + \|x_N - x_N^{meas}\|^2_{Q_x} + \sum_{k=0}^{N-1} \|\mathcal{Y}_k - \mathcal{Y}_k^{meas}\|^2_Q \tag{4.19}$$

$$\text{s.t.} \quad g(z) = 0,$$

where

$$\rho_k = \begin{bmatrix} K_{p_k} \\ K_{ik} \\ K_{bk} \end{bmatrix}, \quad x_k = \begin{bmatrix} s_k \\ v_k \\ I_k \end{bmatrix}, \quad \mathcal{Y}_k = \begin{bmatrix} x_k \\ v_{ref_k} \\ \theta_k \end{bmatrix}, \quad \mathcal{Y}_k^{meas} = \begin{bmatrix} x_k^{meas} \\ v_{ref_k}^{meas} \\ \theta_k^{meas} \end{bmatrix}, \tag{4.20}$$

$$z = \begin{bmatrix} x_0 & v_{ref_0} & \theta_0 & \rho_0 & .. & x_{N-1} & v_{ref_{N-1}} & \theta_{N-1} & \rho_{N-1} & x_N \end{bmatrix}$$

$$g(z) = \begin{bmatrix} x_1 - F(x_0, v_{ref_0}, \theta_0, \rho_0) \\ \rho_1 - \rho_0 \\ x_2 - F(x_1, v_{ref_1}, \theta_1, \rho_1) \\ \rho_2 - \rho_1 \\ \vdots \\ x_N - F(x_{N-1}, v_{ref_{N-1}}, \theta_{N-1}, \rho_{N-1}) \end{bmatrix}, \tag{4.21}$$

and integrated error measurements are constructed as

$$I_{k+1}^{meas} = I_k^{meas} + \Delta t \left(\bar{K}_{i_0} e_k^{meas} + \frac{1}{\bar{K}_{b_0}} (p_{sat}(p) - p) \right), \tag{4.22}$$

where

$$p = \bar{K}_{p0} e_k^{meas} + I_k^{meas}. \tag{4.23}$$

Note that $\bar{K}_{p_0}$, $\bar{K}_{i_0}$ and $\bar{K}_{b_0}$ corresponds to the arrival parameters from the previous MHE run. The result is shown in Section 6.4.

5

Prediction

In previous chapters, models for the truck and the driver were presented along with corresponding parameter identification techniques. These models will now be used in this chapter, together with the road map data (illustrated in Appendix A), to perform predictions of the vehicle motion and the torque at the wheels. Recall that the road map data is based on position and the models and identification techniques are based on time, see Table 1.1. Consequently, the predictions will be performed in space. This requires that the time based models are transformed into space which is explained in the following section.

5.1 Space based dynamics

To be able to perform predictions in space, the truck model (4.2) and the driver models presented in Sections 4.1 and 4.2 need to be transformed into space. By utilizing the chain rule, the space derivative of the speed can be expressed as

$$\frac{dv}{ds} = \frac{dv}{dt} v^{-1}. \tag{5.1}$$

Inserting the second state in the truck model (4.2) into (5.1) yields

$$\frac{dv}{ds} = \frac{1}{m + \frac{J_{wheel}}{r_{wheel}^2}} \left(\frac{p}{v} - C_{roll} mg \cos\theta - \frac{1}{2} C_d \rho_a A_f (v - W)^2 - mg \sin\theta \right) \frac{1}{v}. \tag{5.2}$$

Recall that in Chapter 4, two different driver models were presented. These driver models are also transformed into space, in the same manner, by using the space derivative of the integrator. This will thus yield two closed loop systems which are presented in the sequent subsections.

5.1.1 Closed loop system #1

The first closed loop system is obtained by combining driver model #1 described in Section 4.1 with (5.2) which gives the set of equations

$$\begin{aligned}
\frac{dv}{ds} &= \frac{1}{m + \frac{J_{wheel}}{r_{wheel}^2}} \left(\frac{p_{sat}}{v} - C_{roll} mg \cos\theta - \frac{1}{2} C_d \rho_a A_f (v - W)^2 - mg \sin\theta \right) \frac{1}{v} \\
\frac{dI}{ds} &= \begin{cases} 0, & p > p_{max} \\ \frac{e}{v}, & -p_{max} < p < p_{max} \\ 0, & p < -p_{max} \end{cases} \\
\frac{dt}{ds} &= \frac{1}{v},
\end{aligned} \tag{5.3}$$

where the saturated power is expressed as

$$p_{sat} = \begin{cases} p_{max}, & p > p_{max} \\ p, & -p_{max} < p < p_{max} \\ -p_{max}, & p < -p_{max} \end{cases} \tag{5.4}$$

and the unsaturated power is given by

$$p = K_p e + K_i I. \tag{5.5}$$

The last state in equation (5.3) is included as an extra feature to obtain a corresponding time vector when integrating the model.

5.1.2 Closed loop system #2

The second closed loop system is obtained by combining driver model #2 presented in Section 4.2 with (5.2) which gives the set of equations

$$\begin{aligned}
\frac{dv}{ds} &= \frac{1}{m + \frac{J_{wheel}}{r_{wheel}^2}} \left(\frac{p}{v} - C_{roll} mg \cos\theta - \frac{1}{2} C_d \rho_a A_f (v - W)^2 - mg \sin\theta \right) \frac{1}{v} \\
\frac{dI}{ds} &= \left(K_i e + \frac{1}{K_b} (p_{sat} - p) \right) \frac{1}{v} \\
\frac{dt}{ds} &= \frac{1}{v},
\end{aligned} \tag{5.6}$$

where the saturated power is expressed as

$$p_{sat} = \begin{cases} p_{max}, & p > p_{max} \\ p, & -p_{max} < p < p_{max} \\ -p_{max}, & p < -p_{max} \end{cases} \tag{5.7}$$

and the unsaturated power is given by

$$p = K_p e + I. \tag{5.8}$$

5.2 Prediction algorithm

This section involves the algorithm used for predicting the speed and the torque at the wheels. Hereinafter the parameters in the driver model and the truck model will be assumed to be known. Hence, the parameter identification routines explained in previous chapters are excluded from the prediction algorithm which is emphasized in Figure 5.1. As seen in the figure, the prediction algorithm constitutes of two parts. The first part is referred to as the road plan algorithm which provides vectors of the upcoming road slopes and speed limits. The second part is referred to as the model integrator, which uses these vectors to generate predictions by integrating the system from the current initial states.

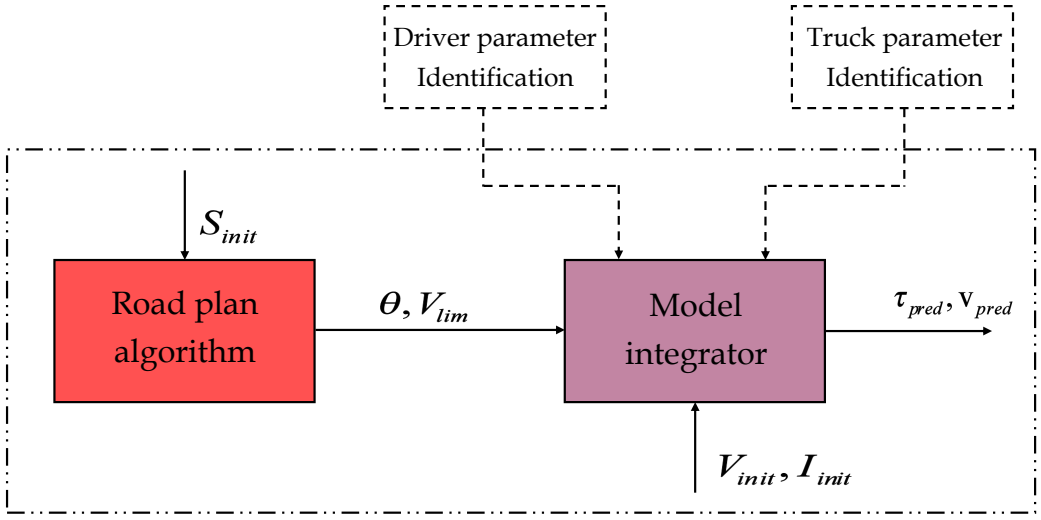

Figure 5.1.: Illustrates the structure of the prediction algorithm when the parameter identification routines are excluded. s_{init}, v_{init} and I_{init} are the initial states of the physical vehicle. Θ and V_{lim} are vectors of upcoming road slope and speed limits. τ_{pred} is the predicted torque at the wheels and v_{pred} is the predicted speed.

5.2.1 Road plan algorithm

The objective of the road plan algorithm is to provide vectors with the upcoming road slope and speed limits from the current position of the vehicle. The algorithm uses the road map data which constitutes of a position vector, a speed limit vector and an altitude vector. For convenience the position vector is denoted $\bar{S}$, the speed limit vector is denoted $\bar{V}$ and the altitude vector $\bar{A}$. The road plan algorithm can be summarized with the following steps:

1. Collect the current position s_{init}.
2. Interpolate $\bar{S}$, $\bar{V}$ and $\bar{A}$ with a desired resolution ds.
3. Differentiate the altitude $\bar{A}$ to obtain the slope.

4. Locate the closest point in $\bar{S}$ to s_{init}.
5. Extract the upcoming speed limits and slope for a given horizon.
6. Output the upcoming slopes Θ and speed limits V_{lim} along with the resolution ds.

5.2.2 Model integrator

The speed and torque predictions are computed by integrating the closed loop system using Θ and V_{lim}, from the road plan algorithm, as inputs. The integration is performed with the forward Euler method described in Section 2.2. A pseudo code of the model integrator is shown in Function 1. Note that, depending on which driver model that is used the model integrator will be slightly different.

The model integrator has been coded in C using the MEX interface in order to gain speed. More information about MEX can be found in [10].

Function 1: Pseudo code for the model integrator.

Input : v_{init}, I_{init}, t_{init}, $\theta[]$, $V_{ref}[]$, K_p, K_i, K_b, p_{max}, ds

Calculate initial step in forward Euler and update initial torque.
// Perform the prediction
for $i := 0, i < length(\theta) - 1, i++$ **do**
    // Error
    $error = V_{ref}(i) - v(i)$;
    // Control law
    Update p according to (5.5) or (5.8)
    // Saturation
    if $p > p_{max}$ **then**
        $p_{sat} = p_{max}$;
    else if $u < -p_{max}$ **then**
        $p_{sat} = -p_{max}$;
    else
        $p_{sat} = p$;
    end
    // State Space Equations
    Update state equations according to (5.3) or (5.6).
    // Apply Forward Euler
    $v(i+1) = v(i) + ds * vdot$;
    $I(i+1) = I(i) + ds * Idot$;
    $t(i+1) = t(i) + ds * tdot$;
    $\tau(i+1) = p_{sat} * r_{wheel}/v(i+1)$;
end

5.3 Implementation in the GSP

This section focuses on the implementation of the prediction algorithm in the GSP. Since the driver models are very similar, only the first driver model presented in Section 4.1 has been implemented in the GSP.

As mentioned previously, the model integrator function has been coded in C. This function has been integrated in Simulink by the use of an S-function. There are several possible ways of creating S-functions based on functions written in C, e.g. writing a wrapper S-function, using an S-function builder block and using the Legacy Code Tool (LCT). The latter is the simplest approach with the feature of auto generating C Mex S-functions from existing C-functions. The LCT has been used in the implementation and more information regarding S-functions and LCT can be found in [11].

The implementation in the GSP is depicted in Figure 5.2. The red block corresponds to the road plan algorithm and the purple block corresponds to the S-function for the model integrator. Note that there is also a light blue block in the figure. The purpose of this block is to estimate the initial condition of the integrated error state since it is not available for measurement. The initial state of the integrator is obtained by simply reformulating the second equation in (4.3) and computing

$$I_k = \frac{p_k - K_p e_k}{K_i}. \tag{5.9}$$

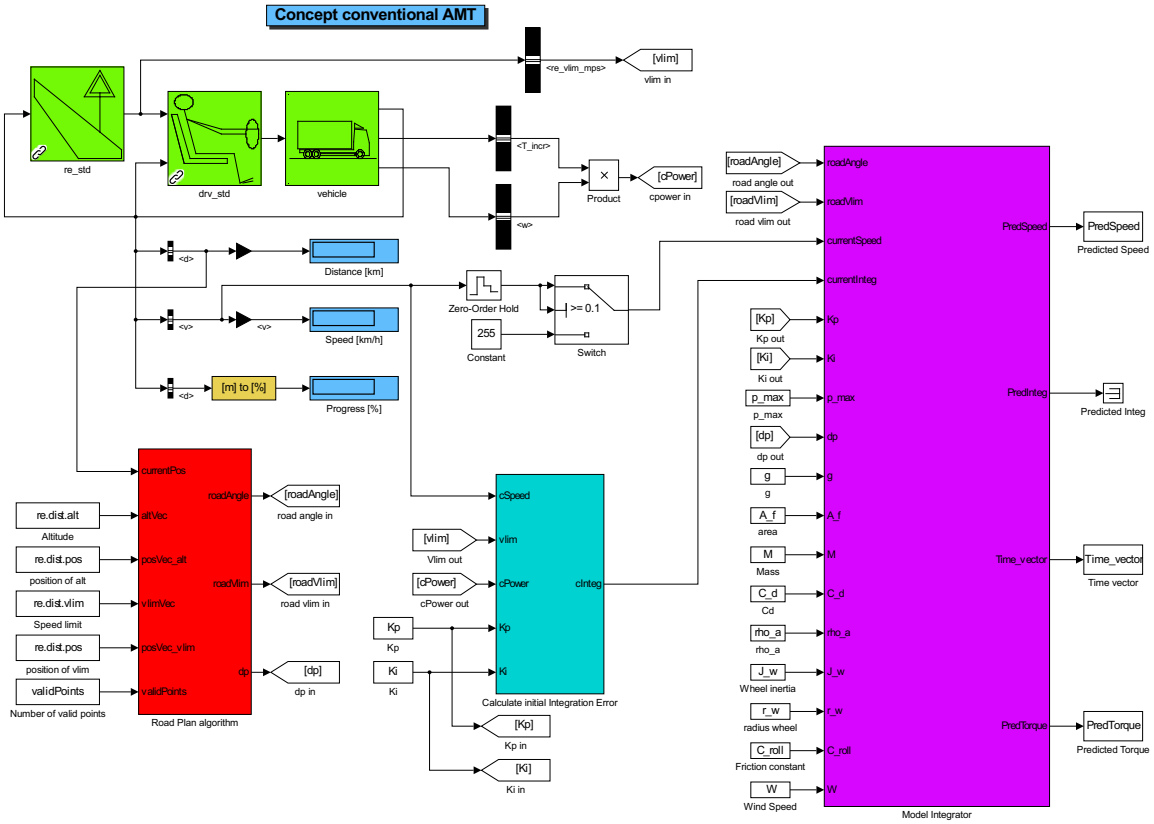

Figure 5.2.: The implementation of the prediction algorithm in the GSP.

5.4 Covariance of the prediction

This section presents an analysis of the covariance of the predictions. The analysis concerns how the covariance in the parameters, states and inputs to the truck model are connected to the covariance of the predictions. It is based on a Monte-Carlo simulation, where several predictions are performed with normally distributed noise added to the parameters, states and inputs.

The Monte-Carlo simulation will reveal that the predictions are normally distributed, which is a property of a linear system (even though the system is nonlinear). Based on this observation, a computationally inexpensive method for estimating the covariance of the predictions is proposed. The method is referred to as a first order uncertainty propagation and has not yet been verified.

5.4.1 Monte-Carlo simulation

The Monte-Carlo simulation was conducted by performing 10 000 predictions with normally distributed noise added to the parameters. The standard deviations of the noises that were used are shown in Table 5.1.

Table 5.1.: Shows the standard deviations of the normally distributed noises used in the Monte-Carlo simulation.

Measured quantity	Mean	Standard deviation	Unit
$C_d A_f$	0	1.29	$[m^2]$
C_{roll}	0	0.0012	[]
I	0	0.30	[]
θ	0	$5.0 \cdot 10^4$	$[rad]$
v	0	0.47	$[m/s]$
W	0	1.25	$[m/s]$

The distribution of the final element in the speed prediction is shown in Figure 5.3 and the distribution of the final element in the torque prediction is shown in 5.4.

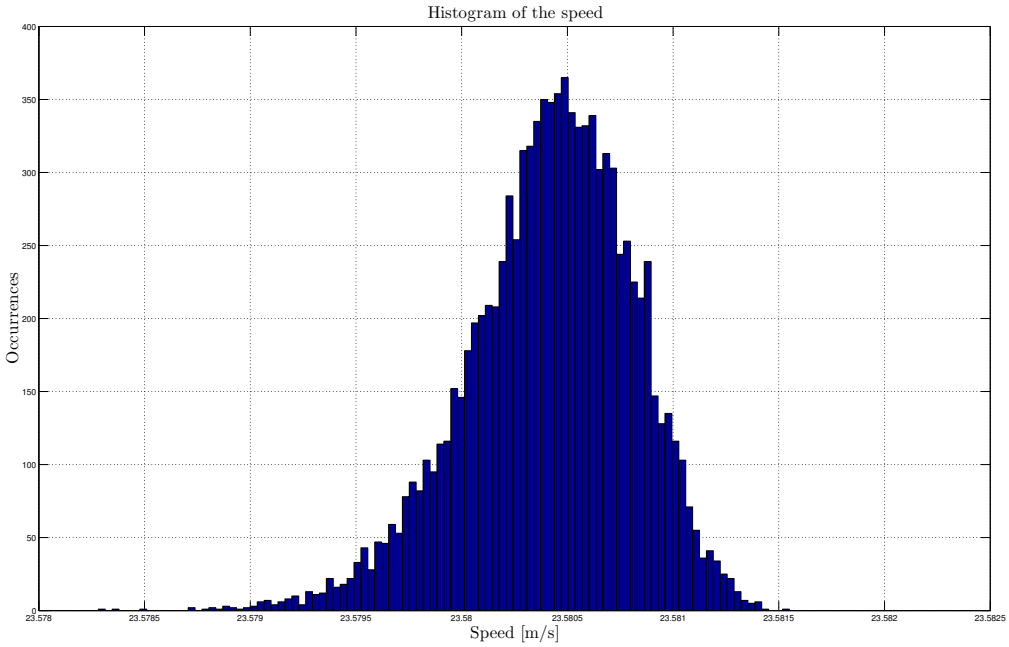

Figure 5.3.: Illustrates the distribution of the final element in the speed prediction obtained after performing a Monte-Carlo simulation with 10 000 predictions.

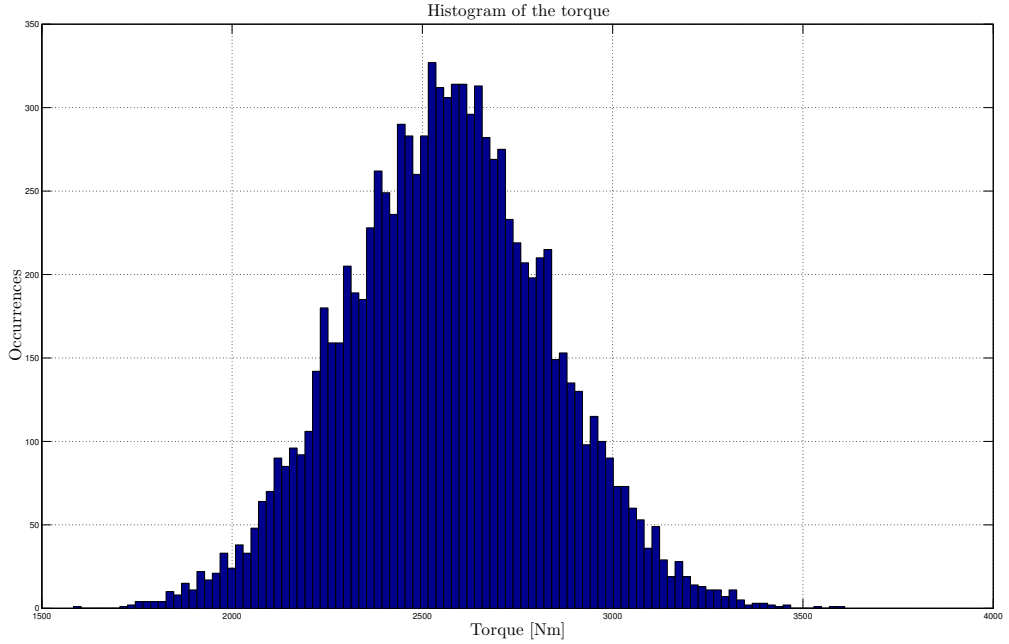

Figure 5.4.: Illustrates the distribution of the final element in the torque prediction obtained after performing a Monte-Carlo simulation with 10 000 predictions.

The distribution of the speed and the torque prediction is evidently close to normal, which means that the prediction algorithm has a behavior close to linear. Hence, it is most likely possible to use a first order uncertainty propagation to compute the covariance of the predictions.

5.4.2 First order uncertainty propagation

This section concerns the first order uncertainty propagation method to compute the covariance of the predictions. The method will be based on the closed loop dynamics (5.6). Since the first order uncertainty propagation method is based on a linearizion of the driver model, the saturation function is approximated with the smooth function described in (4.17). The parameters C_{roll}, C_dA_f and W need to be reformulated as states, which will be motivated further into this subsection. The following discrete time nonlinear state space equations are used

$$\tilde{x}_{k+1} = \underbrace{\tilde{x}_k + dtf(\tilde{x}_k, \theta_k)}_{F(\tilde{x}_k, \theta_k)}, \tag{5.10}$$

where the new state vector is

$$\tilde{x} = \begin{bmatrix} v \\ I \\ C_{roll} \\ C_dA_f \\ W \end{bmatrix} \tag{5.11}$$

and where the continuous dynamics are given by

$$f(\tilde{x}, \theta) = \begin{bmatrix} \frac{1}{m + \frac{J_{wheel}}{r_{wheel}^2}} \left(\frac{p_{sat}}{v} - C_{roll} mg \cos\theta - \frac{1}{2} C_d \rho_a A_f (v - W)^2 - mg \sin\theta \right) \frac{1}{v} \\ \left(K_i e + \frac{1}{K_b} \left(p_{sat} - p \right) \right) \frac{1}{v} \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{5.12}$$

Linearizing the dynamics in (5.10) gives

$$\Delta\tilde{x}_{k+1} = A_k \Delta\tilde{x}_k + B_k \Delta\theta, \tag{5.13}$$

where $A_k = \left. \frac{\partial F(\tilde{x}, \theta)}{\partial \tilde{x}} \right|_{x_k, \theta_k}$ and $B_k = \left. \frac{\partial F(\tilde{x}, \theta)}{\partial \theta} \right|_{x_k, \theta_k}$.

The reason for adding the parameters as states is to keep the matrices A_k and the vectors B_k constant. This allows to easily derive the covariance of the states as

$$\begin{aligned} \sigma^2(\Delta\tilde{x}_{k+1}) =& \mathbb{E}\{\Delta\tilde{x}_{k+1} \Delta\tilde{x}_{k+1}^T\} - \mathbb{E}\{\Delta\tilde{x}_{k+1}\} \mathbb{E}\{\Delta\tilde{x}_{k+1}\}^T \\ =& \mathbb{E}\{ \left(A_k \Delta\tilde{x}_k + B_k \Delta\theta \right) \left(A_k \Delta\tilde{x}_k + B_k \Delta\theta \right)^T \} - \mathbb{E}\{\Delta\tilde{x}_{k+1}\} \mathbb{E}\{\Delta\tilde{x}_{k+1}\}^T \\ =& A_k \sigma^2(\Delta\tilde{x}_k) A_k^T + B_k \sigma^2(\Delta\theta_k) B_k^T. \end{aligned} \tag{5.14}$$

Note that $\tilde{x}_k$ and θ_k is assumed to be uncorrelated and therefore the cross terms disappear. Also note that the expected value of $\Delta\tilde{x}_{k+1}$ equals zero. In a similar manner the covariance can be derived for the torque. Recall that the torque can be expressed as

$$\tau_k = \frac{p_{satk}}{v_k} r_{wheel}. \tag{5.15}$$

Introducing the vector

$$\epsilon_k = \begin{bmatrix} v_k \\ I_k \end{bmatrix} \tag{5.16}$$

and linearizing (5.15) with respect to ϵ_k gives

$$\Delta\tau_k = \frac{\partial \tau_k}{\partial \epsilon_k} \Delta\epsilon_k. \tag{5.17}$$

The covariance of the torque can then be expressed as

$$\sigma^2(\Delta\tau_k) = \mathbb{E}\{\Delta\tau_k \Delta\tau_k^T\} = \frac{\partial \tau_k}{\partial \epsilon_k} \mathbb{E}\{\Delta\epsilon_k \Delta\epsilon_k^T\} \frac{\partial \tau_k}{\partial \epsilon_k}^T. \tag{5.18}$$

6

Results

This chapter concerns the results of the work presented in this thesis. Recall that the parameter identification routines have not yet been implemented in the prediction algorithm. Therefore, the identification routines and the prediction algorithm are validated separately.

6.1 Truck parameter identification using constructed data

Before applying the MHE described in Section 3.4 on GSP measurements, its behavior is evaluated through estimations using constructed data. The data is generated with the truck model derived in Section 2.1 and with known parameters which values are shown in Table 6.1. The evaluation is performed using two sets of data. The first set of data is rich in terms of fluctuations and the second is poor with no fluctuations.

Table 6.1.: Constants used for generating the constructed data.

Parameter	Value	Unit
A_f	9.0	$[m^2]$
C_d	0.53	[]
C_{roll}	$4.8 \cdot 10^{-3}$	[]
g	9.82	$[m/s^2]$
J_{wheel}	14	$[kgm^2]$
m	35737	$[kg]$
ρ	1.18	$[kg/m^3]$
r_{wheel}	0.49	$[m]$
W	5.0	$[m/s]$

The parameters that are used in the MHE scheme are shown in Table 6.2. The sensor noises are assumed to be white Gaussian noises which properties are shown in Table 6.3.

Table 6.2.: Parameter setup for MHE.

Notation	Parameter	Value
Δt	Sample time	0.1
N	MHE Horizon	70
N_{iter}	SQP iterations	1
Q_A	Arrival cost	$0.5I$

Table 6.3.: Properties of the Gaussian distributions used for simulating sensor noise.

Measured quantity	Mean	Standard deviation	Unit
Position	0	7.0	$[m]$
Speed	0	0.5	$[m/s]$
Torque	0	40	$[Nm]$
Slope	0	$1.7 \cdot 10^{-3}$	$[rad]$

The sequent subsections contain many figures and the following summarizes the key observations:

- The parameters C_{roll}, $C_d A_f$ and W can be estimated separately regardless of the richness of the data.
- The true parameter values can not be guaranteed to be found when estimating parameters jointly due to the existence of multiple solutions in the estimation problem.
- The use of poor data does not impact the observability of the parameters.

6.1.1 Estimation using rich data

In this section the MHE are conducted using the rich set of measurement data shown in Figure 6.1. The data set is unrealistic and its purpose is to demonstrate what happens when the data is rich.

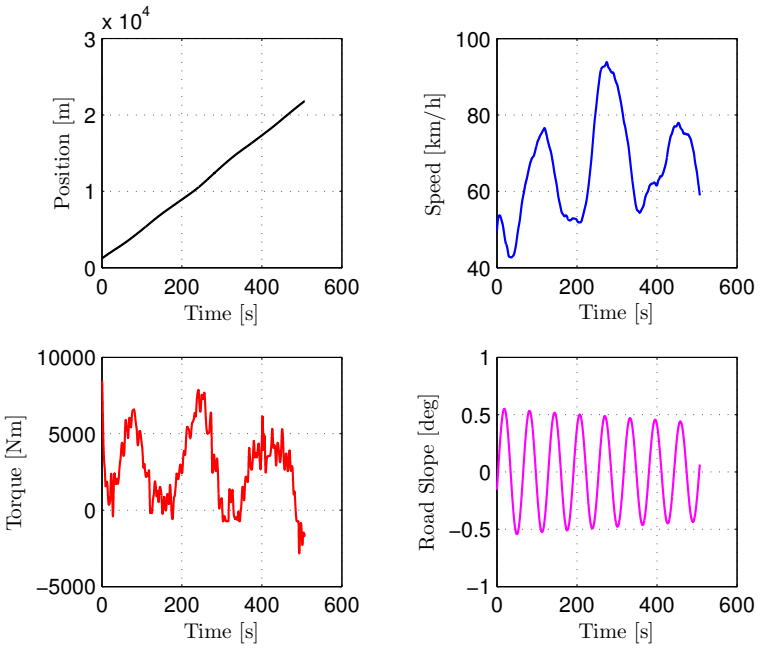

Figure 6.1.: Rich data used for parameter estimation.

The first natural step is to investigate if it is possible to estimate each parameter separately. The outcomes of separately estimating the parameters are shown in Figure 6.2. Clearly, each parameter converges exponentially towards its true value. The wild oscillations in the estimations are due to the measurement noises which creates a randomness.

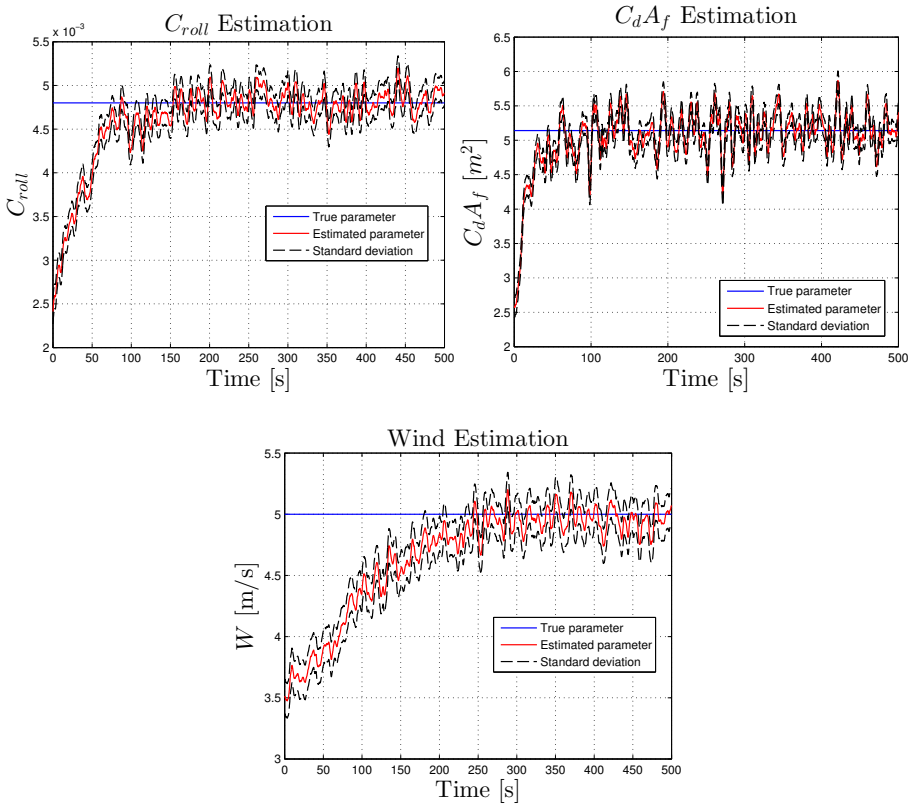

Figure 6.2.: Separate estimation of C_{roll} , C_dA_f and W using rich data.

The results of estimating C_{roll} and C_dA_f jointly are shown in Figure 6.3. The MHE does not manage to converge towards the true parameter values and leaves offset errors in the parameter estimations. It is important to remark here that even though the individual parameters are incorrect the model is still valid, i.e. there exist multiple solutions in the estimation problem.

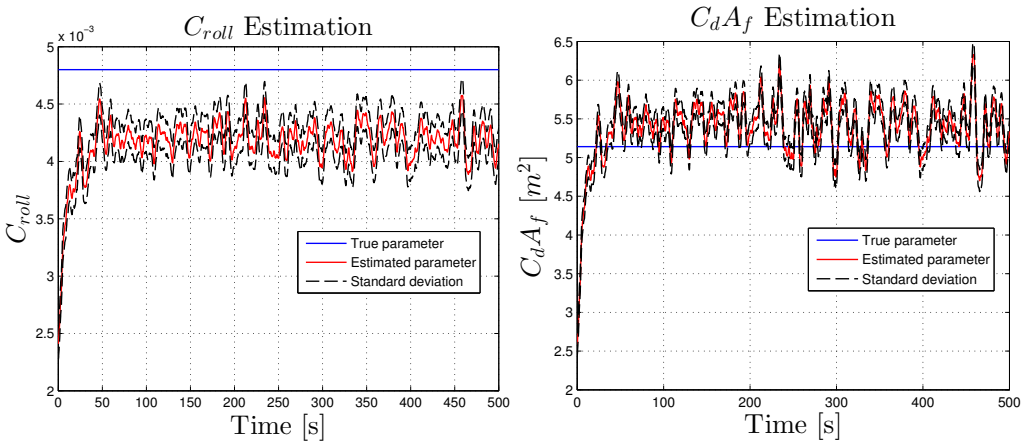

Figure 6.3.: Joint estimation of C_{roll} and C_dA_f using rich data.

Estimating the paramters C_{roll} and W jointly gives the results shown in Figure 6.4. Once again, the MHE does not find the true values of the parameters.

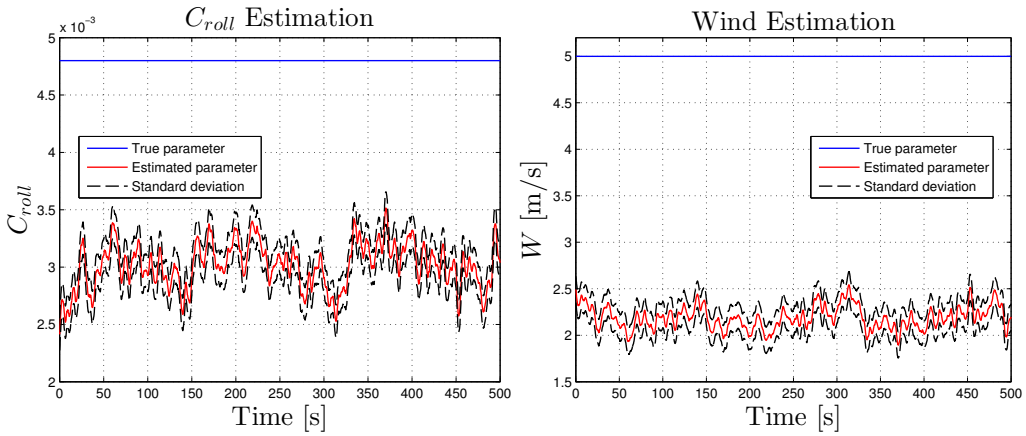

Figure 6.4.: Joint estimation of C_{roll} and W using rich data.

The results of jointly estimating $C_d A_f$ and W are shown in Figure 6.5.

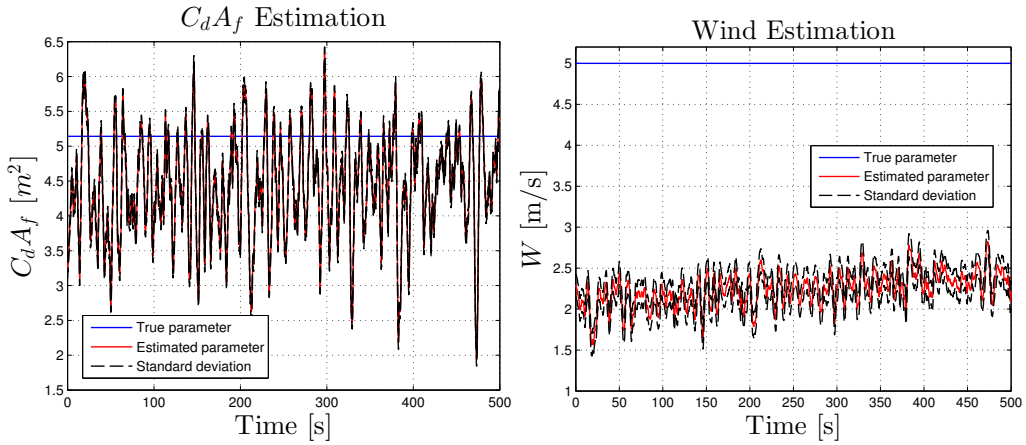

Figure 6.5.: Joint estimation of $C_d A_f$ and W using rich data.

Finally, the results of estimating all the parameters jointly are shown in Figure 6.6. Not suprisingly, the MHE does not converge towards the true parameter values.

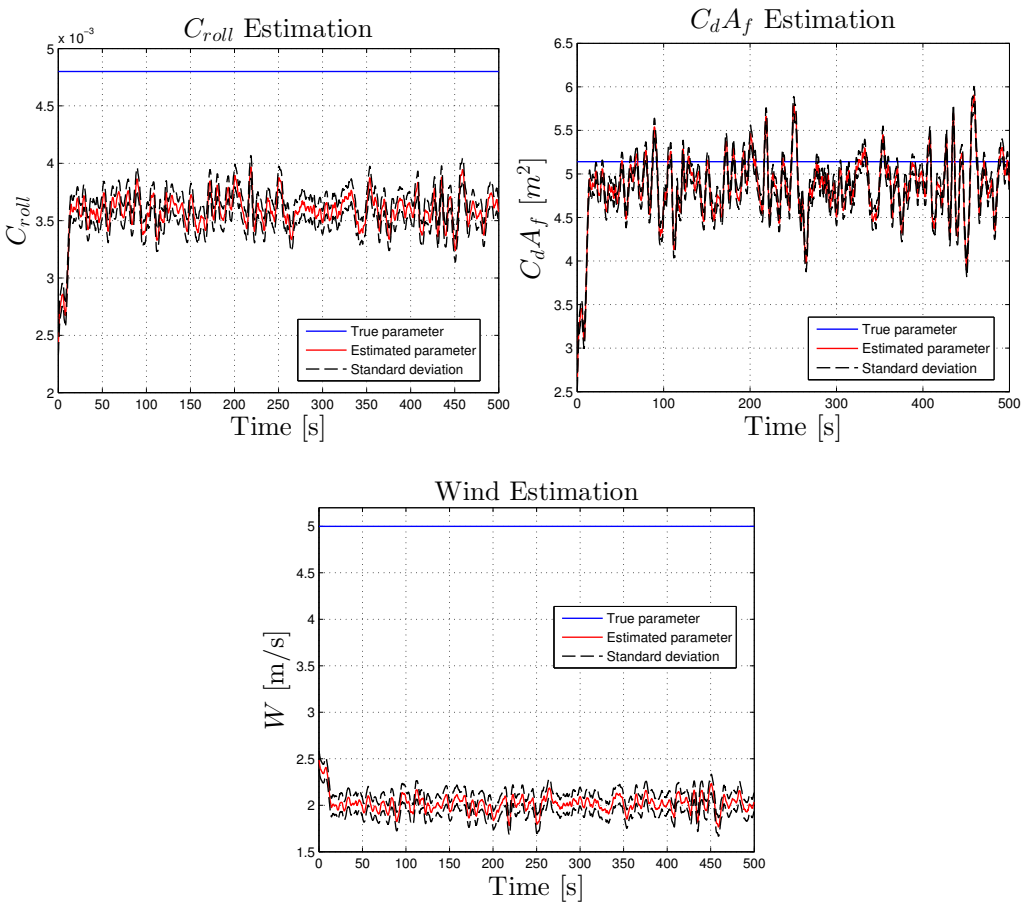

Figure 6.6.: Joint estimation of C_{roll}, C_dA_f and W using rich data.

6.1.2 Estimation using poor data

To explore the affect of estimating using poor measurement data, experiments have been performed on the data set shown in Figure 6.7. As seen in the figures the torque, the slope and the speed are constant.

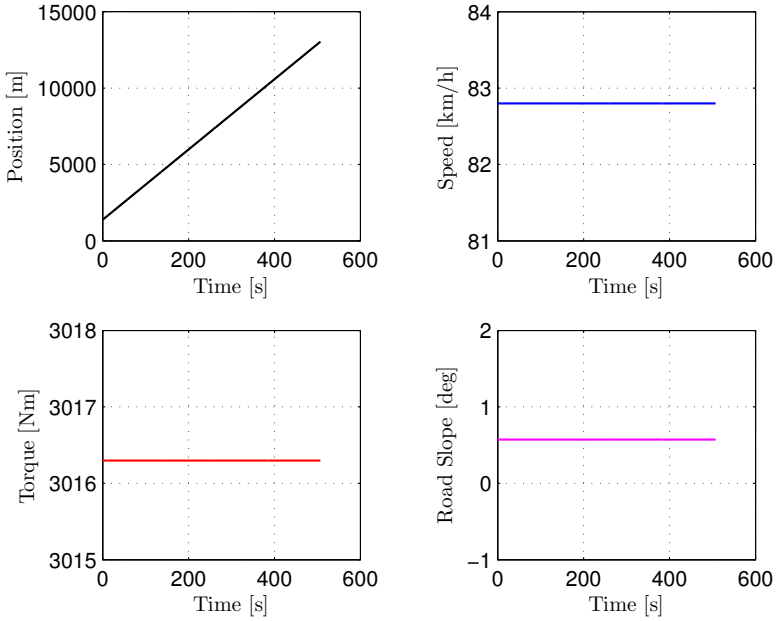

Figure 6.7.: Poor measurement data used for parameter estimation.

Estimating all the parameters separately gives the results shown in Figure 6.8, which shows that the MHE exponentially converge towards the true solution for all parameters. Note that the use of poor measurement data does clearly not impact the observability of these parameters.

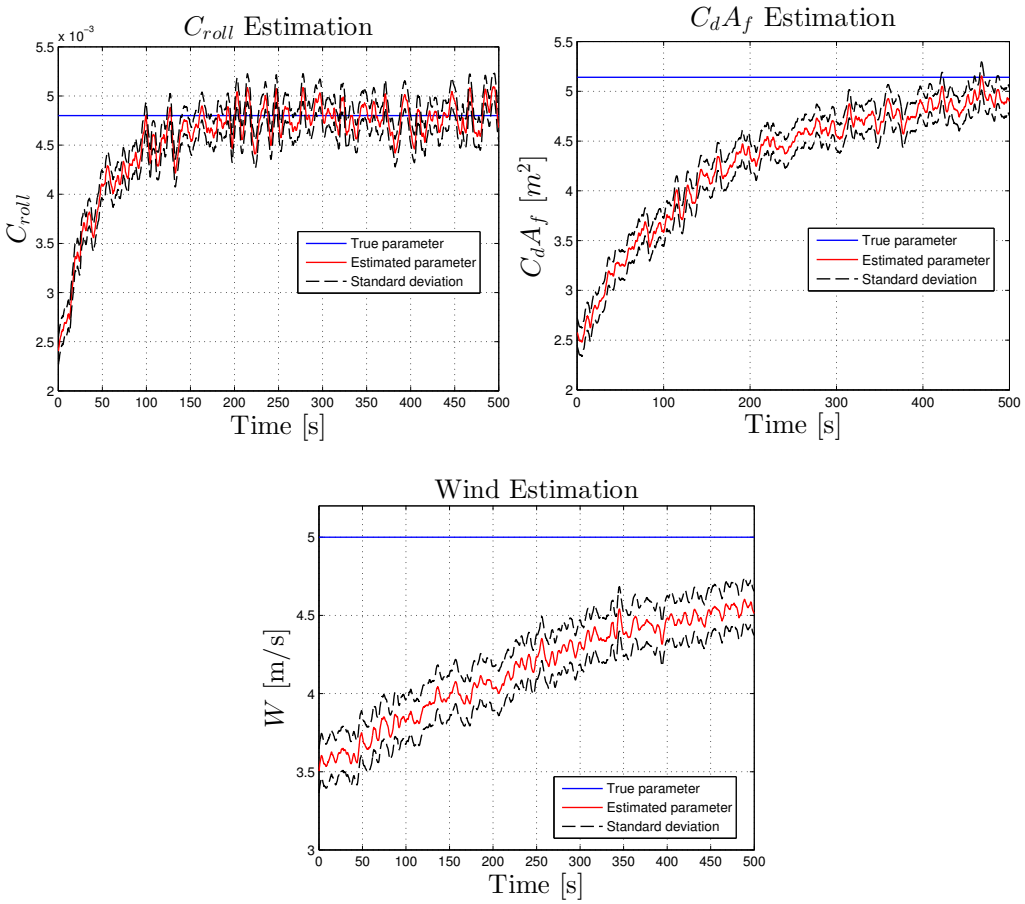

Figure 6.8.: Separate estimation of C_{roll}, C_dA_f and W using poor data.

The results of estimating the parameters C_{roll} and C_dA_f jointly are shown in Figure 6.9.

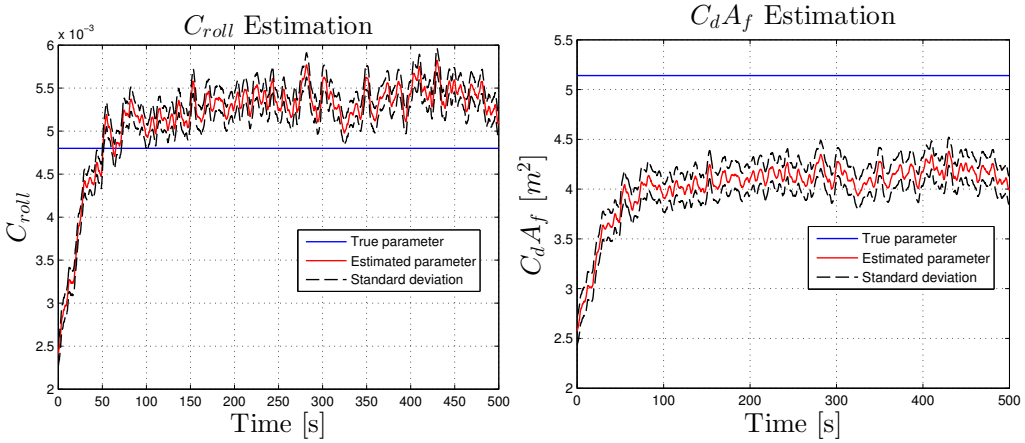

Figure 6.9.: Joint estimation of C_{roll} and C_dA_f using poor data.

The results of estimating C_{roll} and W jointly are shown in Figure 6.10. Further, the joint estimation of $C_d A_f$ and W is shown in Figure 6.11.

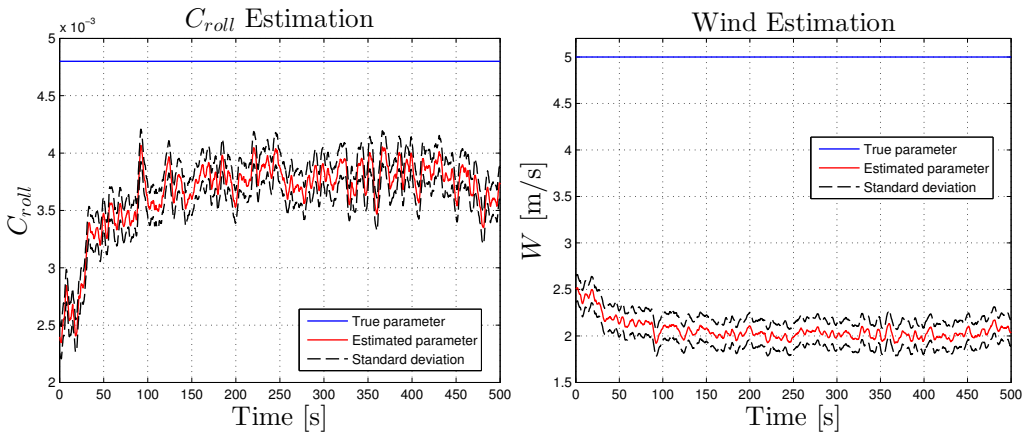

Figure 6.10.: Joint estimation of C_{roll} and W using poor data.

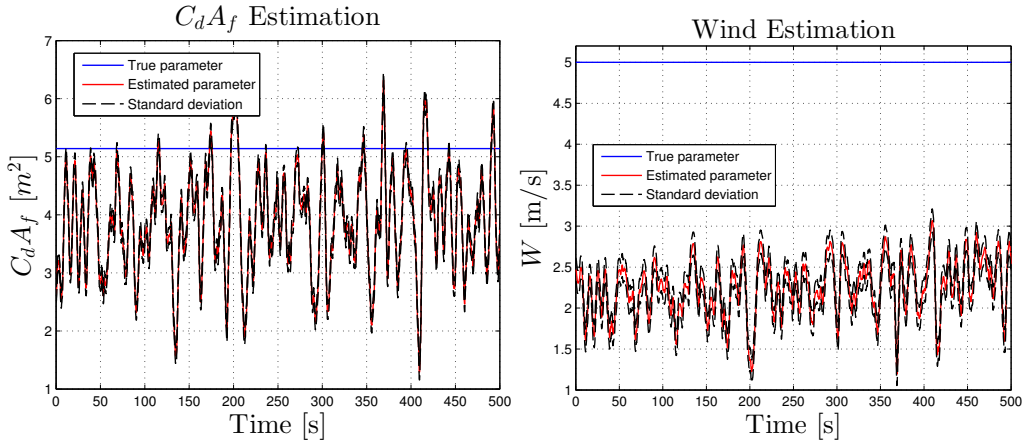

Figure 6.11.: Joint estimation of $C_d A_f$ and W using poor data.

Finally, estimating all parameters jointly gives the results shown in Figure 6.12.

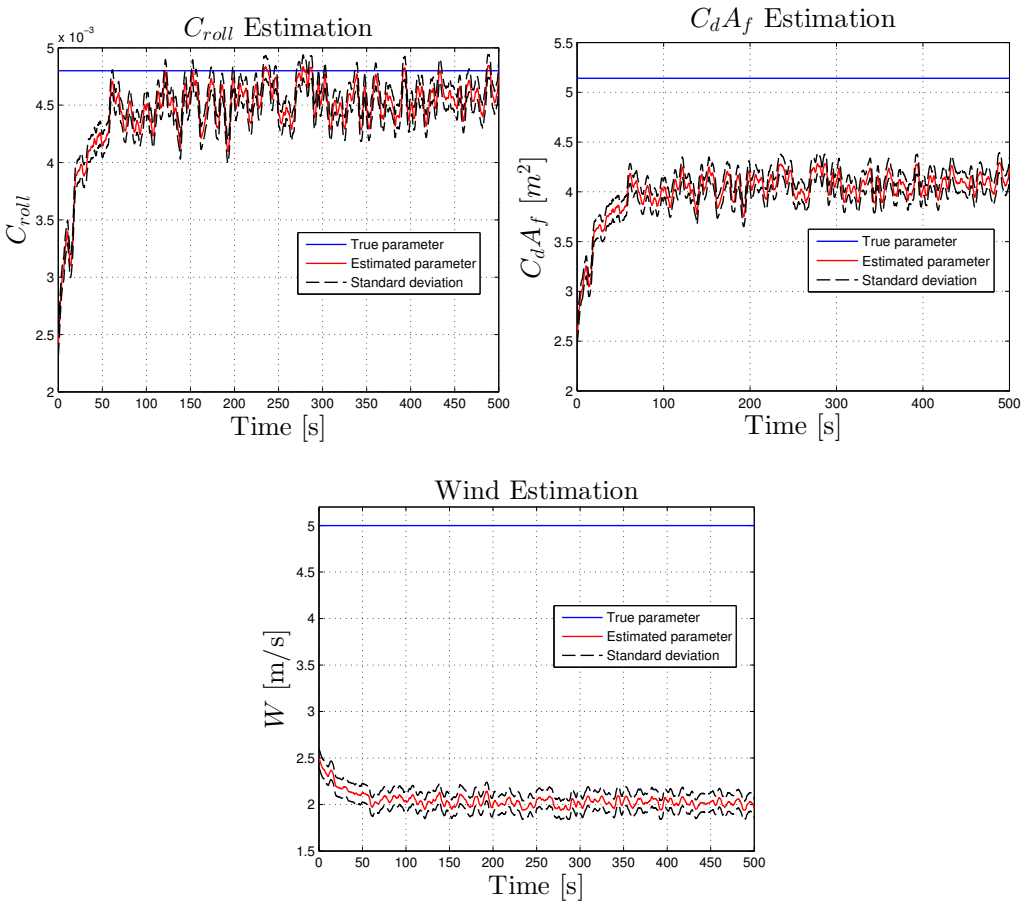

Figure 6.12.: Joint estimation of the parameters C_{roll}, C_dA_f and W using poor data.

6.2 Truck parameter identification using GSP data

To verify the MHE scheme, described in Section 3.4, estimations have been performed on GSP measurements from the drive cycle between Borås and Landvetter shown in Appendix B. The data is depicted in Figure 6.13 which is a short window of the complete sequence. In the simulations, the wind speed has been set to zero. This means that only the parameters C_{roll} and $C_d A_f$ are estimated.

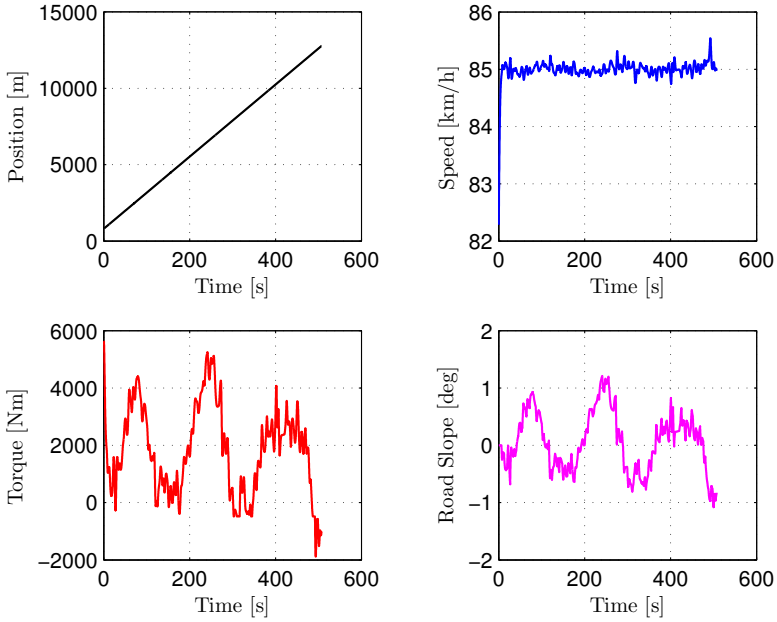

Figure 6.13.: Measurement data extracted from the GSP.

The parameters used to setup the MHE are shown in Table 6.4 and the results are shown in Figure 6.14.

Table 6.4.: Parameter used in the MHE.

Notation	Parameter	Value
Δt	Sample time	0.1
N	MHE Horizon	70
N_{iter}	SQP iterations	1
Q_A	Arrival cost	$0.5I$

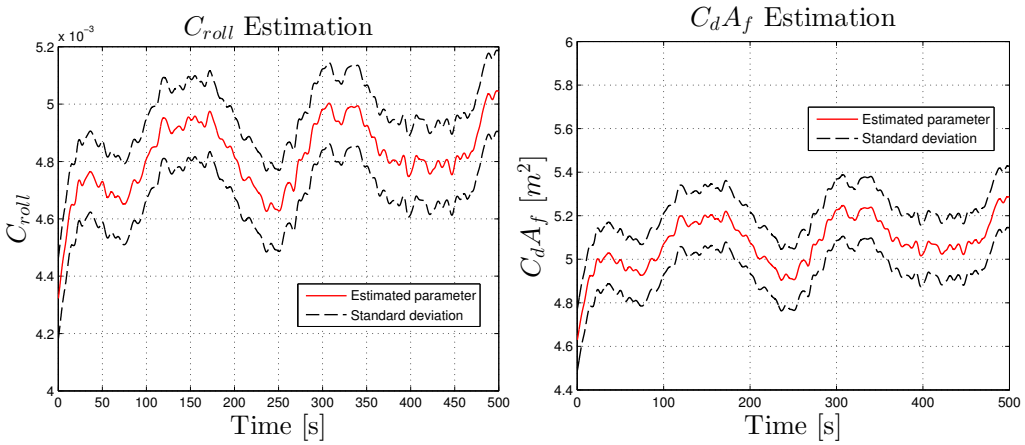

Figure 6.14.: Joint estimation of C_{roll} and C_dA_f using GSP data.

The MHE is verified separately and has not yet been implemented in the prediction algorithm. Hence, the average values after convergence have been computed which are used in the predictions presented later in this chapter.

- C_{roll}^{AVG} = 0.0048
- $C_dA_f^{AVG}$ = 5.0842

6.3 Open loop parameter identification

The open loop identification technique, described in Section 6.3, has been tested using GSP measurement from the drive cycle between Borås and Landvetter shown in Appendix B. The technique is evaluated by testing it on four different windows of data, with different characteristics, which are extracted from the measurements. The parameters obtained in the experiments are summarized in Table 6.5, where each window of data has a corresponding figure.

Table 6.5.: Shows the parameters that were obtained for each figure.

Param. / Fig.	K_p	K_i	I_0
6.15	$7.6 \cdot 10^5$	$2.6 \cdot 10^5$	0.3
6.16	$6.6 \cdot 10^5$	$2.2 \cdot 10^5$	0.3
6.17	$3.5 \cdot 10^4$	$3.6 \cdot 10^4$	2.5
6.18	$2.3 \cdot 10^4$	-369.2	-227.0

The outcome using the first window of data is illustrated in Figure 6.15, where the upper plot shows the fitted and the measured power. The second and the third plot shows the speed error and the integrated speed error respectively. Note that the fitted power corresponds to the model output when using the parameters obtained from solving (4.11) and the speed error as input. As the upper plot depicts, the fitted power matches the measured power well when it is positive. This behavior is reasonable, since negative power is excluded in the objective. Also note that the measured power is never saturated and is mostly above zero.

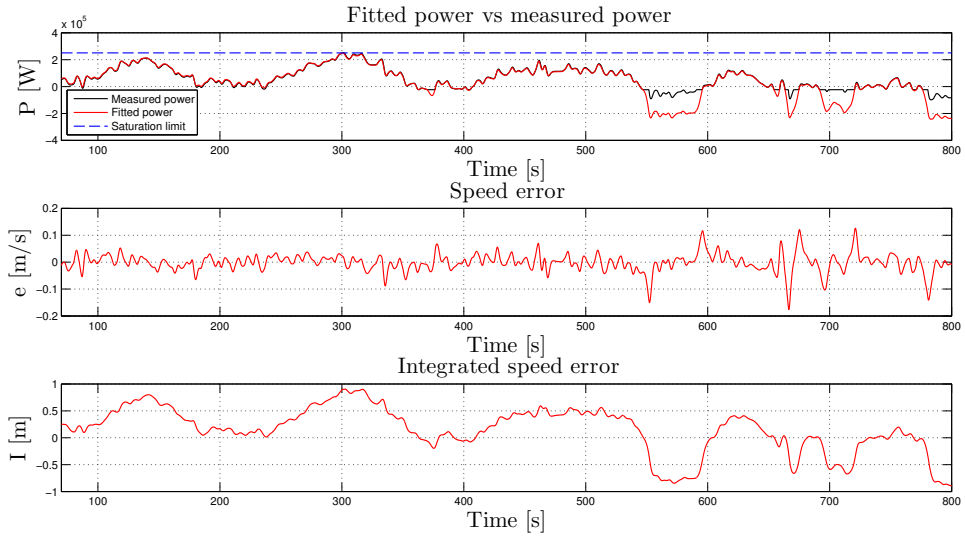

Figure 6.15.: Shows the identification results using the first window of data. The upper plot shows the power at the wheels. The second and the third plot shows the speed error and the integrated speed error respectively.

The second window of data and its results are shown in Figure 6.16. This window is similar to the first one but includes more negative data at the end of the sequence. Note that the fit is reasonably good during the first 850 seconds, but at the end the power becomes negative during a longer period of time. During this period of time, the integrator accumulates too much error which impairs the fit at the end. This indeed gives indication that the integrator should not continue to accumulate error while the power is negative, which reveals a weakness in the algorithm.

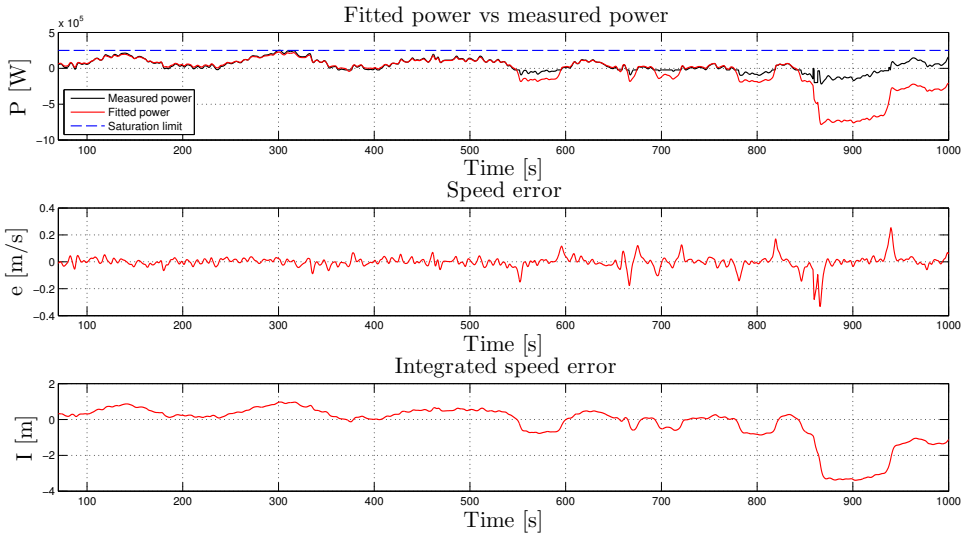

Figure 6.16.: Shows the identification results using the second window of data. The upper plot shows the power at the wheels. The second and the third plot shows the speed error and the integrated speed error respectively.

The third window includes saturation and is shown in Figure 6.17. Note that when the power is saturated, the gears are shifted which give arise to a drop in the power. Using the anti-windup technique purposed in (4.11), the integrator accumulates error during the power drop which is not endeavored. Since the error is large during the saturation, the integrated error increases significantly. The optimizer will then compensate by adjusting I_0 which will result in a decreased K_i. The fact that the error is large during saturation will also force the optimizer to decrease K_p.

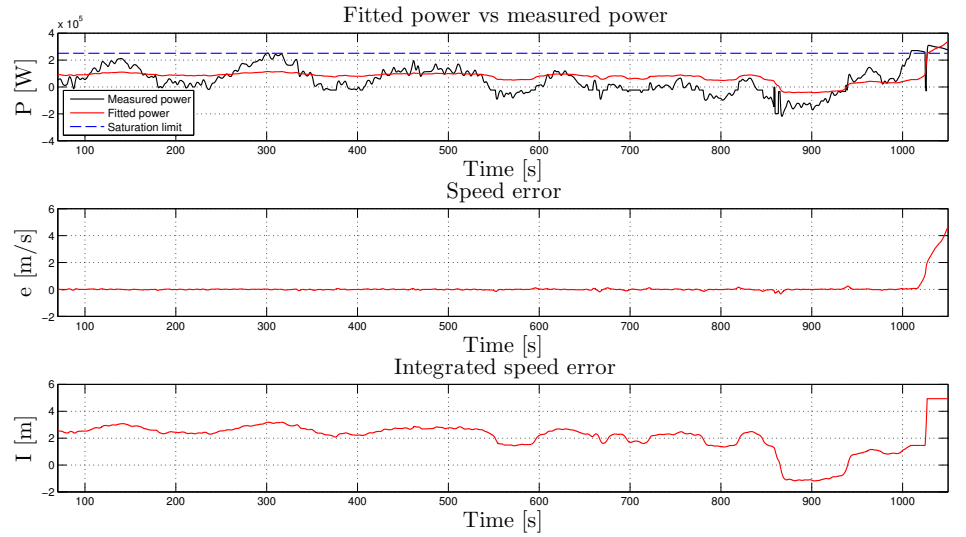

Figure 6.17.: Shows the identification results using the third window of data. The upper plot shows the power at the wheels. The second and the third plot shows the speed error and the integrated speed error respectively.

The fourth window is shown in Figure 6.18. The window includes two saturations in the power, which results in a high integrated error due to the gear shifts. The optimizer will then try to compensate by pushing down I_0 to a negative value. Since the integrated error is below zero for the entire sequence, K_i will be negative which is not physically meaningful.

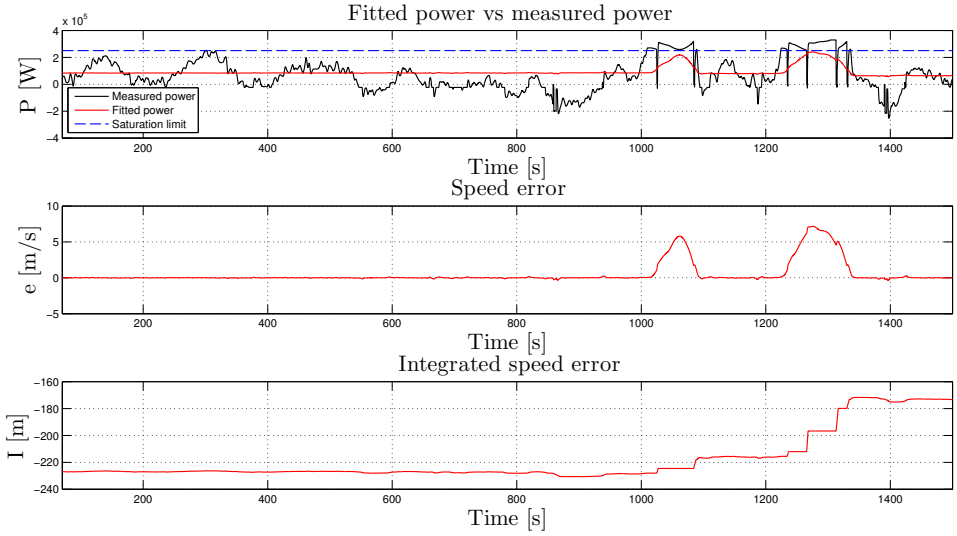

Figure 6.18.: Shows the identification results using the fourth window of data. The upper plot shows the power at the wheels. The second and the third plot shows the speed error and the integrated speed error respectively.

The following summarizes the key observations of the open loop identification technique:

- It works well when the power is positive and not saturated.
- It can not handle negative power for a long period of time since the truck changes dynamics when it is braking.
- It can not handle saturated power, since the integrator accumulates to much error during the gear shifts.

6.4 Closed loop parameter identification

Since the closed loop MHE described in Section 6.4 is not functioning properly, it has not been tested using GSP measurements. To illustrate its behavior, it is instead applied on constructed data with the same model as in the MHE. In order to make the problem less complex, only K_p and K_i are used as decision variables. An estimation using constructed data is shown in Figure 6.19. The MHE is initialized with an initial guess that is 90 percent of the true values of the parameters. Note that the MHE has an unstable behavior and the parameters do not converge towards their true values.

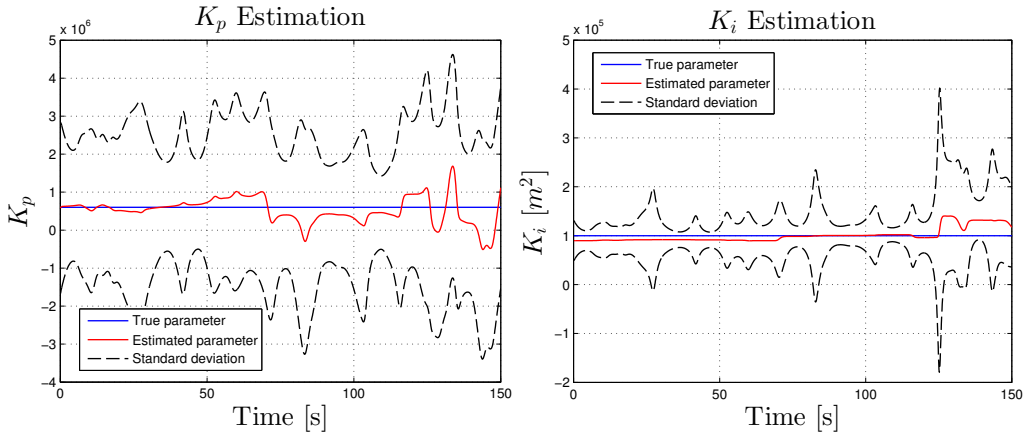

Figure 6.19.: Closed loop identification of K_p and K_i using constructed data.

6.5 Prediction

This section presents predictions performed in the GSP using the algorithm described in Section 5.2. The predictions have been perfomed with the GSP set on cruise controller mode. Recall that only the prediction algorithm using driver model #1 has been implemented in the GSP. The following figures will only show one prediction with 65 kilometer horizon.

Performing a prediction on the Borås-Landvetter drive cycle, using the parameters obtained in Figure 6.15, gives the results shown in Figure 6.20. Note that the figure includes the predicted power as well, to better illustrate the behaviors. The gear shifts and the correct saturation levels are not captured in the torque (and also not in the power). Consequently, this affects the speed prediction during the accelerations and the saturations.

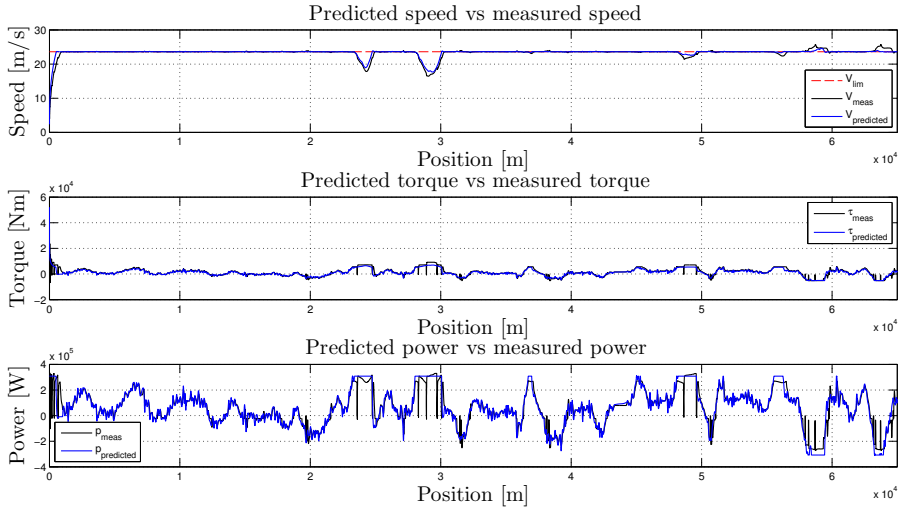

Figure 6.20.: Illustrates the prediction of the speed, the torque at the wheels and the power at the wheels using the driver model parameters: $K_p = 7.5832 \cdot 10^5$ and $K_i = 2.6019 \cdot 10^5$.

6.5.1 Sensitivity of the driver model parameters

This section will demonstrate how variations in the driver model parameters affect the predictions. In the first prediction illustrated in Figure 6.20, the parameters gave good predictions. Hence, these parameters will serve as a starting point to explore what happens when the parameters are perturbed from this point.

Figure 6.21 reveals the affect of decreasing K_p one order of magnitude. Note that the predictions start to oscillate which indicates that the algorithm is quite sensitive to decreases in K_p.

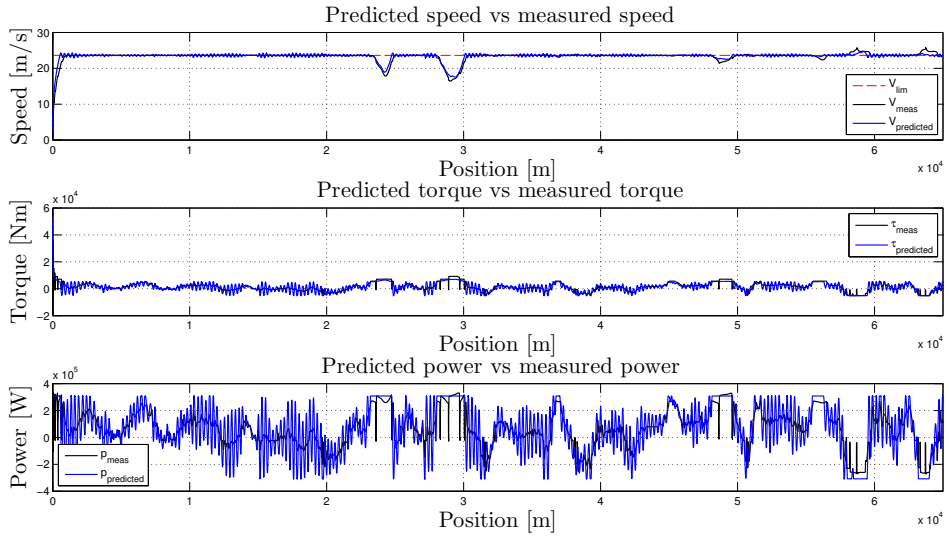

Figure 6.21.: Illustrates the prediction of the speed, the torque at the wheels and the power at the wheels using the driver model parameters: $K_p = 7.5832 \cdot 10^4$ and $K_i = 2.6019 \cdot 10^5$.

The next experiment, shown in Figure 6.22, illustrates what happens when K_p is increased one order of magnitude. Note that the speed is fluctuating less, but the predictions are still rather good.

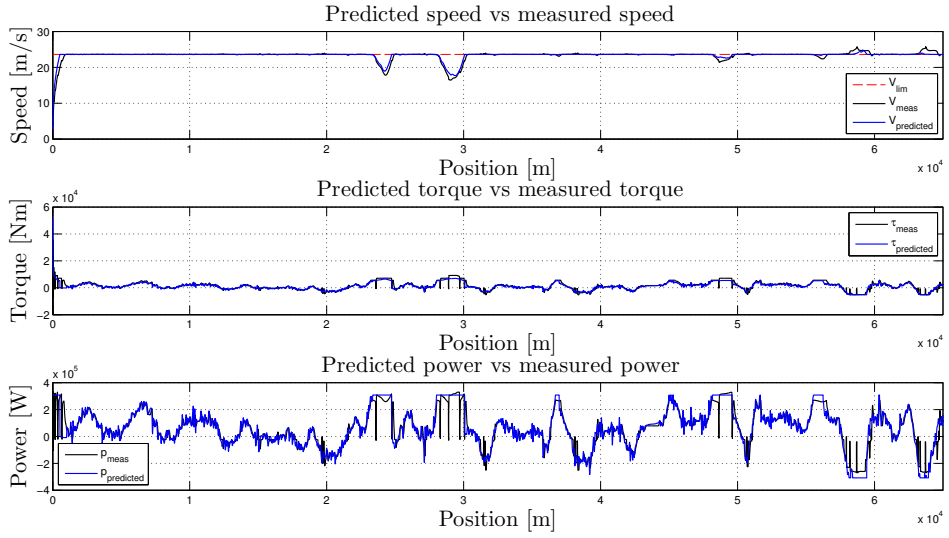

Figure 6.22.: Illustrates the prediction of the speed, the torque at the wheels and the power at the wheels using the driver model parameters: $K_p = 2.5832 \cdot 10^6$ and $K_i = 2.6019 \cdot 10^5$.

Next case illustrates what happens when K_i is decreased four orders of magnitude, which is shown in Figure 6.23. The figure reveals that reducing K_i massively provides a little more fluctuations, but overall it has not a significant impact. This is due to that K_i is coupled with the integration term and its size determines how fast the speed error is

pushed to zero.

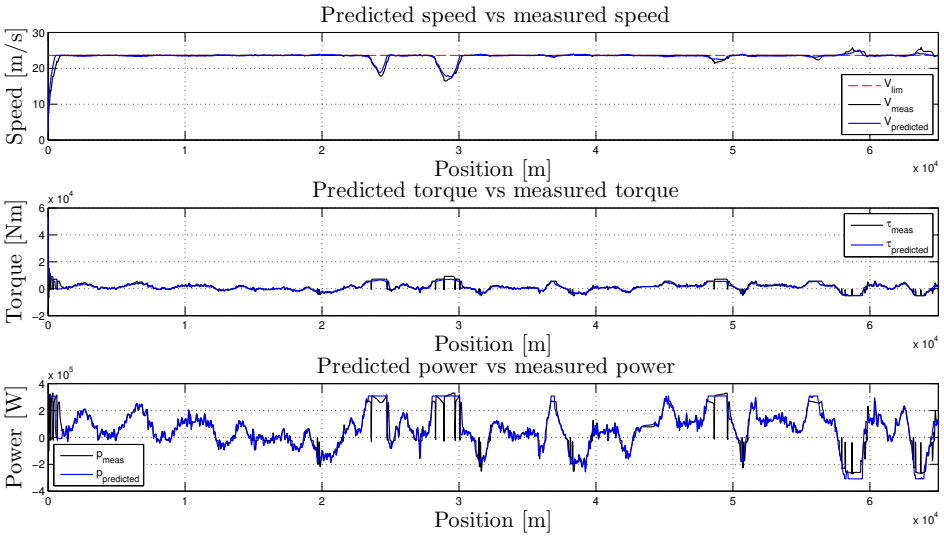

Figure 6.23.: Illustrates the prediction of the speed, the torque at the wheels and the power at the wheels using the driver model parameters: $K_p = 7.5832 \cdot 10^5$ and $K_i = 2.6019 \cdot 10^1$.

The last predictions are shown in Figure 6.24 and illustrates the impact of increasing K_i one order of magnitude. The results show that the speed is less fluctuating while the torque is fluctuating more. Nevertheless, the predictions are apparently rather good.

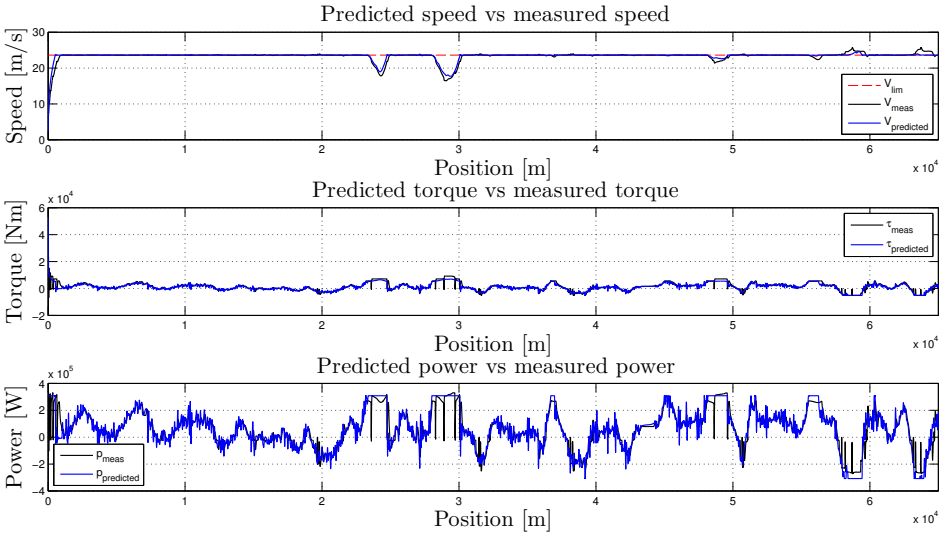

Figure 6.24.: Illustrates the prediction of the speed, the torque at the wheels and the power at the wheels using the driver model parameters: $K_p = 2.5832 \cdot 10^5$ and $K_i = 2.6019 \cdot 10^6$.

6.6 Execution times

In this section, the execution times for the model integrators in both the prediction algorithm and the MHE scheme will be presented. The execution times are measured with a machine equipped with i7 3.5GHz/64bits processor.

6.6.1 Model integration in the prediction algorithm

The execution times for the model integrator, described in Section 5.2.2, for both driver models have been measured with a prediction horizon of three kilometers and a step size of one meter. The obtained results are summarized in Table 6.6.

Table 6.6.: Measured execution times for the model integrator in the prediction algorithm.

Driver model #1	Driver model #2
63.4μs	62.5μs

6.6.2 Model integration in the MHE

Recall that the MHE in Section 3.4 uses a model integrator with sensitivities in the construction of the equality constraints, which is one of the main time consuming parts in the scheme. The execution time for generating the equality constraints, using a horizon of $N = 50$, has been measured to approximately $0.15ms$.

7

Discussion

7.1 Prediction

The prediction algorithm was verified in Chapter 6 using measurements from the GSP set on cruise controller mode. One could argue that it still might be feasible to use it on a real driver, since a real driver is likely to behave almost as a PI-controller when it comes to long haul drives. However, to acquire a more realistic driver behavior, the driver model ought to be further developed to utilize the preview information the driver actually sees on the road.

Figure 6.20 reveals that the prediction algorithm does not capture the gear shifts and the correct saturation levels. Consequently, the speed dips are entirely captured. This is simply due to that the powertrain dynamics are not considered in the driver model, which could advantageously be included to improve the prediction algorithm.

The model integrators, coded in C, were executed in the microsecond range with a simple forward Euler method. The execution times can be even further reduced by optimizing the code and/or using a better integration method e.g., the Runge-Kutta method. Hence in a real-time context, the model integration is not an issue.

7.2 Truck parameter identification

The behavior of the MHE scheme, developed for identifying the parameters in the truck model, was analyzed using constructed measurement data in Section 6.1. The results revealed that it is not possible to recover the true parameter values when estimating multiple parameters jointly. This is not an artifact of the MHE scheme in itself, but is actually due to the existence of multiple solutions in the estimation problem. Even though the parameters are not correctly estimated, they will still yield an accurate truck model if they are used together. The results also showed that poorness in the measurement data do not have an impact on the observability of the parameters.

The tuning of the MHE scheme has a straight forward procedure as presented in 3.4.3. The only tricky part is the tuning of the arrival cost, which should reflect how much the previously estimated parameters are trusted. The arrival cost has a clear connection to the convergence rate of the MHE. For example, if the arrival cost is high then the parameters will not be allowed to change much from the previous MHE iteration and vice versa. Another natural consequence of having a high arrival cost is that it reduces the covariance of the MHE.

In order for the MHE scheme to be implemented in real-time, it needs to be efficiently written in C. The MHE scheme consists of mainly two time consuming procedures, namely the model integration to obtain the equality constraints in (3.18) and the matrix factorization to solve the KKT system (3.11). The execution time for the model integration using a horizon of $N = 50$ was measured to approximately $0.15ms$. According to [12], a standard QP solver can have execution times in the millisecond range. Therefore, one could argue that solving the KKT system can be done even faster, since aiming at solving a QP without inequality constraints. Hence, the MHE scheme is most likely to have execution times in the millisecond range.

7.3 Open loop parameter identification

The open loop identification method was shown to not work effectively regardless of the data characteristics. The method functioned well when the measured power was positive and not saturated, which is illustrated in Figure 6.15. However, the method was unsuccessful when the power was negative or included saturations.

The reason to the problems with negative power is the fact that the truck is changing dynamics when it is braking, which apparently cannot be captured by a simple PI-controller. One possible way to handle this could be to use two models, one model for acceleration and one model for braking.

It was assumed in Chapter 4 that the power at the wheels has a constant saturation limit. As seen in Figure 6.18 the saturation is not constant and also includes outliers due to the gear shifts, which violates the assumption. Consequently, problems arise to identify the parameters since these behaviors are not captured by the driver model.

The bottleneck with this method is to fit a PI-controller on a very complex dynamics including both the driver and the powertrain, which was shown to be intrinsically hard. From the performed experiments, it can be concluded that the open loop identification method is not favorable to use in this context.

7.4 Closed loop parameter identification

As mentioned previously in Section 6.4, the closed loop identification approach is inherently hard due to the nonlinearity and non-smoothness of the closed loop dynamics. The main difficulty in the closed loop MHE, was the fact that the integrated error was not available for measurement. Without the measurement of the integrated error, the driver model parameters are unobservable. The integrated error was therefore constructed using the current measurements and the arrival parameters from the previous MHE run. However, this approach was unsuccessful and resulted in instability of the MHE.

A possible approach that is worth investigating in future work is to replace the integrated error in the objective function with the power at the wheels, which is indeed available for measurement. However, this method would no longer be a pure closed loop identification method but a mixture between open loop and closed loop. Consequently, it would also give arise to similar issues, as experienced in the open loop identification, that the power includes saturations and gear shifts. These issues could possibly be easier to handle in the context of MHE.

8

Conclusion

The prediction algorithm, without the identification routines, has been successfully implemented in the GSP. The algorithm is able to predict the vehicle speed and the torque at the wheels when the GSP is set on cruise controller mode. The model integrator, in the algorithm, runs in the microsecond range.

The prediction algorithm can be improved by further developing the driver model. One possible improvement is to include the powertrain dynamics and gear shift strategies in the driver model.

The MHE scheme for identifying the truck model parameters is able to estimate the aerodynamic drag coefficient, the rolling friction coefficient and the average wind speed. The MHE can be efficiently coded in C to most likely execute in the millisecond range.

The open loop parameter identification method for identifying the driver model parameters proved to be very difficult, since the dynamics of the driver and the powertrain had a much higher degree of complexity than the driver model itself. Therefore, this method is not favorable to use in this context.

The closed loop parameter identification for identifying the driver model parameters has not yet been fully investigated. The main difficulty is the fact that the integrated error state is not available for measurement, which is needed for making the parameters observable. The approach for constructing the integrated error from the current measurements and the arrival parameters was unsuccessful and resulted in instability of the MHE.

9

Future work

The main area of improvement in this thesis is the driver parameter identifcation. In order for the prediction algorithm to be useful in a real application, a robust method to acquire the driver model parameters is needed. The closed loop MHE is one approach that could be further developed and explored in a future work, by e.g. utilizing the power measurements in the objective as discussed in 7.4.

When having a functioning identification method for the driver model parameters, the driver model could be further developed to improve the predictions. The following suggests possible improvements of the driver model that could be accomplished in a future work:

- Include the powertrain dynamics and gear shift strategies.
- Extend the model by utilizing the preview information the driver actually sees on the road, such as upcoming speed limits and road slopes.

Finally, the first order uncertainty propagation, described in Section 5.4.2, could be implemented and verified to obtain a computationally inexpensive method for estimating the covariance of the predictions.

Bibliography

[1] E. Hellström, "Look-ahead control of heavy vehicles," Ph.D. dissertation, 2010.

[2] E. Union. (2014) Complete vehicle energy-saving technologies. [Online]. Available: http://www.convenient-project.eu/fe\#

[3] M. C. Grant and S. P. Boyd. (2014) Cvx User's Guide. [Online]. Available: http://web.cvxr.com/cvx/doc/intro.html\#what-is-cvx

[4] E. Hairer, S. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations I - Nonstiff Problems*. Berlin: Springer, 1993.

[5] J. B. Rawlings and D. Q. Mayne, *Model Predictive Control - Theory and Design*. Madison, Wisconsin: Nob Hill Publishing, 2009.

[6] E. L. Haseltime and J. B. Rawlings, "A Critical Evaluation of Extended Kalman Filtering and Moving Horizon Estimate," TWMCC Texas Wisconsin modeling and control consortium, Tech. Rep., Mar. 2003.

[7] J. Nocedal and S. J. Wright, *Numerical Optimization*. New York: Springer, 2006.

[8] M. Diehl, H. Bock, J. Schlöder, R. Findeisen, Z. Nagy, and F. Allgöwer, "Real-time Optimization and Nonlinear Model Predictive Control of Processes Governed by Differential-algebraic Equations." *J. Proc. Contr.*, vol. 12, no. 4, pp. 577 – 585, 2002.

[9] D. Pole, *Linear Algebra - A Modern Introduction*. Boston: Brooks/Cole, Cengage Learning, 2011.

[10] P. Getreuer. (2010) Writing Matlab C/Mex Code. [Online]. Available: http://classes.soe.ucsc.edu/ee264/Fall11/cmex.pdf

[11] MathWorks. (2014) Integrate C Functions Using Legacy Code Tool. [Online]. Available: http://www.mathworks.se/help/simulink/sfg/integrating-existing-c-functions-into-simulink-models-with-the-legacy-code-tool.html

[12] A. Domahidi, A. Zgraggen, M. Zeilinger, M. Morari, and C. Jones, "Efficient interior point methods for multistage problems arising in receding horizon control." *In IEEE Conference on Decision and Control (CDC)*, pp. 668 – 674, December 2012.

A

Road map data

The code below depicts the structure of the road map data which is included in the GSP and is available for every drive cycle. In the thesis, the second column corresponds to the position, denoted $\bar{S}$. The third column corresponds to the speed limits, denoted $\bar{V}$. The fourth column corresponds to the altitude, denoted $\bar{A}$.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%      row            pos           vlim          alt
%    ________________________________________________________
%      [1]            [m]           [m/s]         [m]
%
Road_Matrix = [
       1.0000 ,        0.0000 ,     23.6111 ,     128.2000
       2.0000 ,     1000.0000 ,     23.6111 ,     128.2100
       3.0000 ,     1028.0500 ,     23.6111 ,     127.9200
       4.0000 ,     1084.1500 ,     23.6111 ,     127.8700
       5.0000 ,     1140.2400 ,     23.6111 ,     127.7300
       6.0000 ,     1196.3400 ,     23.6111 ,     127.2000
       7.0000 ,     1252.4400 ,     23.6111 ,     126.7600
       8.0000 ,     1306.54 ,       23.6111 ,     126.3800
       9.0000 ,     1364.6300 ,     23.6111 ,     126.5700
      10.0000 ,     1420.7300 ,     23.6111 ,     126.1900
      11.0000 ,     1476.8300 ,     23.6111 ,     125.3200
      12.0000 ,     1532.9300 ,     23.6111 ,     125.5900
      13.0000 ,     1589.0200 ,     23.6111 ,     125.4400
      14.0000 ,     1645.1200 ,     23.6111 ,     125.2100
      15.0000 ,     1701.2200 ,     23.6111 ,     125.2000
      16.0000 ,     1757.3200 ,     23.6111 ,     124.9600
      17.0000 ,     1813.4100 ,     23.6111 ,     124.9100
      18.0000 ,     1869.5100 ,     23.6111 ,     125.0700
```

```
    19.0000  ,   1925.6100  ,     23.6111  ,     125.2700
    20.0000  ,   1981.7100  ,     23.6111  ,     125.7500
    21.0000  ,   2037.8000  ,     23.6111  ,     126.1400
    22.0000  ,   2093.9000  ,     23.6111  ,     126.2900
    23.0000  ,   2150.0000  ,     23.6111  ,     126.6300
    24.0000  ,   2206.1000  ,     23.6111  ,     127.5500
    25.0000  ,   2262.1900  ,     23.6111  ,     128.1700
    26.0000  ,   2318.2900  ,     23.6111  ,     128.7600
    27.0000  ,   2374.3900  ,     23.6111  ,     129.1900
    28.0000  ,   2430.4900  ,     23.6111  ,     130.1300
    29.0000  ,   2486.5800  ,     23.6111  ,     130.8200
    30.0000  ,   2542.6800  ,     23.6111  ,     131.4100
    31.0000  ,   2598.7800  ,     23.6111  ,     132.3700 ];
```

B

GSP measurements

From the GSP set on cruise controller mode, data is extracted for the drive cycle Borås-Landvetter. In figure B.1 the speed is shown, in B.2 the torque at the wheels is shown, in B.3 the power at the wheels is shown, in B.4 the road slope is shown.

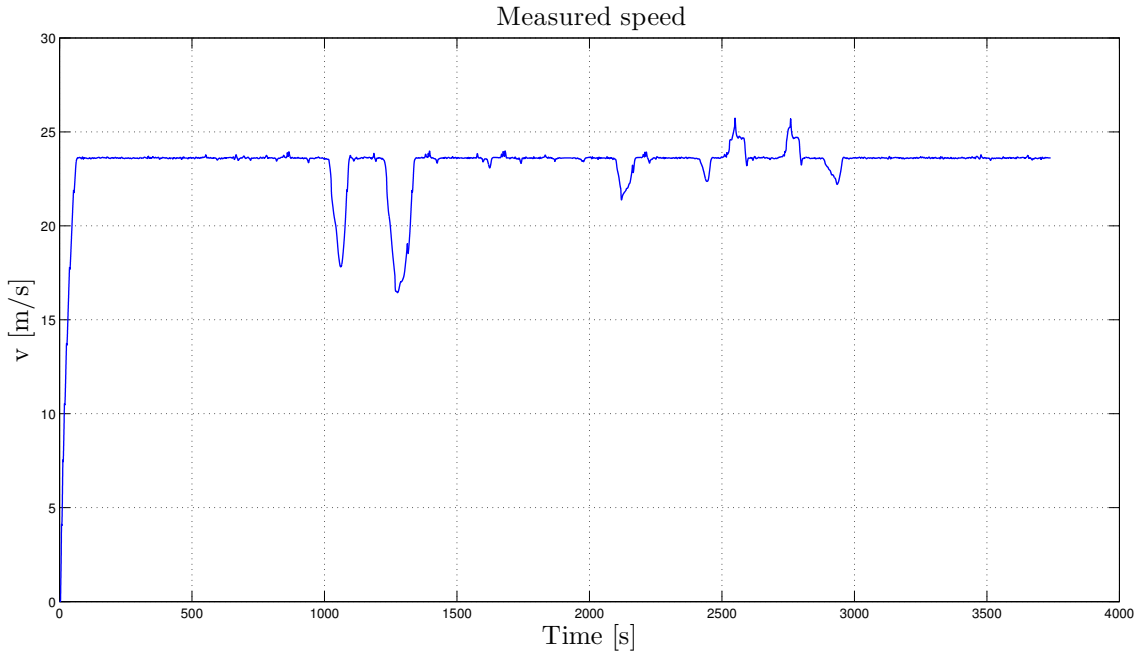

Figure B.1.: Illustrates the measured speed.

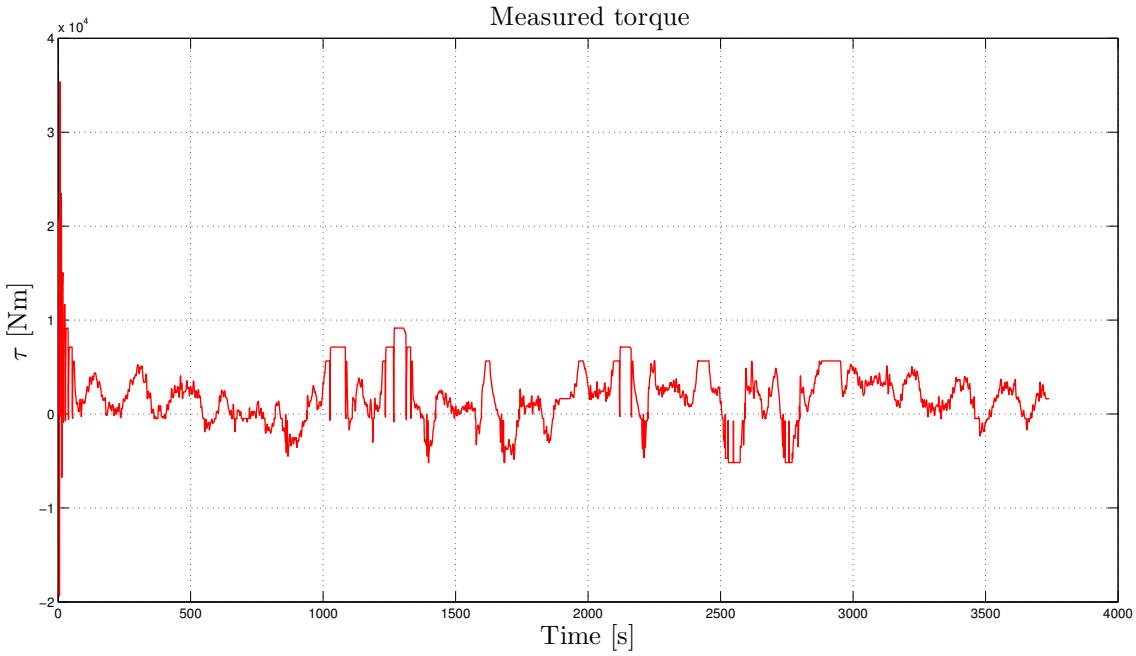

Figure B.2.: Illustrates the measured torque at the wheels.

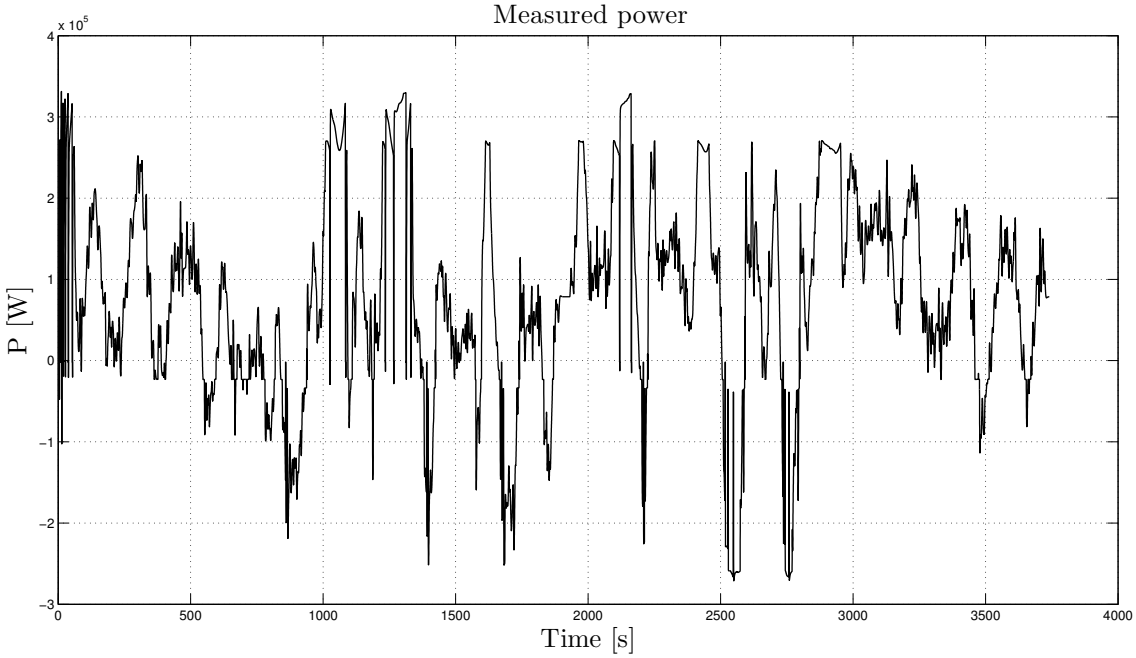

Figure B.3.: Illustrates the measured power at the wheels.

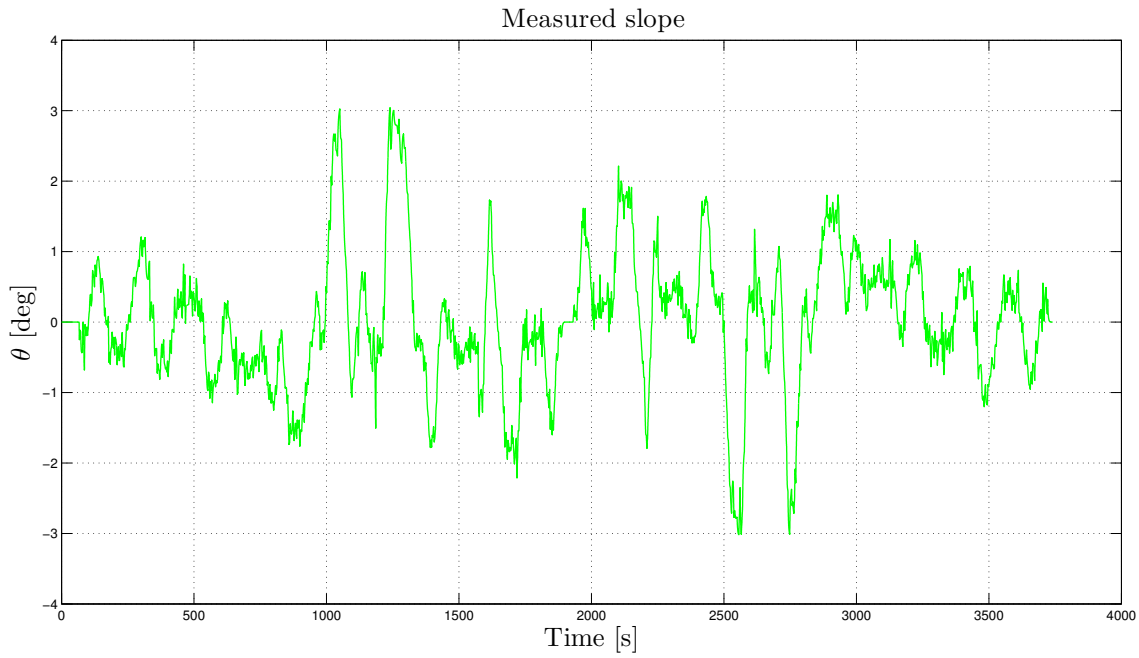

Figure B.4.: Illustrates the measured road slope.