UiO : Department of Physics
University of Oslo

Modeling and parameter estimation of an unmanned ground vehicle with a continuously variable transmission

Auby, Per Magnus

Master's Thesis, Spring 2016

Modeling and parameter estimation of an unmanned ground vehicle with a continuously variable transmission

Auby, Per Magnus

2016-05-23

Abstract

This thesis is about modeling of the longitudinal dynamics of an unmanned ground vehicle. This includes development of a longitudinal dynamics model for the vehicle under consideration as well as estimation of unknown parameters in the model. The parameters are estimated from measurements provided by the vehicle's instrumentation. The proposed use of the model is to improve the vehicle's control system.

The unmanned ground vehicle is based a Polaris Ranger utility vehicle, which has a continuously variable transmission (CVT). The CVT consist of a rubber belt and two variable radius pulleys, which allows the speed ratio of the transmission to be continuously varied. A mechanical control system regulates the speed ratio of the transmission based on the engine speed and torque load. A good model of the CVT and its mechanical control system is assumed to be vital for an accurate model of the vehicle.

A mathematical model describing the engine, CVT, drive line, tires, brakes, body, and terrain dynamics is proposed. The proposed CVT model is a new model, but its accuracy is not yet well proven. Simulations of the model show the expected behavior.

The proposed parameter estimation is based on maximum likelihood estimation. A state estimator based on the unscented Kalman filter (UKF) is used to calculate the likelihood of the observed measurements. The model parameters are optimized using the likelihood calculated by the UKF as criterion. To decrease the computation time, the parameter estimation uses a two stage approach where parameters unrelated to the drive line dynamics are estimated first using a separate set of observations and a lower order filter model.

Estimation of parameters from simulated measurements gave reasonable results, but with non-negligible bias for some of the parameters. Simulations of the model with estimated parameters did not give results good enough to be useful for control system development directly. The simulations followed the general shape of the response of the experimental observations, but generally with a different magnitude. The model with estimated parameters is accurate enough for use as a filter model for state estimation.

As the parameter estimation gave slightly biased results even for simulated data, it is hard to quantify how much of the inaccuracy is caused by the bias in the parameter estimation algorithm and how much is caused by errors in the model. It is nevertheless believed that at least some of the inaccuracy is caused by errors in the model. Suggestions for improvements to the both the model and the parameter estimation algorithm are given, but due to lack of time they could not be implemented.

It is concluded that the proposed model and parameter estimation approach have good potential to give satisfactory results, but that it will require additional work.

Preface

This thesis is was written as the final part of my two year master's degree in electronics and information technology with specialization in cybernetics at the University of Oslo[1]. It contains work done between 2016-01-18 and 2016-05-23, at the University Graduate Center Kjeller (UNIK). The thesis is solely written by the author, though parts of the thesis is based on previous research by other authors. References to the sources of previous research are given where applicable. All illustrations are the authors original work. The authors background is a bachelors degree in electronics and information technology with specialization in automation from Oslo and Akershus University College of Applied Sciences[2].

I would like to thank my supervisors from the Norwegian Defence Research Establishment (FFI): researcher Kim Mathiassen, researcher Magnus Baksaas, and researcher Marius Thoresen, as well as the rest of FFIs unmanned ground vehicle (UGV) project, for providing me the with the subject of this thesis: modelling and parameter estimation of the longitudinal dynamics of FFIs experimental UGV, OLAV. I would also like to thank my supervisor at UNIK, professor Oddvar Hallingstad.

Magnus Baksaas has been of special help in gathering measurements from OLAV's instrumentation. Due to the experimental nature of OLAV this is a task requiring two persons: one person to steer the vehicle using the steering wheel and one person to electronically control the throttle and brake actuators using a gamepad; which was an interesting experience...

Finally a special thanks to my girlfriend and family for supporting me, in particular with warm dinner when coming home after a long day of working on my thesis.

Kjeller, 2016-05-23

Auby, Per Magnus

[1] Norwegian: Master i elektronikk og datateknologi med studierettning kybernetikk ved Univeristet i Oslo
[2] Norwegian: Høgskolen i Oslo og Akershus (HiOA)

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

1.1 Motivation

The Norwegian Defence Research Establishment (FFI) have for many years conducted research on unmanned vehicles. The most prominent of these research efforts has been on autonomous underwater vehicle (AUV) by being part of the development and operation of the HUGIN AUV. During the later years there has been a push to make unmanned vehicles truly autonomous. FFI has therefore started a number of projects focusing on autonomous vehicles, both aerial, surface, underwater and ground vehicles. On of these projects is the development of the experimental unmanned ground vehicle (UGV) platform OLAV, short for off-road light autonomous vehicle (OLAV).

Off-road light autonomous vehicle (OLAV) is based on a Polaris Ranger terrain utility vehicle outfitted with necessary actuators and instrumentation for autonomous operation. The actuators control the steering, throttle, brakes and gear selection. As of writing this the instrumentation includes a state of the art inertial and global navigation satellite system (GNSS) aided navigation system similar to the system found on HUGIN, measurements of the engine speed and the speed of the rear differential. The instrumentation will also include a computer vision system, but it is not yet implemented. The vehicles control system is implemented on a conventional x86 computer running Ubuntu Linux with the Robotic Operating System (ROS) framework.

Currently only a simple empirically tuned proportional-integral-controller for longitudinal velocity regulation is implemented. To develop a more sophisticated control system the dynamics of the vehicle has to be investigated. This entails developing a mathematical model of the vehicle and estimating its unknown parameters. The model can then either be used to develop a new control system or used directly for some variations of optimal or predictive model control. Such a model could also prove useful for more than just the low level control of the vehicle. It could also be used by the path planner for validating and choosing the best path, used to improve the navigation system, and for fault detection of the vehicle.

The Polaris Ranger has a rubber belt continuously variable transmission (CVT). A CVT is a transmission which can vary its speed ratio continuously between a lower and upper bound without disconnecting the engine from the transmission. This is unlike a transmission found in a typical car which has a set of discrete gears providing a discrete set of speed ratios and has to interrupt engine power by a clutch to change gears. A rubber belt CVT accomplishes this by using a rubber V-belt and two variable radius V-belt pulleys. As implemented in the Polaris Ranger the belt and pulleys also provides the initial engagement of the engine. The CVT in the Polaris Ranger uses a mechanically control system to regulate the CVTs speed ratio based on the engine speed and the applied torque load. The CVT is assumed to significantly influence the vehicles behaviour and is therefore an important part of a model of the vehicle.

1.2 Purpose and goals

The goal of this master thesis is to develop a mathematical model describing the longitudinal dynamics of OLAV. It should take into account engine and brake system dynamics, tire and ground interaction, the inclination of the vehicle and terrain, and other factors found to significantly impact the vehicles behaviour. This will require some form of parameter estimation as there are likely to be a number of unknown parameters.

The general goals of the thesis can be summed up by the following points:

A. Investigate and develop a model of the longitudinal dynamics of the vehicle.

B. Identify and estimate unknown parameters in the model.

C. Develop a state estimator based on the model.

D. Verify the model and the state estimators performance.

Some of these goals overlap, for example goal B and C, as the parameter estimation approach used, maximum likelihood, already requires a state estimator.

1.3 Contribution

Modeling of vehicle dynamics is hardly a new field of research, although most of the literature describes development of complex models. These complex models are more appropriate for design studies, than directly appropriate for state estimation filter models or models used in control systems. The least researched part of the vehicle is the transmission, which is a rubber belt CVT. Few low order CVT models appropriate for the scope of the model are described in the literature.

Literature describing parameter estimation applied to vehicle dynamics are usually concerned with a single part of the system, for example the friction characteristics of the during braking [22], steering dynamics with known speeds of wheels [19], the shifting dynamics of a rubber belt CVT using a specialized test rig [12] etc. Because specialized equipment necessary to isolate the different subsystems of the vehicle are not available, the proposed problem for this thesis requires that the engine, CVT, tires, and overall longitudinal dynamics of the vehicle are investigated simultaneously. The limited measurements of the state of the drive line, only the speed of the engine and the rear wheels, makes this challenging.

The main contribution of this thesis is the new model for the rubber belt CVT as well as testing the feasibility of estimating the parameters in a complete stochastic model of the longitudinal dynamics. The model includes the engine, CVT, tires, and body dynamics, as well as a stochastic model of the inclination of the terrain. The unknown parameters in the model are estimated by using a state estimator to calculate the likelihood of the observed measurements and optimizing the parameters based on the likelihood.

1.4 Outline

Chapter 2: Background Presentation of the theoretical background used: System identification, state estimation, optimization, general dynamic system modelling, and vehicle dynamics modelling.

Chapter 3: Unmanned vehicle platform OLAV Description of the unmanned ground vehicle, OLAV. The general specifications of the vehicle as well as instrumentation and control systems.

Chapter 4: Vehicle model Presentation of the mathematical model of the longitudinal dynamics of the vehicle, how it was developed, and simulations of the model.

Chapter 5: Parameter estimation algorithm Presentation of the state estimator, filter models, criterion and optimization strategy used to estimate the parameters in the model.

Chapter 6: Parameter estimation Description of the more practical aspects of the parameter estimation: determining which parameters are known a priori and which parameters has to be estimated, test of the parameter estimation on simulated measurements, and description off the experiments performed to gather the observations used for the parameter estimation.

Chapter 7: Estimation results and model performance Comparison of the observed vehicle behaviour and simulation of the model with the estimated parameters.

Chapter 8: Discussion Discussion on the performance of the model and parameter estimation, and recommendations for further work.

Chapter 9: Conclusion Summary of the goals achieved, the most significant findings, and the final conclusion.

Appendix A: MATLAB code Appendix with the MATLAB code implementing the vehicle model, filter models, unscented Kalman filter (UKF) and parameter estimation algorithm.

Appendix B: Derivation of the drive line equations Appendix containing the derivation of the equations for the drive line model from the bond graph model of the drive line presented in section 4.3.4.

Chapter 2

Background

This chapter presents some of the theoretical background used to develop the model, state estimator, and parameter estimation presented in this thesis. First a general overview of system identification and different parameter estimation techniques is given, then state estimation and Kalman filtering with emphasis on the UKF, solving of differential equations using Runge Kutta numerical integration, a very brief overview of optimization theory and the particle swarm optimization algorithm, bond graph modeling, and finally some vehicle dynamics modeling theory. The presented material is generally introductory and deeper understanding have to be be found in the provided references.

2.1 System identification

System identification is to create mathematical models of physical systems using observations and statistics. Generally, such models can be placed into three categories: *white-*, *black-*, and *grey-box-models*. The difference between these categories is how much the model I based on knowledge about the physical laws governing the system and how much it is based on observation of the system. *White-box* models are models that are completely defined by prior knowledge about the system, while *black-box* are completely based on observations of the systems and no model structure is assumed. *Grey-box* models are somewhere in between the two others; both based on prior knowledges to build a general model structure as well as observations to determine the unknown quantities in the model.

System identification can be summed up by finding the transformation, g, that transform some set of known inputs $\mathcal{U}$ ($\mathcal{U}$ may be an empty set) to a set of observation $\mathcal{Z}$:

$$g : \mathcal{U} \to \mathcal{Z} \tag{2.1}$$

If a gray-box model for the system is assumed, the general structure of the transformation g is assumed known, but dependent on a set of parameters, here represented by a vector of parameters $\underline{\gamma}$:

$$g(\underline{\gamma}) : \mathcal{U} \to \mathcal{Z} \tag{2.2}$$

This is usually referred to as parameter estimation.

This section is based on [14], but using a different notion.

2.1.1 Deterministic and stochastic state space models

One possible structure for the transformation g is a state space model. The internal state of the system, represented by the state vector $\underline{x}$, is modelled by a state transition function f. The observations of the system is modelled by a measurement function, h, which relates the internal system state to the observations. Both f and h can be dependant on the current state, input, time, the parameters, and possible some unknown disturbance $\underline{v}$ and $\underline{w}$. Depending on the particular system, state transition can either be modeled by a difference equation:

$$\underline{x}_{k+1} = f(k, \underline{x}_k, \underline{u}_k, \underline{v}_k, \underline{\gamma}) \tag{2.3}$$

or a differential equation:

$$\dot{\underline{x}}(t) = f(t, \underline{x}(t), \underline{u}(t), \underline{v}(t), \underline{\gamma}) \tag{2.4}$$

Although the measurement function can be continuous, most systems have measurements sampled in discrete time.

$$\underline{z}_k = h(k, \underline{x}_k, \underline{u}_k, \underline{w}_k, \underline{\gamma}) \tag{2.5}$$

If the state is modelled as a continuous system, $\underline{x}_k$ and $\underline{u}_k$ are the state and input sampled at the time t_k.

If the random disturbances $\underline{v}$ and $\underline{w}$ are zero, the system is said to be deterministic and a given set of inputs $\mathcal{U}$ will always give the same observations $\mathcal{Z}$. If the disturbances $\underline{v}$ and $\underline{w}$ are not zero, but random processes, the system model is stochastic and will not give the same observations given the same set of inputs.

2.1.2 Parameter estimation

Parameter estimation is generally carried out by defining a criterion function $\mathcal{J}$ which is a function of the set of observations, $\mathcal{Z}$, the set of known inputs, $\mathcal{U}$, and the parameters, $\underline{\gamma}$. $\mathcal{J}(\mathcal{Z},\mathcal{U},\underline{\gamma})$ should monotonically decrease as the observations predicted by the model gets closer to the observations and the true model parameters, $\hat{\underline{\gamma}}$, should be the global minimum of the criterion. Finding the true parameters can then be found by minimizing the criterion.

$$\hat{\underline{\gamma}} = \min_{\underline{\gamma}} \mathcal{J}(\mathcal{Z},\mathcal{U},\underline{\gamma}) \tag{2.6}$$

Minimum mean squared error

A common criterion function is the mean-squared-error. It is defined as:

$$J_{\text{MSE}} = \frac{1}{N} \sum_{k=1}^{N} \left(\bar{\underline{z}}_k(\mathcal{U}_k, \underline{\gamma}) - \underline{z}_k\right)^T \left(\bar{\underline{z}}_k(\mathcal{U}_k, \underline{\gamma}) - \underline{z}_k\right) \tag{2.7}$$

where $\underline{z}_k$ is a vector of the observation at time k, $\bar{\underline{z}}(\mathcal{U}_k, \underline{\gamma})$ is a vector with the observation predicted by the model at time k from the set of inputs up to time k and the parameter vector $\underline{\gamma}$, and N is the total number of observations. The predicted observation is calculated by simulating the state with

$$\underline{x}_{k+1} = f(k, \underline{x}_k, \underline{u}_k, \underline{\gamma}) \tag{2.8}$$

or

$$\dot{\underline{x}}(t) = f(t, \underline{x}(t), \underline{u}(t), \underline{v}(t), \underline{\gamma}) \tag{2.9}$$

and sampling the predicted observations using the measurement function

$$\underline{z}_k(\mathcal{U}_k, \underline{\gamma}) = h(k, \underline{x}_k, \underline{u}_k, \underline{\gamma}) \tag{2.10}$$

This mean square error criterion can only be used for systems where the process equation of the state space model is deterministic.

The mean squared error is a popular criterion as it is easy to implement, has low computational complexity and works well for deterministic systems. Mean square error can also be made to work reasonably well for systems with stochastic measurements by adding a weight matrix, W, and giving smaller weight to the observations with larger uncertainties.

$$J_{\text{WMSE}} = \frac{1}{N} \sum_{k=1}^{N} \left(\bar{\underline{z}}_k(\mathcal{U}_k, \underline{\gamma}) - \underline{z}_k\right)^T W \left(\bar{\underline{z}}_k(\mathcal{U}_k, \underline{\gamma}) - \underline{z}_k\right) \tag{2.11}$$

Maximum likelihood

For stochastic systems, another approach has to be used as the system state is unknown, and comparing the predicted and observed measurements directly is therefore meaningless. The criterion function therefore has to incorporate both the statistics of the observation model and the process model. The likelihood, $p(\mathcal{Z}_k : \mathcal{U}_k, \underline{\gamma})$, increases with how likely it is that the observations $\mathcal{Z}_k$ are of the model with parameters $\underline{\gamma}$ given inputs $\mathcal{U}_k$. The most likely candidate for the true parameters $\hat{\underline{\gamma}}$ is thereby the parameters that maximizes the likelihood function. The negative likelihood is therefore a possible candidate as criterion function for stochastic systems. In practice it can be easier to calculate the natural logarithm of the likelihood, abbreviated log likelihood or $\xi(\mathcal{Z}_k, \underline{\gamma})$, giving the maximum log likelihood criterion:

$$\mathcal{J}_{MLL} = -\xi(\mathcal{Z}_k, \underline{\gamma}) \tag{2.12}$$

Section 2.2.3 describes how the log likelihood can be calculated with an UKF.

2.2 The Unscented Kalman filter

The Kalman filter is an optimal state estimator for linear stochastic systems with additive zero mean Gaussian noise developed by Rudolf Emil Kálmán, which is extensively described in the literature, for example in [2]. These systems are generally written in the form of a process and measurement model:

$$\underline{x}_{k+1} = F_k \underline{x}_k + L_k \underline{u}_k + \underline{v}_k \tag{2.13}$$

$$\underline{z}_k = H_k \underline{x}_k + B_k \underline{u}_k + \underline{w}_k \tag{2.14}$$

where F_k, L_k, H_k, and B_k are are matrices and $\underline{v}$ and $\underline{w}$ are uncorrelated zero mean Gaussian noise processes with covariance Q_k and R_k. Assuming that the model of the system is correct, the Kalman filter gives an minimum mean square error estimate of the system state, with mean $\hat{\underline{x}}_k$ and covariance $\hat{P}_k$, from the measurements $\mathcal{Z}_k = \underline{z}_1, \ldots, \underline{z}_k$, and the a priori state estimate given by $\bar{\underline{x}}_0$ and $\bar{P}_0$.

The Kalman filter is a recursive estimation algorithm. Between each measurement the previous best estimate of the system state, $\hat{\underline{x}}_{k-1}$ and $\hat{P}_{k-1}$, is predicted to the next time step using the process model(eq. (2.13)). The predicted state, $\bar{\underline{x}}_k$ and $\bar{P}_k$, describes the most likely system state before any information from the measurements are incorporated. This is called the time update. The covariance of the prediction, $\bar{P}_k$, as well as the measurement model(eq. (2.14)) is then used to compute an optimum gain, called the Kalman gain, $\mathcal{K}_k$, which is used to combine the information from the prediction and the measurements to an optimum state estimate, $\hat{\underline{x}}_k$ and $\hat{P}_k$. This is called the measurement update. The time update and measurement update is repeated recursively from the initial state estimate, $\hat{\underline{x}}_0$ and $\hat{P}_0$, until all measurements, $\mathcal{Z}_N$ are used.

The Kalman filter can not be used for nonlinear systems or systems with non additive noise. Such systems can generally be written as:

$$\underline{x}_{k+1} = \underline{f}\left(k, \underline{x}_k, \underline{u}_k, \underline{v}_k\right) \tag{2.15}$$

$$\underline{z}_k = \underline{h}\left(k, \underline{x}_k, \underline{u}_k, \underline{w}_k\right) \tag{2.16}$$

where $\underline{f}$ and $\underline{h}$ are arbitrary functions of the state, input, time, and the disturbance processes. For these systems one of the many extensions of the Kalman filter for nonlinear systems must be used. The most popular of these extensions has been the extended Kalman filter (EKF) which uses a Taylor expansion to predict the covariance of the state estimate. The UKF is another extension of the Kalman filter which is regarded to have better performance than the EKF. The EKF is generally only implemented using a first order Taylor expansion, while the UKF approximates the mean and covariance of the state with the same accuracy as as a third order Taylor expansion with similar computational complexity to a first order EKF[11, 25]. The UKF was originally proposed by Julier, Uhlmann, and Durrant-Whyte in [11]. The variant used in this thesis was presented by Wan and Merwe in [25].

2.2.1 The unscented transformation

The UKF is based the concept of the *unscented transformation* which is a method used to approximate the statistics of a random variable, with $\underline{\bar{x}}$, and covariance P undergoing a nonlinear transformation $\underline{y} = \underline{g}(\underline{x})$. This can be done by propagating a large number of realizations of $\underline{x}$ through $\underline{g}(\underline{x})$ and approximating the statistics, but this is computationally expensive. The *unscented transformation* instead uses a small number of carefully chosen samples called *sigma points*, to represent the distribution instead of a large number of realizations of $\underline{x}$. The statistics can then be approximated by a weighted sum and covariance calculation.

In [25] the following method for choosing the sigma points and estimating the statistics of the transformation g of the stochastic vector variable $\underline{x}$ with N_x elements, mean $\underline{\bar{x}}$, and covariance $\hat{P}_x$, is presented:

$$\lambda = \alpha^2(N_x + \kappa) - N_x \tag{2.17}$$

$$\mathcal{X}_0 = \underline{\hat{x}} \tag{2.18}$$

$$\mathcal{X}_i = \underline{\hat{x}} + \left(\sqrt{(N_x + \lambda)\hat{P}_x}\right)_i, \quad i = 1.\ldots, N_x \tag{2.19}$$

$$\mathcal{X}_i = \underline{\hat{x}} - \left(\sqrt{(N_x + \lambda)\hat{P}_x}\right)_i, \quad i = L + 1.\ldots, 2N_x \tag{2.20}$$

$$\mathcal{X} = \begin{bmatrix}\mathcal{X}_0 & \mathcal{X}_1 & \ldots & \mathcal{X}_{2N_x}\end{bmatrix} \tag{2.21}$$

$$W_0^{(m)} = \frac{\lambda}{N_x + \lambda} \tag{2.22}$$

$$W_0^{(c)} = \frac{\lambda}{N_x + \lambda} + (1 - \alpha^2 + \beta) \tag{2.23}$$

$$W_i^{(m)} = W_i^{(c)} = \frac{1}{2(N_x + \lambda)}, \quad i = 1, \ldots, 2N_x \tag{2.24}$$

$$\mathcal{Y}_i = \underline{g}\left(\mathcal{X}_i\right) \tag{2.25}$$

$$\underline{\bar{y}} = \sum_{i=0}^{2N_x} W_i^{(m)} \mathcal{X}_i \tag{2.26}$$

$$\bar{P}_y = \sum_{i=0}^{2N_x} W_i^{(c)} \left(\mathcal{Y}_i - \underline{\bar{y}}\right)\left(\mathcal{Y}_i - \underline{\bar{y}}\right)^T \tag{2.27}$$

where $\underline{\bar{y}}$ and $\bar{P}_y$ are the mean and covariance of the transformed variable.

α, β, and κ are the tuning parameters that decides the positioning of the sigma points and the weights. α determines the spread of the sigma points around the mean and is set somewhere between 0 and 1. Typically a small value of about $1\text{E}{-3}$. β is dependent on the shape of the distribution. $\beta = 2$ is optimal for Gaussian distributions. κ is a secondary scaling parameters and is typically set to zero.

2.2.2 UKF algorithm

The UKF algorithm is a straight forward application of the unscented transformation to the original Kalman filter equations. The variant presented here is based on the version presented by Wan and Merwe in [25].

The algorithm is started from the initial a priori state estimate:

$$\underline{\hat{x}}_0 = E\{\underline{x}_0\} \tag{2.28}$$

$$\hat{P}_0 = E\left\{(\underline{x}_0 - \underline{\hat{x}}_0)(\underline{x}_0 - \underline{\hat{x}}_0)^T\right\} \tag{2.29}$$

The mean and covariance is augmented with the mean and covariance of the disturbance

processes:

$$\hat{\underline{x}}_0^a = \begin{bmatrix} \hat{\underline{x}}_0^T & 0 & 0 \end{bmatrix}^T \tag{2.30}$$

$$\hat{P}_0^a = \begin{bmatrix} \hat{P}_0 & 0 & 0 \\ 0 & Q & 0 \\ 0 & 0 & R \end{bmatrix} \tag{2.31}$$

This augmentation is strictly only necessary when the disturbances are non additive, which they are in the general case. The number of elements in the augmented state is $N_{x^a} = N_x + N_v + N_w$. N_x is the number of system state, N_v the number of process disturbances, and N_w the number of measurement disturbances.

The following steps are then done recursively for $k \in \{1, \ldots, N\}$, where N is the number of available measurements:

1. Calculate $N_i = 2N_{x^a} + 1$ sigma points from the augmented mean and covariance:

$$\mathcal{X}_{k-1}^a = \begin{bmatrix} \hat{\underline{x}}_{k-1}^a & \hat{\underline{x}}_{k-1}^a \pm \sqrt{(N_{x^a} + \lambda)\hat{P}_{k-1}^a} \end{bmatrix} = \begin{bmatrix} \mathcal{X}_{k-1}^x \\ \mathcal{X}_{k-1}^v \\ \mathcal{X}_{k-1}^w \end{bmatrix} \tag{2.32}$$

   The notation $\mathcal{X}^x$, $\mathcal{X}^v$, and $\mathcal{X}^w$ will be used to refer to the rows containing the state, process noise and measurement noise sigma points.

2. Propagate the sigma points through the system equations, $\underline{f}\left(\underline{x}, \underline{u}, \underline{v}\right)$ and $\underline{h}\left(\underline{x}, \underline{u}, \underline{w}\right)$ (Time update):

$$\bar{\mathcal{X}}_{i,k}^x = \underline{f}\left(\mathcal{X}_{i,k-1}^x, \underline{u}_k, \mathcal{X}_{i,k-1}^v\right), \quad i = 0, \ldots, 2N_{x^a} \tag{2.33}$$

$$\bar{\mathcal{Z}}_{i,k} = \underline{h}\left(\bar{\mathcal{X}}_k^x, \underline{u}_k, \mathcal{X}_{i,k-1}^w\right), \quad i = 0, \ldots, 2N_{x^a} \tag{2.34}$$

   i is used to index the columns in the sigma point matrix, which contains the individual sigma points.

3. Calculate the mean and covariance of the predicted state, $\hat{\underline{x}}_k$ and $\hat{P}_k$, and measurement, $\bar{\underline{z}}_k$ and $P_{\bar{\underline{z}}\bar{\underline{z}},k}$, as well as their cross covariance, $P_{\bar{\underline{x}}\bar{\underline{z}},k}$, from the sigma points:

$$\bar{\underline{x}}_k = \sum_{i=0}^{N_i} W_i^{(m)} \bar{\mathcal{X}}_{i,k}^x \tag{2.35}$$

$$\bar{P}_k = \sum_{i=0}^{N_i} W_i^{(m)} \left(\bar{\mathcal{X}}_{i,k}^x - \bar{\underline{x}}_k\right)\left(\bar{\mathcal{X}}_{i,k}^x - \bar{\underline{x}}_k\right)^T \tag{2.36}$$

$$\bar{\underline{z}}_k = \sum_{i=0}^{N_i} W_i^{(m)} \bar{\mathcal{Z}}_{i,k}^x \tag{2.37}$$

$$P_{\bar{\underline{z}}\bar{\underline{z}},k} = \sum_{i=0}^{N_i} W_i^{(m)} \left(\bar{\mathcal{Z}}_{i,k} - \bar{\underline{z}}_k\right)\left(\bar{\mathcal{Z}}_{i,k} - \bar{\underline{z}}_k\right)^T \tag{2.38}$$

$$P_{\bar{\underline{x}}\bar{\underline{z}},k} = \sum_{i=0}^{N_i} W_i^{(m)} \left(\bar{\mathcal{X}}_{i,k}^x - \bar{\underline{x}}_k\right)\left(\bar{\mathcal{Z}}_{i,k} - \bar{\underline{z}}_k\right)^T \tag{2.39}$$

4. Update the estimated state, expressed by $\bar{\underline{x}}_k$ and $\bar{P}_k$ with measurement at time k, $\underline{z}_k$ to the a posteriori state estimate (measurement update):

$$\mathcal{K}_k = P_{\bar{\underline{x}}\bar{\underline{z}},k} P_{\bar{\underline{z}}\bar{\underline{z}},k}^{-1} \tag{2.40}$$

$$\delta\underline{z}_k = \underline{z}_k - \bar{\underline{z}}_k \tag{2.41}$$

$$\hat{\underline{x}}_k = \hat{\underline{x}}_k + \mathcal{K}_k \delta\underline{z}_k \tag{2.42}$$

$$\hat{P}_k = \bar{P}_k + \mathcal{K}_k P_{\bar{\underline{z}}\bar{\underline{z}},k} \mathcal{K}_k^T \tag{2.43}$$

2.2.3 Computing the log likelihood from UKF state estimate

As noted in section 2.1.2 one possible criterion used to optimize the parameters is the likelihood of the measruments $\mathcal{Z}_k$ given the model with parameter vector $\underline{\gamma}$. This is written as $p(\mathcal{Z}_k : \underline{\gamma})$ and is a relative measure of the likelihood that measurements $\mathcal{Z}_k = \underline{z}_1, \underline{z}_2, \ldots, \underline{z}_k$ could be produced by the model with parameters $\underline{\gamma}$. The measurements are independent observations, meaning that the likelihood of the set of all observations up to time k, $p(\mathcal{Z}_k : \underline{\gamma})$, can be written as the likelihood of the observation $\underline{z}_k$ and all prior observations, $\mathcal{Z}_{k-1}$, $p(\underline{z}_k, \mathcal{Z}_{k-1} : \underline{\gamma})$. By using Bayes formula:

$$p(a, b) = p(a|b)p(b) \tag{2.44}$$

the likelihood at time k can then be expressed by a product of conditional likelihoods at the prior time steps:

$$\begin{aligned} p(\mathcal{Z}_k : \underline{\gamma}) &= p(\underline{z}_k, \mathcal{Z}_{k-1} : \underline{\gamma}) \\ &= p(\underline{z}_k | \mathcal{Z}_{k-1} : \underline{\gamma}) p(\mathcal{Z}_{k-1} : \underline{\gamma}) \\ &= \prod_{i=1}^{k} p(\underline{z}_i | \mathcal{Z}_{i-1} : \underline{\gamma}) \end{aligned} \tag{2.45}$$

As the first observation is at $k = 1$, $\mathcal{Z}_0$ is an empty set ant $p(\underline{z}_1 | \mathcal{Z}_0 : \underline{\gamma}) = p(\underline{z}_1 : \underline{\gamma})$.

As the UKF produces a Gaussian estimate of the system state with mean $\underline{\hat{x}}_k$ and covariance $\hat{P}_k$ containing all information from $\mathcal{Z}_k$ given a model with parameter vector $\underline{\gamma}$, the likelihood can be expressed by the state estimate produced by the UKF:

$$p(\underline{z}_k | \mathcal{Z}_{k-1} : \underline{\gamma}) = p(\underline{z}_k | \underline{\hat{x}}_{k-1}, \hat{P}_{k-1} : \underline{\gamma}) \tag{2.46}$$

As the measurement noise is assumed to be Gaussian, the likelihood for $\underline{z}_k$ can be expressed by a normal distribution with mean and covariance equal the predicted measurement mean $\underline{\bar{z}}_k$ and covariance $P_{\underline{\bar{z}}\underline{\bar{z}},k}$ which both are functions of the parameter vector:

$$\begin{aligned} p(\underline{z}_k | \underline{\hat{x}}_{k-1}, \hat{P}_{k-1} : \underline{\gamma}) = \mathcal{N}\left(\underline{z}_k; \underline{\bar{z}}_k(\underline{\gamma}), P_{\underline{\bar{z}}\underline{\bar{z}},k}(\underline{\gamma})\right) = \\ \frac{1}{(2\pi)^{N_z/2} |P_{\underline{\bar{z}}\underline{\bar{z}},k}(\underline{\gamma})|^{1/2}} e^{-\frac{1}{2}\left(\left(\underline{z}_k - \underline{\bar{z}}_k(\underline{\gamma})\right)^T P^{-1}_{\underline{\bar{z}}\underline{\bar{z}},k}(\underline{\gamma}) \left(\underline{z}_k - \underline{\bar{z}}_k(\underline{\gamma})\right)\right)} \end{aligned} \tag{2.47}$$

The likelihood can then be expressed by a product of normal distribution:

$$p(\mathcal{Z}_k : \underline{\gamma}) = \prod_{i=1}^{k} \mathcal{N}\left(\underline{z}_i; \underline{\bar{z}}_i(\underline{\gamma}), P_{\underline{\bar{z}}\underline{\bar{z}},i}(\underline{\gamma})\right) \tag{2.48}$$

Because the natural logarithm is a monotonic function, maximising the logarithm of the likelihood is equivalent to maximising the likelihood. Exploiting the properties of the normal distribution, the logarithm of the likelihood $\ln p(\mathcal{Z}_k : \underline{\gamma}) = \xi(\mathcal{Z}_k : \underline{\gamma})$ can be written as en easier computable sum:

$$\xi(\mathcal{Z}_k : \underline{\gamma}) = -\frac{1}{2} \sum_{i=1}^{k} \left[N_z \ln 2\pi + \ln |P_{\underline{\bar{z}}\underline{\bar{z}},i}(\underline{\gamma})| + \delta \underline{z}_i^T(\underline{\gamma}) P^{-1}_{\underline{\bar{z}}\underline{\bar{z}},i}(\underline{\gamma}) \delta \underline{z}_i(\underline{\gamma}) \right] \tag{2.49}$$

where $\delta \underline{z}_i(\underline{\gamma})$ is the innovation at time i given the parameters $\underline{\gamma}$:

$$\delta \underline{z}_k(\underline{\gamma}) = \underline{z}_k - \underline{\bar{z}}_k(\underline{\gamma}) \tag{2.50}$$

Both the innovation $\delta \underline{z}_i(\underline{\gamma})$ and $P_{\underline{\bar{z}}\underline{\bar{z}},i}(\underline{\gamma})$ are calculated by the UKF from the measurements $\mathcal{Z}_i$ and filter model with the parameters $\underline{\gamma}$. $\xi(\mathcal{Z}_k : \underline{\gamma})$ can thereby be computed recursively as part of the UKF algorithm.

This derivation is derivation is shown many places in the literature [5, 14] and generally the same for all state estimators which provides an Gaussian state estimate.

2.3 Numerical solutions to ordinary differential equations

In addition to MATLABs numerical solvers, the fourth order Runge Kutta method is used where the overhead of the MATLAB solvers is to large. It is used to approximate a continuous differential equation, $\dot{\underline{x}} = \underline{f}(\underline{x})$, as a discrete time difference equation, $\underline{x}_{k+1} = f(\underline{x}_x)$. The integration of $\dot{\underline{x}}$ from $\underline{x}_k = \underline{x}(t_k)$ to $\underline{x}_{k+1} = \underline{x}(t_{k+1})$ with time step $dt = t_{k+1} - t_k$ is estimated by the fourth order Runge Kutta as:

$$\underline{k}_1 = \underline{x}_k + dt\underline{f}\left(\underline{x}_k\right) \tag{2.51}$$

$$\underline{k}_2 = \underline{x}_k + dt\underline{f}\left(\underline{x}_k + \frac{dt}{2}\underline{k}_1\right) \tag{2.52}$$

$$\underline{k}_3 = \underline{x}_k + dt\underline{f}\left(\underline{x}_k + \frac{dt}{2}\underline{k}_2\right) \tag{2.53}$$

$$\underline{k}_4 = \underline{x}_k + dt\underline{f}\left(\underline{x}_k + dt\underline{k}_3\right) \tag{2.54}$$

$$\underline{x}_{k+1} = \underline{x}_k + \frac{dt}{6}\left(\underline{k}_1 + 2\underline{k}_2 + 2\underline{k}_3 + \underline{k}_4\right) \tag{2.55}$$

2.4 Optimization

Optimization is finding the optimum input values, $\underline{\gamma}$, to a function a criterion function, $\mathcal{J}$, that depending on the requirements, either maximizes or minimizes the criterion. As all optimization problems can be regarded as minimization problems by adding a minus sign to the maximization problem, only the minimization problem will be considered from here:

$$\hat{\underline{\gamma}} = \min_{\underline{\gamma}} \mathcal{J}(\underline{\gamma}) \tag{2.56}$$

The most common approach is gradient based approaches. From an initial point in the parameter space, $\underline{\gamma}_0$, the input parameters are moved in the negative direction of the gradient of criterion. One way to look at this is that the criterion defines a terrain above the parameter space with height equal to the criterion value. The parameter vector is allowed glide down to one of the possibly many valleys in this terrain until it reaches the bottom following the steepest slope.

There exists numerous implementations of algorithms that solves optimization problems using this approach. An example of such an algorithm is MATLAB's *fmincon*. *fmincon* also allows the boundaries of the parameter space to be defined with linear and nonlinear boundaries.

There can possibly be many local minimums of the criterion, i.e. valleys in the criterion terrain that does not reach down to the absolute lowest value of the criterion. The minimum with the lowest criterion of all minimums is called the global minimum, and it is often desired to find the global minimum. Gradient based optimization algorithms can for many criteria functions have problems with finding the global minimum because the criteria are filled with local minimums. One possible solution is to optimize the criterion from a large amount of initial parameters and hope that one of the initial set of parameter is close enough to the global minimum to find them, but this can, depending on dimensionality of the problem be very time consuming.

2.4.1 Particle swarm optimization

One of the many algorithm that tries to solve the global optimization problem is the *particle swarm* algorithm. The version described here is the one found in MATLAB's *Global Optimization Toolbox* based on the deception given by [8, 9]. Particle swarm optimization is an biologically inspired optimization algorithm, inspired by the flocking of birds and insects . The algorithm starts with an initial set of particles in the parameter space. The particles are either drawn randomly from the set of allowable parameters or to values assumed to be close to the optimal values. The particles are initially assigned a random velocity. For each

iteration of the algorithm, the criterion is evaluated at all the particles. The particles velocities are then modified by a weighted sum of their current velocity, the position of with the lowest criterion, and the position with the lowest criterion in the particles immediate neighborhood. This makes the particles attracted to positions in the parameter space with known good criterion values, but retains some inertia, so they will not necessarily pass through the exact location of the minimum as they fly around.

Initially the particles will fly around in the parameter space more or less randomly. As good criterion values are found, particles will start to gather in swarms around the minima. As better criterion values are discovered the swarm will gradually migrate to the currently best known solution. As the algorithm does not know if any of the particles actually are at a minimum of the criterion, algorithm must use some other stopping criteria. This is either a maximum number of iterations, number of iterations with no improvement in best criterion, a time limit, or if a criterion value better than some set threshold is found. Even tough the aim of the algorithm is to improve the chances of finding a global minimum it is not guaranteed to be found.

According to the article describing the original version of the Particle Swarm Optimization algorithm [15] it is particularly well suited for nonlinear and discontinuous problems and consistently found the global minimum of the Schaffer f6 function which is a popular benchmark for genetic optimization algorithms. Though it has to be noted that the algorithm described in [15] differs slightly from the one described by [8, 9], but the general idea is the same.

2.5 Bond graph modeling

Bond graph modeling is a technique for modeling systems that involves power interactions. It can be applied to both collections of subsystems and the power interactions internally in the subsystems. The general idea is to connect together different elements by power bonds consisting of an oppositely directed flow and effort. A flow can be velocity, current and so on, while efforts are voltages, torques, forces etc. Elements can have one or more ports the accepts a bond.

This section is based on [13] by Karnop, Margolis, and Rosenberg and the appendix about bond graph modeling in [20] by Mashadi and Crolla. This is only a small overview of bond graph modeling, sufficient for understanding the bond graphs presented in this thesis.

2.5.1 Ports and bonds

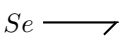
A port is a power connection of a system. It contains either a effort input and a flow output or and effort output and a flow input. An element with more than one port is called a multiport. Connection between ports are called bonds. Figure 2.1 shows the symbol used to represent the bond. The arrow points in the direction of the power flow.

Figure 2.1: Bond graph symbol for a bond.

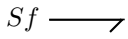
2.5.2 Standard elements

Sources

There are two different elementary sources in bond graph modeling: sources of effort, Se, and sources of flow, Sf. These are used as inputs to the models as well as energy sources in the model. Figure 2.2 shows the symbols used to represent the sources.



(a) Source of effort



(b) Source of flow

Figure 2.2: Bond graph symbols for sources of effort and sources of flow.

Junctions

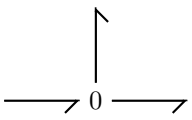
Junctions are multi port elements connecting two or more bonds. Junctions are energy conserving and therefore consumes no power. There are two different kinds of junctions: 0-junctions and 1-junctions. In a 0-junction effort is preserved. This means that all efforts are equal, and because it does not consume any power the flows must sums to zero:

$$e_i = e_j, \quad \sum_i f_i = 0 \tag{2.57}$$

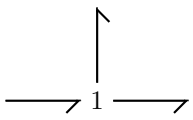
A 1-junction preserves flow, and therefore has the complete opposite behaviour:

$$f_i = f_j, \quad \sum_i e_i = 0 \tag{2.58}$$

Figure 2.3 shows the symbols for 0 and 1 junctions.



(a) 0-junction



(b) 1-junction

Figure 2.3: Bond graph symbols for 0-junctions and 1-junctions.

Resistive elements

Resistive elements, R, are consumers of energy. They are one port elements representing loss of energy in the system. Examples of resistive elements are electrical resistors, mechanical dampers and friction. The general equation governing a resistive elements is:

$$e = \Phi_R(f) \tag{2.59}$$

where Φ_R is some function relating the flow to the effort. When Φ_R is linear the relationship is simply:

$$e = bf \tag{2.60}$$

Unsurprisingly this is directly equivalent to Ohm's of resistance. Figure 2.4 shows the bond graph symbol for a resistive element.

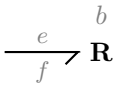


Figure 2.4: Bond graph symbol for a resistive element.

Capacitive elements

Capacitive elements, C, serves as energy storage elements in the form of displacement or charge. They can for example represent capacitors and springs. The displacement is defined as the integral of flow:

$$q = \int f\, dt \tag{2.61}$$

The resulting effort is governed by the nonlinear capacitance function Φ_C:

$$e = \Phi_C^{-1}(q) \tag{2.62}$$

or in the linear case:

$$e = \frac{q}{c} \tag{2.63}$$

C is directly equivalent to the capacitance of a capacitor or the inverse of the spring constant, k, if used to model a spring. For a spring the charge, q, is equivalent with the displacement:

$$e = kq \quad \leftrightarrow \quad F = kx \tag{2.64}$$

Figure 2.5 shows the bond graph symbol for a capacitive element.

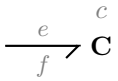


Figure 2.5: Bond graph symbol for a capacitive element.

Inertial elements

Inertial elements, I, stores energy as momentum. Inertial elements can be used represent masses, moments of inertias or inductive coils. The momentum, p, is defined as the integral of the effort:

$$p = \int e\, dt \tag{2.65}$$

and in the general case the resulting flow is governed by the nonlinear function Φ_I:

$$f = \Phi_I^{-1} \tag{2.66}$$

ans in the linear case the inertia parameter i:

$$f = \frac{p}{i} \tag{2.67}$$

i can be looked at as mass, moment inertia or inductance depending on the system that is being modelled. Figure 2.6 shows the bond graph symbol for an inertia element.

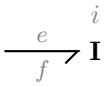


Figure 2.6: Bond graph symbol for a inertial element.

Transformers

Transformers, TF, are two port elements with a proportional relationship between incoming effort, e_i, and outgoing effort, e_o:

$$e_o = me_i \tag{2.68}$$

As only R-elements dissipate energy the reverse relation must apply to the flow:

$$f_i = mf_o \tag{2.69}$$

Transformers are used to model levers, speed ratios and electrical transformers. Figure 2.7 shows the bond graph symbol for a transformer.

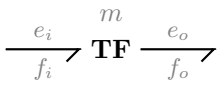


Figure 2.7: Bond graph symbol foe a transformer element.

Gyrators

In addition to the transformer there is another two port element, the gyrator, GY. Gyrators has a relationship between the outgoing flow and the ingoing effort:

$$f_o = re_i \tag{2.70}$$

and to keep the power through the gyrator constant:

$$f_i = re_o \tag{2.71}$$

where r is called the gyrator ratio. A typical example of a gyrator is an electric motor. The torque (effort) of the output shaft is equal to the current (flow) and the speed (flow) of the output shaft is equal to the voltage (effort). Figure 2.8 shoes the bond graph symbol for a gyrator.

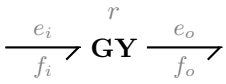


Figure 2.8: Bond graph symbol foe a gyrator element.

2.5.3 Modulated elements

The elements in the previous section can also come in varieties with modulated relationships. This means that the function or parameter which influences the relationship the efforts and flows can be changed according to one or more modulation parameters. This can for example be a variable resistance, a transform representing a variable gearing ratio, a variable source of effort or flow etc. A modulated element is marked by putting and M in front of the usual symbol.

2.5.4 Deriving the equations of motion from a bond graph model

Typically the goal of a bond graph model is to derive a set of equations describing dynamics of the system that is being modeled. This usually ends up as a state space model, where the states are defined by the I- and C-elements. The first step in the process of finding this state space model is to assign causality to the bonds in the model, by determining weather it is effort or flow which is the *cause* and *effect*.

Causality strokes are used to mark the causality of the bonds. For the element at the same end of bond as the stroke the effort is defined as the cause, and the flow as the effect. Figure 2.9 may show this more clearly.

Bonds from sources have uniquely defined causality. Effort from a source of effort is always the *cause* while the flow is always the *effect* and inversely for a source of flow. Resistive elements are indifferent to the causality and can therefore be assigned either way. Bonds to transformers must have opposite causality strokes meaning that the flow is the

(a) Effort acting on P is the *cause* and flow from P is the *effect*

(b) Flow going to P is the *cause* and effort from P is the *effect*

Figure 2.9: Causality strokes for sources acting on a general one port element P.

cause on one side and *effect* on the other, while for gyrators causality must be equal for both bonds.

The causality for C- and I-elements can in principle be placed either way, but it is strongly advised to place it in such a way that it results in an integral relation. For C-elements this means that the flow is the *cause* and for I-elements that effort is the *cause* which means the *effect* of either can be described by an integral. Or more conveniently the derivative of the momentum, p, of an I-element is the effort, e, of the bond:

$$\dot{p} = e \tag{2.72}$$

and derivative of the momentum, q, of a C-element is the flow, f, of the bond:

$$\dot{q} = f \tag{2.73}$$

Due too how the junctions are defined, the preserved quantity, which is equal for all the connected bonds, can only be the cause of one of the bonds and the effect of all the others. The bond with the *cause* causality is called the strong bond. For a 0-junction this means that that the strong bond has the causality stroke pointed at the junction and for a 1-junction that the strong bond has the causality stroke pointed away. Figure 2.10 shows two very simple bond graphs with correct causality for the junctions and C- and I-elements with integral causality.

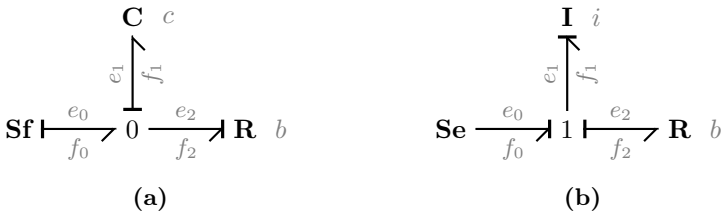


Figure 2.10: 0-junctions and 1-junctions with valid causality configuration and C- and I-elements with.

The bond graph shown in fig. 2.10a has one active element, a capacitance. The capacitance has integral causality meaning that the derivative of charge is equal to the flow of the bond: $\dot{q_1} = f_1$, *causing* the effort e_1 to be $e_1 = q/c$. As it is connected to a 0- or equal effort junction, the sum of flows must be zero: $f_0 - f_1 - f_2 = 0$, meaning that $f_1 = f_0 - f_2$. f_0 comes from comes from a source of effort and is thereby defined to be: $f_0 := Sf$. f_2 is caused by the effort applied to the resistance which is defined as: $f_2 = e_2/b$. As the resistance and connected to a 0-junction the effort, e_2, is equal to the other efforts, but as the C-element is given the strong bond its effort is assumed to be the *cause* of the others. f_2 is therefore $f_2 = e_1/b = q/(bc)$ and the differential equation for the system can be written as:

$$\dot{q} = Sf - \frac{q}{bc} \tag{2.74}$$

Similarly the equation for the bond graph in fig. 2.10b can be shown to be:

$$\dot{p} = Se - b\frac{p}{i} \tag{2.75}$$

because the I element has integral causality, $e_1 = e_0 - e_2$, $e_0 = Se$, $e_2 = bf_2$, $f_0 = f_1 = f_2$, and $f_1 = p/i$. The methods for finding the equations for larger bond graphs folows the same procedure: identify elements with integral causality, define the input or flow or effort in terms of sources and the outputs of other elemens using the rules for the 0- and 1-junctions. Finding the equations for bond graph models with derivative causality can be substantially harder and is not covered here.

These two examples also demonstrate the typical numbering conventions used when working with bond graphs: sources are number 0 and 00 and so on if there are more than one. The numbering starts at one with all elements with integral causality, then derivative causality, then passive elements, and finally the remaining bonds unnumbered bonds.

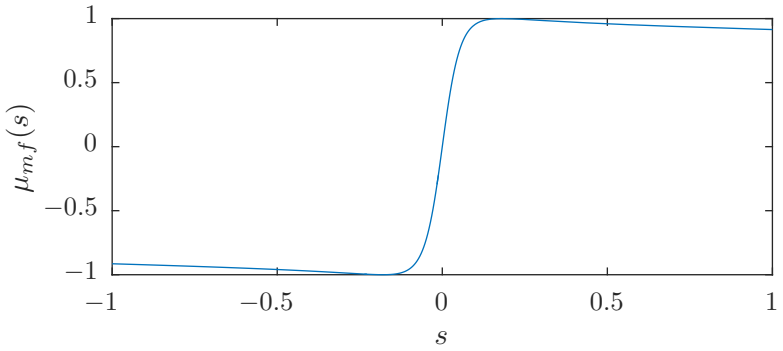


Figure 2.11: Typical curve generated by the magic formula for dry asphalt.

2.6 Ground vehicle modeling

For most parts of the vehicle model developed in this thesis is derived using simple Newtonian Physics has been sufficient. For the tires and engine, existing models from the literature where used. Although a new model for the rubber belt CVT is developed, a description of the working principle as well as some modeling efforts described in the literature will be presented.

2.6.1 Tire modeling

A good tire model is of vital importance when developing a vehicle model, as all force that influence the vehicle dynamics, excluding air resistance, are transfered through the tires. As only longitudinal dynamics are under consideration, it is only necessary to model the longitudinal traction force and the rolling resistance.

Longitudinal traction force

A well regarded tire model is the *magic tire* formula, originally developed by Hans B. Pacejka [6, 10, 24]. It is an empirical model relating the tire slip (a ratio or percentage describing how much the tire is sliding) to a friction coefficient which in combination with the tire normal force is used to calculated the tractive force of the tire. The friction coefficient is calculated by the following formula:

$$\mu_{mf}(s) = D\sin\left(C\arctan\left(Bs - E\left(Bs - \arctan(Bs)\right)\right)\right) \tag{2.76}$$

where B, C, D, and E are tuning parameters. fig. 2.11 shows a typical curve generated by the magic tire formula. The parameters can be derived from a larger set of tire physical tire parameters, but are usually determined experimentally to give the best fit. The resulting traction force is a product of the coefficient of friction calculated by the magic formula and the normal force:

$$F_x(s, F_z) = \mu_{mf}(s)F_z \tag{2.77}$$

This linear relationship is generally only valid close to the normal force the tire is designed for as excessive tire load actually reduces the coefficient friction by compressing the rubber and thereby making it stiffer an less able to conform to irregularities in the ground surface.

There are a few different formulations of the tire slip which is a relative measure of how much the tire is slipping. A free rolling tire has zero slip. A tire which is sliding and not rotating has, depending on formulation 1, 100 % or ∞ slip. In this thesis the formulation that gives slip as ratio in the range -1 to 1 and uses the same equation for slip during acceleration and braking is used:

$$s = \frac{\omega_w r_{\text{eff}} - v}{\max(|\omega_w|, |v|, v_{\min})} \tag{2.78}$$

v is the linear velocity of the wheel center, ω_w is the angular velocity of the wheel, and the product $\omega_w r_{\text{eff}}$ gives the linear velocity of the tire surface. The effective tire radius, r_{eff}, is

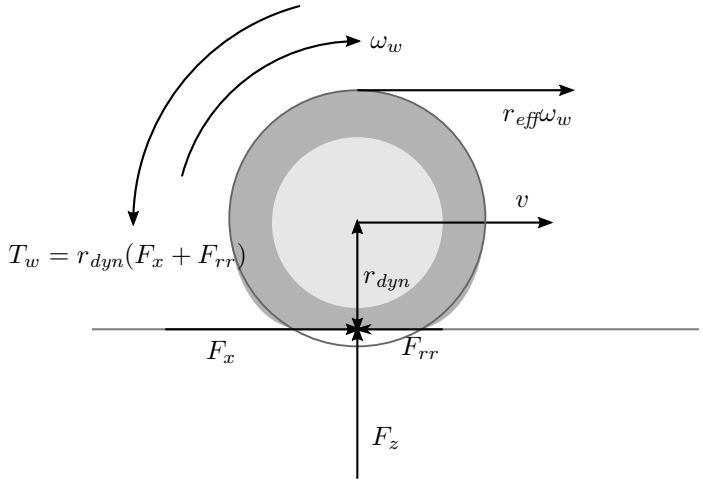


Figure 2.12: Tire forces, torques and velocities.

calculated from the effective circumference, which is the distance the wheel center travels with one full rotation of the wheels with no slipping. r_{eff} is not equal to the dynamic tire radius, r_{dyn}, which is the distance from the wheel center to the tire ground contact when the tire is loaded. Neither is it necessary equal to the static tire radius, r_{stat}, which is the unloaded tire radius. The v_{min} is added to the denominator in the slip calculation to avoid the singularity that occurs when both v and ω_w are zero. fig. 2.12 summarizes some of the tire forces and velocities.

The *magic formula* is a description of stationary tire forces. For slow changes it is still sufficient, but for situations where the slip changes rapidly, like during hard acceleration the dynamic tire behaviour must be take into account. A crude approximation to dynamic tire behaviour is the introduction of a first order time delay as proposed in [24]:

$$\dot{F}_x = \frac{1}{\tau}\left(F_{x,\text{stat}} - F_x\right) \tag{2.79}$$

where $F_{x,\text{stat}}$ is the static friction force calculated by the magic tire formula.

Rolling resistance

The rolling resistance is primarily caused by deformation of the tire as it is rolling. Other factors can be the ground surface irregularities and deformation of the ground. It is therefore reasonable to assume that the rolling resistance torque is a function of the normal force, tire pressure, tire construction, and the angular velocity of the wheel. According to [24] the following is a good approximation:

$$\mu_{rr} = \mu_0 + \mu_1 v^2 \tag{2.80}$$

$$F_{rr} = \mu_{rr} F_z \tag{2.81}$$

where μ_0 and μ_1 has to be found experimentally, and $\mu_0 = 0.015$ and $\mu_1 = 7\text{E}{-6}\,\text{s}^2\,\text{m}^{-2}$ are typical values for a normal car. The low speed rolling resistance is therefore primarily dominated by the static component of μ_r, μ_0.

2.6.2 Engine modeling

The most common method for simulating an internal combustion engine in a vehicle dynamic model, is to model the torque as a nonlinear function of throttle position, α, and angular velocity of the crankshaft, ω_e. Friction in the engine may or may not already be taken into account by the torque function. The engine torque, T_e and the torque load connected to the engine, T_l, acts on an inertia, I_e, representing the engine's crankshaft, cylinders, and flywheel. This model can be summed up with the following differential equation:

$$\dot{\omega_e} = \frac{1}{I_e}\left(T_e\left(\alpha, \omega_e\right) - T_l\right) \tag{2.82}$$

$T_e(\alpha, \omega_e)$ can be represented by a lookup table derived from experimental data or as a function of some of the engine parameters. The parameters are typically maximum torque or power, the angular velocity producing the maximum, the angular velocity at idle, fuel type, and other known quantities of the engine. An example of such a torque function is the *Magic Torque Formula* presented in [20]:

$$T_e(\alpha, \omega_e) = [1 + \exp(A - B\alpha)]^{-C\omega_e^D} T_{e,wot}(\omega_e) \tag{2.83}$$

Here $T_{e,wot}(\omega_e)$ is the torque curve with the throttle fully open, and A, B, C, and D are coefficients that scales the torque at different throttle settings and angular velocities. In [10]

$$T_{e,wot} = P_1 + P_2\omega_e + P_3\omega_e^2 \tag{2.84}$$

where the coefficients, P_1, P_2, and P_3 are defined by the peak engine power, $P_{e,\max}$, and the angular velocity of the peak, $\omega_{e,P_{e,\max}}$:

$$P_1 = \frac{P_{e,\max}}{\omega_{e,P_{e,\max}}} \quad P_2 = \frac{P_{e,\max}}{\omega_{e,P_{e,\max}}^2} \quad P_3 = -\frac{P_{e,\max}}{\omega_{e,P_{e,\max}}^3} \tag{2.85}$$

An internal combustion engine will also have some time delays associated with it. There is a delay in opening the throttle valve, air travelling through the intake manifold, and a delay caused by the fact that no positive torque is produced unless one of the cylinders are firing. For many applications it is sufficient to model this as a time delay between the throttle and throttle valve, but depending on the on the model they may be so small that they can be ignored.

2.6.3 Rubber Belt Continuously Variable Transmission

CVTs are transmissions that can vary their speed ratio continuously between a lower and upper bound. The benefit of a CVT is that the engine can kept at its optimum operating speed for a larger range of vehicle speeds than with a conventional transmission with a discrete set of gears. The CVT variant presented here is the rubber bely CVT. The primary source for the section has been the Service Manual for the Polaris Ranger [7] which contains a brief overview of the working principle of Polaris's Polaris Variable Transmission (PVT) system as well as multiple diagrams of the different parts. Even though Polaris calls it a PVT and not CVT the studies [12, 23] of other CVT systems describes identical working principle.

A rubber belt CVT accomplishes the continuous speed ratio variation with two V-pulleys and a rubber V-belt. A V-belts and V-pulleys have V shaped cross sections and only the sides of the belt should be in contact with the pulley. The pulley connected to the engine is called the primary pulley and the pulley connected to the output shaft is called the secondary pulley. The pulleys consists of two conical sheaves with variable distance between them. When the sheaves of one of the pulleys are pressed together with greater force than the sheaves of the other pulley, the belt gets forced outward in the pulley grove of the pulley experiencing the greater force and pulled inwards in the other. The effective radius of the two pulleys have thereby changed. As the speed ratio is dependant on the radii of the pulleys it has also changed. The speed ratio is obtained by:

$$R_{\text{cvt}} = \frac{r_s}{r_p} \tag{2.86}$$

where r_p is the radius of the primary pulley and r_s is the radius of the secondary. Increasing the radius of the primary pulley while increasing the radius of the secondary reduces the speed ratio of the CVT and is generally referred to as up-shifting. The opposite action decreases the speed ratio and is called down-shifting. A diagram of the working principle of a rubber belt CVT is shown in fig. 2.13 on the next page.

The actuators that produces the clamping force on the primary pulley, F_p, and secondary pulley, F_s, and the control system used can vary. The actuators and control system used in the CVT in the Polaris Ranger are completely mechanical. On the primary side a system

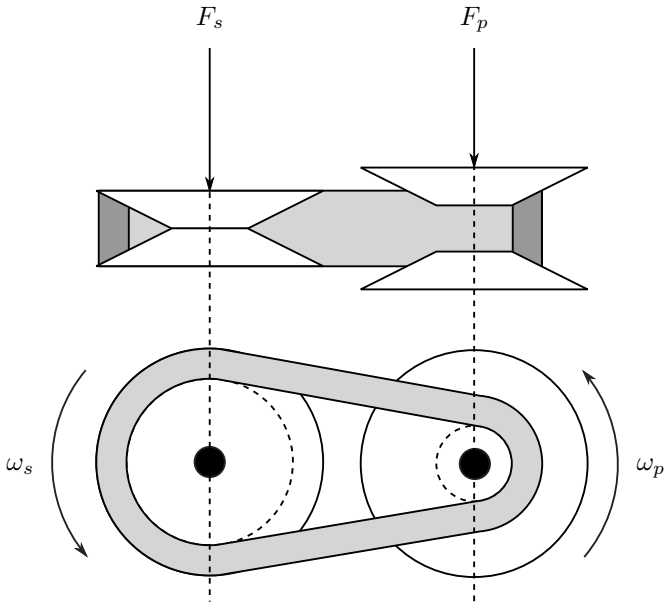


Figure 2.13: Simplified diagram of a rubber belt CVT. Shows the two pulleys consisting of two conical sheaves each from a above and straight on the rational axes. The actuators that produces the primary and secondary clamping forces, F_p and F_s, are not included.

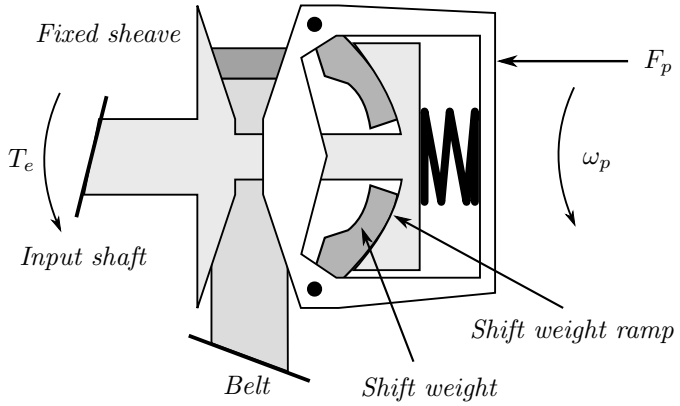


Figure 2.14: Simplified diagram of primary pulley speed sensing device. The clamping force on the primary pulley, F_p, is produced by weights, called shift weights.

of centrifugal weights, called *shift weights*, is used. Making the primary pulley clamping force primary dependant on the engine speed. The device fitted to the secondary pulley is compressed with a spring and compresses the pulley more when torque is applied.

fig. 2.14 shows a simplified digram of the primary pulley and the shift weight system. The assembly consists of a fixed pulley sheave connected to the input shaft and moveable pulley sheave. A spring forces the pulley to be open. A set of shift weights are used to make the compression of the pulley speed dependant. As the speed of the primary pulley the shift weights are forced outward. Due to the pivot point of the shift weights and the ramp they push against this compresses the pulley. The spring, pivot point, weight and shape of the shift weights, and the curve of the shift weight ramp, influences the force which compresses the primary pulley and are tuned to give the desired response.

When the engine is at idle the tension in the spring is sufficient to to contract the force exerted by the shift weights and the belt is not in contact with the sides of the primary pulley. Initial engagement of the transmission is thereby also controlled by the primary pulley.

Figure 2.15 on the next page shows a simplified diagram of the secondary pulley. While the primary pulley is sensitive to the engine speed, secondary pulley is sensitive to the torque

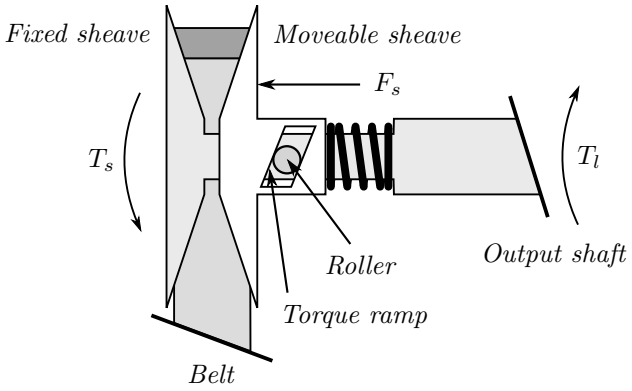


Figure 2.15: Diagram of secondary pulley torque sensing device. T_s is the torque applied to the secondary pulley, and T_l is the load connected to the CVT. The fixed pulley sheave is connected to the output shaft. The moveable pulley sheave can rotate about and move axially along the output shaft restricted by the roller and torque ramp as well as the secondary pulley spring. The roller is fixed to the output shaft.

experienced by the secondary pulley, T_s. The secondary pulley also consists of a fixed and movable pulley sheave. The fixed sheave is connected to the output shaft. The movable and fixed pulley sheaves are pressed together by a spring. The movable pulley sheave can rotate a small amount in relation to the fixed sheave, but this rotation is coupled to the distance between two sheaves by a ramp and a roller. Torque applied to the secondary pulley will therefore add to the force compressing it.

Most of the rubber belt CVT models found in the literature are models are meant to aid in the design of these transmissions, they are therefore highly complex, high order models, taking into account all the dimensions and physical interactions [3, 4, 12]. They are therefore not found suitable for use in the model developed in this thesis.

One of the simplest CVT model in the literature is the following model[16]:

$$\dot{i_{\text{cvt}}} = m\left(i_{\text{cvt}}\right)\omega_p F_s\left(kpks - \frac{F_p}{F_s}\right) \tag{2.87}$$

i_{cvt} is the speed ratio of the CVT. ω_p is the angular velocity of the primary pulley, F_p and F_s are the actuating forces acting on the primary and the secondary pulleys, $kpks$ is the force ratio required to hold a steady speed ratio and $m\left(i_{\text{cvt}}\right)$ is a factor relating the change of speed ratio to the current speed. $kpks$ and $m\left(i_{CVT}\right)$ has to be found experimentally. Although this is not a complete model as the primary and secondary pulley actuators are not included.

Chapter 3

Unmanned vehicle platform OLAV

OLAV is an experimental autonomous vehicle under development by FFI. It is based on a Polaris Ranger XP 900 EPS light off-road utility vehicle. The ranger has been outfitted with actuators and instrumentation necessary for the vehicle to be controlled by a computer.

3.1 Polaris Ranger XP 900 EPS

The Ranger XP 900 EPS is light off-road utility vehicle produced by Polaris Inc. A picture of this specific model is shown in fig. 3.1 on the following page. Polaris is primarily known for their snowmobiles, but have diversified to all terrain, recreational and utility vehicles. The Ranger therefore incorporates some technologies that are more common in snowmobiles, motorcycles and quad-bikes than traditional cars. An example of this is the rubber belt CVT.

The XP 900, signifies that this is the "Xtreme Performance" version of the Ranger with an gasoline engine with around 900 cm^3 cylinder volume (actual volume is 875 cm^3). EPS means that this is a version with electric power steering. A very useful feature in a car that will be electronically controlled as no additional steering actuator is required. The vehicle has three seats and a cargo box. The Ranger has selectable rear- or four-wheel drive and lockable differentials, two forward gears, reverse, neutral and park in addition to the gearing ratio provided by the CVT. An general overview of the specifications of the vehicle is provided in table table 3.1 on the next page.

The Ranger XP 900 is equipped with a CVT, marketed by Polaris as a PVT. The PVT is a rubber belt CVT. Polaris claims that the PVT should keep engine speed constant for a given throttle setting regardless off the load and that full throttle should keep the engine at the peak of the power curve [7]. The mechanics of the CVT is presented in greater detail in section 2.6.3. In addition to the PVT the Ranger also have a conventional transmission between the CVT and the wheels. It provides the High and low gear ranges, reveres, neutral and park.

3.2 Modifications

The conversion of the ranger to an unmanned vehicle has required a large number of modification in the form of additional instrumentation, computers, and actuators, as well as additional batteries and power delivery system. Figure 3.2 shows a picture of OLAV in its current state. The modifications has added about 500 kg to the mass of the vehicle, which can be seen by comparing the ground clearance of the vehicle in the two pictures.

3.2.1 Actuators

Actuation of the steering uses the existing power steering system of the ranger. This is done by fooling the sensor that measures the torque on the steering column. To precisely set the

Figure 3.1: Picture of a standard Polaris Ranger XP 900 EPS

Source: Polaris Industries Inc.

Dry weight	617 kg
Wheelbase	2.057 m
Track width	1.473 m
Engine	4-stroke DOHC twin Cylinder
Displacement	875 cm^3
Fuel system	Electronic fuel injection
Engine idle speed	1250 ±50 RPM
Engine max operating speed	7250 RPM
Driving system type	PVT, selectable 4-wheel drive, lockable differential
Transmission speed ratio high range	10.4:1
Transmission speed ratio low range	25.59:1
Transmission speed ratio reverse	22.92:1
Tire diameter front	0.66 m
Tire diameter rear	0.66 m
Tire Pressure front	65 kPa
Tire pressure rear	83 kPa
Brake system	4 wheel hydraulic disc brakes (two piston front calipers and one piston rear calipers)

Table 3.1: General specifications for Polaris Ranger XP 900 EPS.[7]

Figure 3.2: Picture of OLAV.

Source: FFI

Actuator	Range
Steering angle	$-30°$ to $30°$
Throttle	0 % to 100 %
Brake pedal travel	0 % to 100 %
Gear selection	Low/High/Neutral/Reverse/Park

Table 3.2: Actuators fitted to OLAV and their control range.

steering angle it is also outfitted with a sensor that measures the steering angle. This measurement is then used for control of the steering angle. As the vehicle has drive-by-wire throttle it can also be controlled directly. For safety reasons the brakes are actuated by an electric actuator that depresses the brake pedal. This insures that the brakes can be operated by any occupants in the vehicle in case of an emergency. The original gear selector is removed an replaced by an electric actuator. All the actuators are connected to a programmable logic controller (PLC) that can be accessed by the rest of the control system through Ethernet using the Modbus TCP/IP protocol. The actuators and the PLC that controls them where installed by Maritime Robotics.

3.2.2 Instrumentation

The speed of the rear wheels is measured at the differential by using a sensor measures that detects teeth of the final drive gear. This gives a reasonably accurate but noisy reading and no information about the direction. Engine speed is read from the engine management system.

The navigation system is provided by the navigation research group at FFI. It consists of high precision inertial measurement unit (IMU) unit and GNSS as well as the filters required for a precise position estimate. It provides precise measurements of the vehicles speed, orientation and position.

The software on OLAV runs on a Intel NUC computer. This is a standard commercial of

the shelf computer with an x86 processor. The operating system is Ubuntu Linux with the addition of the ROS framework. Communication with senors or actuators are done through either USB or Ethernet.

Chapter 4

Vehicle model

In this chapter the developed mathematical model for the longitudinal dynamics of OLAV is presented, how it was derived and the assumptions which were made. The approach taken was to first brake the vehicle down to a set of subsystems, deciding models for the individual subsystems and finally combining the models into a complete model of the longitudinal dynamics of the vehicle. At the end of the chapter a summary of the parameters in the model is given and simulations of the model presented.

4.1 The intended use of the model and limitations

When designing a model of physical systems it is important to take into consideration what the model will be used for and what information is available about the system. Figure 4.1 on the following page shows a fairly complete bond graph model of the subsystems of the Polaris Ranger. The front differential, drive shafts, and transfer case are removed for simplicity as the four-wheel drive system is usually disabled. Even with this slight simplification, the vehicle is obviously a complex system.

Figure 4.2 on page 29 shows a further simplified model. In this simplified model, only the longitudinal dynamics are taken into account. The steering system can therefore be removed. As the vehicle is assumed to not be turning, the left and right wheels can be assumed to be turning at the same rate. The left and right tires on the front and rear axles can therefore be combined and the differentials can be removed. This simplified bond graph model will be used to develop the vehicle model. In the following sections the different subsystems will be examined and combined to a complete model.

4.2 Engine model

The engine is modeled as an rotational inertia element, I_e, a drive torque, T_e, that is a function of the angular velocity of the engine and the throttle setting, and the torque load applied to the engine, T_l.

$$\dot{\omega_e} = \frac{1}{I_e}\left[T_e\left(\alpha, \omega_e\right) - T_l\right] \tag{4.1}$$

The engine torque is modeled by a linear combination of an approximation of the torque curve at full throttle, and viscous friction with coefficient b_e at zero throttle:

$$T_e\left(\alpha, \omega_e\right) = \alpha T_{\text{wot}}\left(\omega_e\right) - (1-\alpha) b_e \omega_e \tag{4.2}$$

As should be apparent the throttle is assumed to be in the range 0 to 1. The torque curve approximation used is from [10] and is defined by the peak power of the engine, P_M, and the angular velocity producing the peak power ω_M.

$$T_{wot}(\omega_e) = \sum_{i=1}^{3} P_i \omega_e^{i-1} = P_1 + P_2\omega_e + P_3\omega_e^2 \tag{4.3}$$

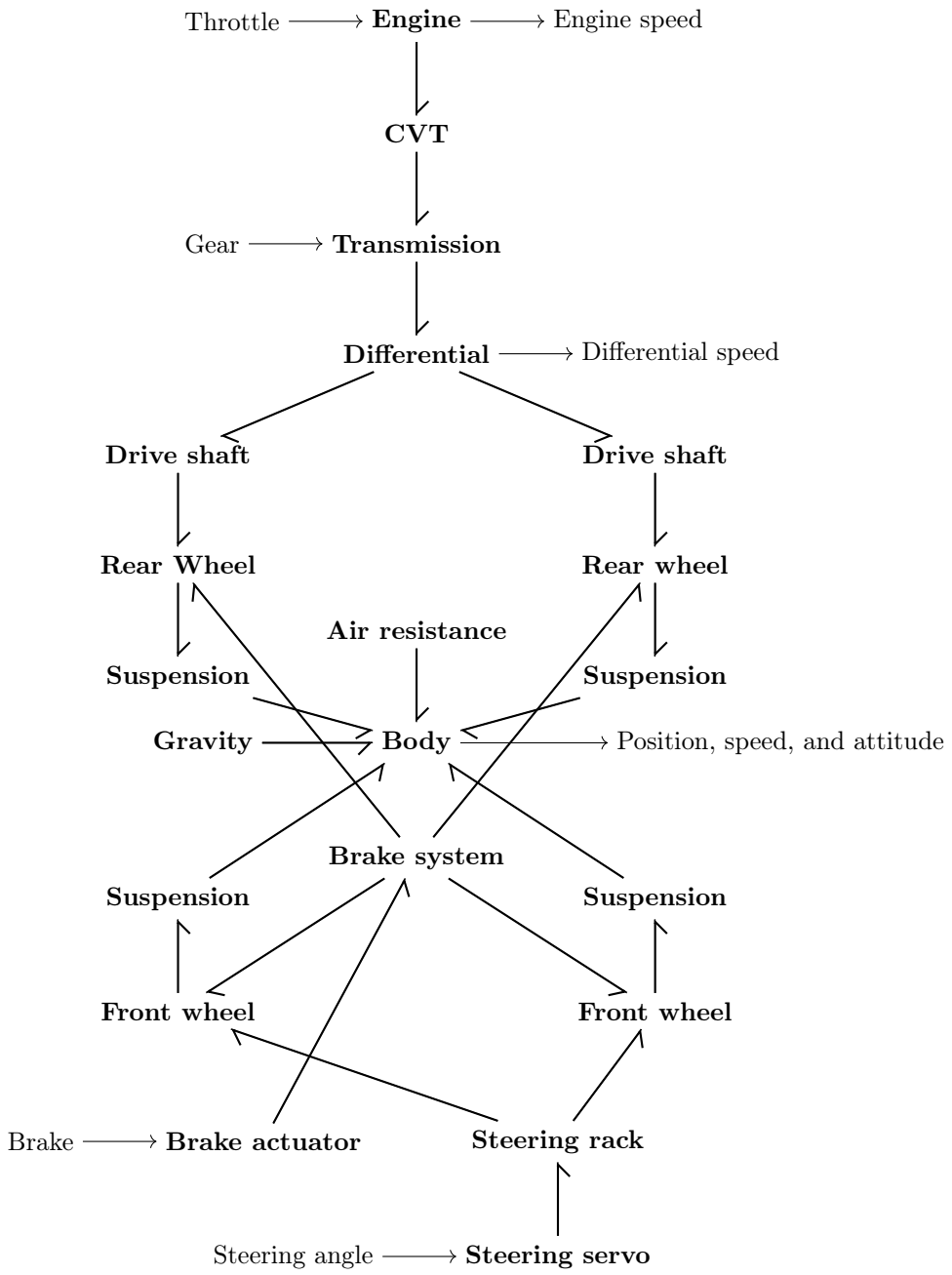


Figure 4.1: Bond graph model of Polaris Ranger subsystems. Transfer case and front differential and drive shafts are removed for simplicity.

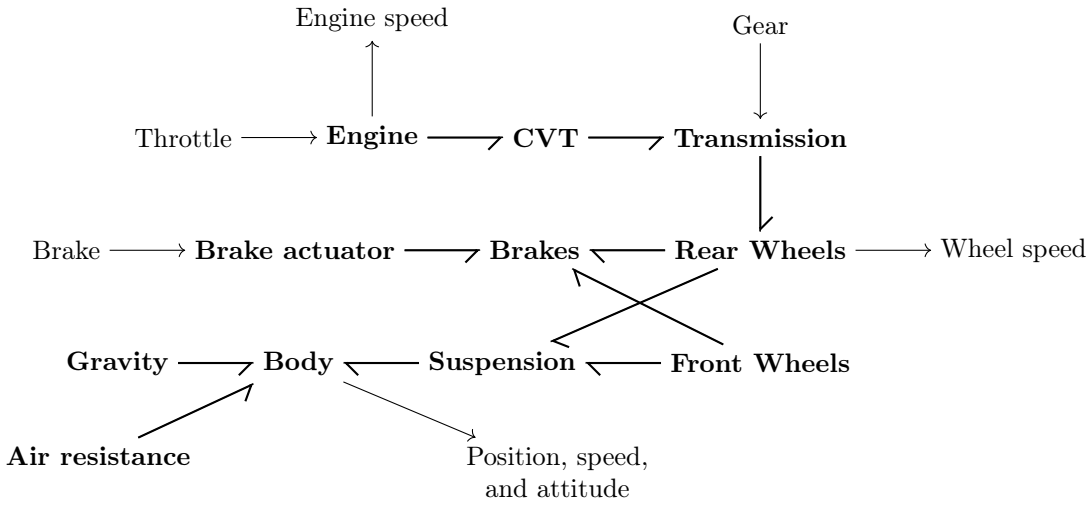


Figure 4.2: Reduced bond graph model of vehicle subsystem.

$$P_1 = \frac{P_M}{\omega_M}, \quad P_2 = \frac{P_M}{\omega_M^2}, \quad P_3 = -\frac{P_M}{\omega_M^3} \tag{4.4}$$

As the engine has to be able to idle, the throttle has to be partially open at idle. This controlled by the engine management system. The engine management also restricts the maximum engine speed to $\omega_{e,\max}$ by cutting the throttle when it is reached. The vehicle also has an optional speed limiter that works by cutting the throttle. To accommodate these constraints a simple engine controller has to be part of the engine model.

The idle controller is modeled by calculating the throttle needed to keep the engine at $\omega_{e,\text{idle}}$ by solving eq. (4.2) for $\omega_{e,\text{idle}}$ with $T_e = 0$. Additional throttle proportional to the error is added when falling bellow $\omega_{e,\text{idle}}$:

$$\alpha_{\text{idle}} = \frac{b_e \omega_{e,\text{idle}}}{T_{wot}(\omega_{e,\text{idle}}) + b_e \omega_{e,\text{idle}}} + p_{\text{idle}} \left(\omega_{e,\text{idle}} - \omega_e\right) \tag{4.5}$$

p_{idle} is a tuning parameter for the proportional part of the idle controller. As both the rev limiter and speed limiter cuts the throttle when either the engine or vehicle speed reaches set limits they are simply modeled by setting the throttle to zero when the limits are exceeded. These rules for the throttle setting applied by the engine controller, α_c, can be summarized with the following piecewise equation:

$$\alpha_{\text{c}}\left(\alpha, \omega_e, v_w\right) = \begin{cases} 0 & \text{if } \omega_e \geq \omega_{e,\text{rl}} \\ 0 & \text{if } v_w \geq v_{\text{sl}} \\ \alpha_{\text{idle}} & \text{if } \alpha_{\text{idle}} \geq \alpha \\ \alpha & \text{otherwise} \end{cases} \tag{4.6}$$

Figure 4.3 shows a plot of the torque produced by the complete engine model including the engine controller with sane values for the different parameters.

4.3 Engine, CVT, and transmission

The drive line consist of the engine, CVT, transmission, differential, and the rear wheels. A model for the engine was already presented in section 4.2. As the drive line is a series of power transferees, bond graph modelling was chosen as a framework to derive the initial model structure. Figure 4.10 on page 33 shows the developed bond graph model of drive train.

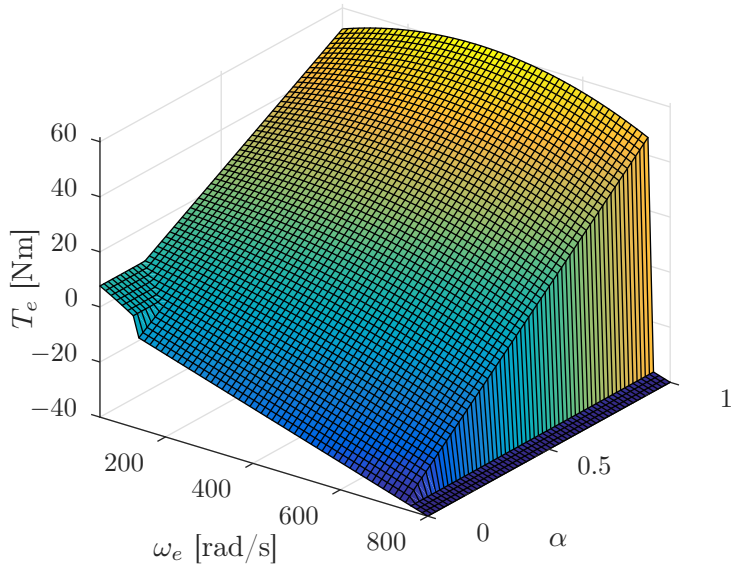


Figure 4.3: Plot of the torque of the engine produced by the engine model with $P_M = 60\,\mathrm{kW}$, $\omega_M = 816\,\mathrm{rad\,s^{-1}}$ (7800 RPM), $b_e = 5\text{E}{-2}\,\mathrm{N\,m\,s\,rad^{-1}} i$, $\omega_{e,\mathrm{idle}} = 131\,\mathrm{rad\,s^{-1}}$ (1250 RPM), and $p_{\mathrm{idle}} = 1\text{E}{-3}$.

4.3.1 Engine bond graph model

As described in section 4.2, the engine is modeled as a rotating mass with moment of inertia I_e influenced by the engine torque modeled by the nonlinear function $T_e(\alpha, \omega_e)$, and the torque applied to engine output shaft, T_l. As the engine is directly connected to the primary pulley of the CVT, the load torque, T_l, is the torque produced at the CVT's primary pulley, T_p. The engine can be described by the bond graph in fig. 4.5 consisting of a modulated source of effort representing T_e, a source of effort representing T_l, and an inertia element representing I_e, all connected to a common flow node.

4.3.2 Transmission and differential bond graph model

As the model only describe the longitudinal dynamics of the vehicle, the two rear wheels are assumed to always rotate at the same rate, and the differential is therefore only modeled as the speed ratio between its input and output shafts, called the final drive ratio. The selectable speed ratio of the transmission is combined with the final drive ratio. This combined ratio can be described by a modulated transformer, **MTF**, with ratio $R_t(g)$, which is a function of the selected gear. At each end of the modulated transformer there is a common flow node modelling the transmission input and output shafts. The transmission input shaft is connected to the secondary pulley of the CVT. The torque from the secondary pulley is modeled by a source of effort. The inertia of the input shaft, including the secondary pulley, is modeled by an inertia, I_t, and the friction of the transmission with a resistance, with coefficient b_t. The output shaft of the differential is connected to the rear axle and the wheels. The rear axle and wheels are modeled by an inertia, I_{rw}, with angular speed ω_{rw}. In addition to the torque from the transmission the rear axle the torque from the brakes, T_{bk}, and from the traction and rolling resistance of the rear wheels, T_{rw}. T_{bk} and T_{rw} are modeled in sections 4.4 and 4.5 and are here only input variables modeled by sources of effort. The bond graph model for the transmission and rear axle is shown in fig. 4.6.

This bond graph model is not very useful as it is because one of the inertia elements has to be given a derivative causality. It is therefore rearranged as shown in fig. 4.7 with the inertias combined using the speed ratio of the transmission. The resistance is also moved to the other common flow bond to eliminate the common flow bond on the input side of the transmission.

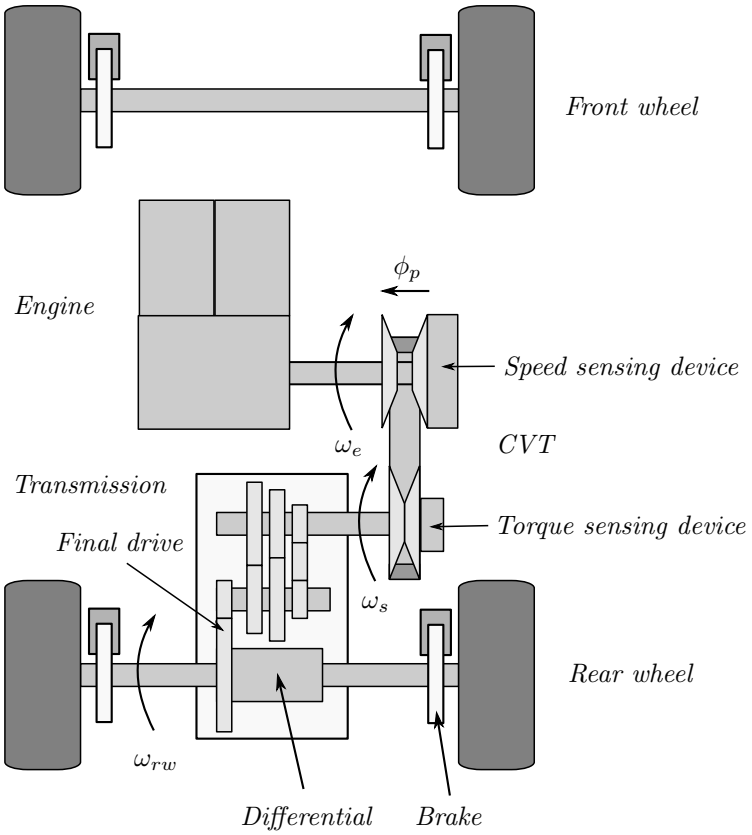


Figure 4.4: Simplified diagram of drive line components.

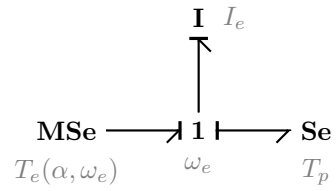


Figure 4.5: Bond graph model of engine.

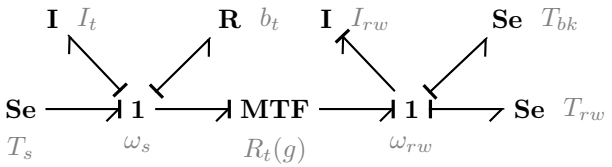


Figure 4.6: Bond graph model of transmission and rear axle.

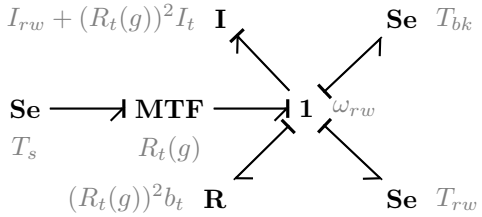


Figure 4.7: Equivalent bond graph model of transmission and rear axle without derivative causality.

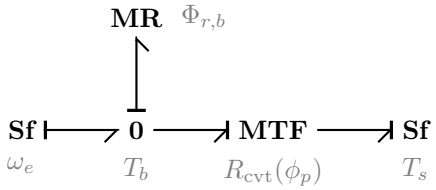


Figure 4.8: Bond graph model of CVT belt

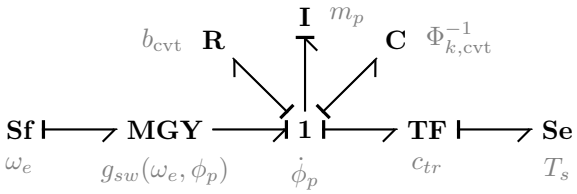


Figure 4.9: Bond graph model of the axial displacement of the pulleys.

4.3.3 CVT bond graph model

There are two different paths for the power to flow through the CVT: along the belt and through the speed sensing device on the primary pulley and the torque sensing device on the secondary pulley resulting in the axial displacement of the pulleys. Power flow along the belt is considered first.

The contact between the belt and the primary is modeled by a common effort node and a resistance governed by the nonlinear modulated resistance $\Phi_{r,b}$. $\Phi_{r,b}$ will be decided later. The primary pulley is modeled as a source of flow connected to the common effort node representing the belt pulley contact. The speed ratio caused by the different radii of the pulleys is modeled by a modulated transformer, with ratio $R_{\text{cvt}}(\phi_p)$, which is a function of the unit less displacement state variable ϕ_p. ϕ_p describes how much the primary pulley is compressed and thereby, assuming the belt does not stretch, the radii of the two pulleys. It is assumed that the belt never slips at the secondary pulley, and it is therefore modeled by a source of flow directly connected to the modulated transformer. This results in the bond graph shown in fig. 4.8

The axial displacement of the two pulleys is assumed to be completely coupled by the belt, they are therefore modeled by a common flow node with connected to a inertia m_p, which is the mass of the movable pulley sheaves. The friction in the system is modeled by a resistance, b_{cvt}. Both the torque sensing device on the primary pulley and the torque sensing device on the secondary pulley are fitted with springs. The springs are directed the same way and can therefore be combined an modeled by a common capacitance element with the nonlinear capacitance $\Phi^{-1}_{k,\text{cvt}}$. It is represented by the inverse because it represents a spring.

The speed sensing device converts the speed of the primary pulley into a force pushing on the movable pulley sheave. This means that it can be modeled by a gyrator. It is therefore modeled by a modulated gyrator, with a gyration relationship $g_{sw}(\phi_p, \omega_e)$, which is a function of the pulley displacement, ϕ_p, and engine speed, ω_e. The engine speed, ω_e, is modeled by a source of flow.

The torque sensing device converts the torque experienced by the secondary pulley, T_s, into a force which compresses the secondary pulley sheaves. It is modeled by a transformer with transformation relationship, c_{tr}, and T_s by a source of effort.

The bond graph for the axial displacement is shown in fig. 4.9.

4.3.4 Complete drive line model

The bond graphs for the individual drive line components, shown in figs. 4.5 and 4.7 to 4.9 can be combined to a complete bond graph model of the drive line, as shown in fig. 4.10. This is done by connecting together their source of efforts and flows with common flow and effort nodes. The resulting model has two input variables, throttle settings, α, and gear,

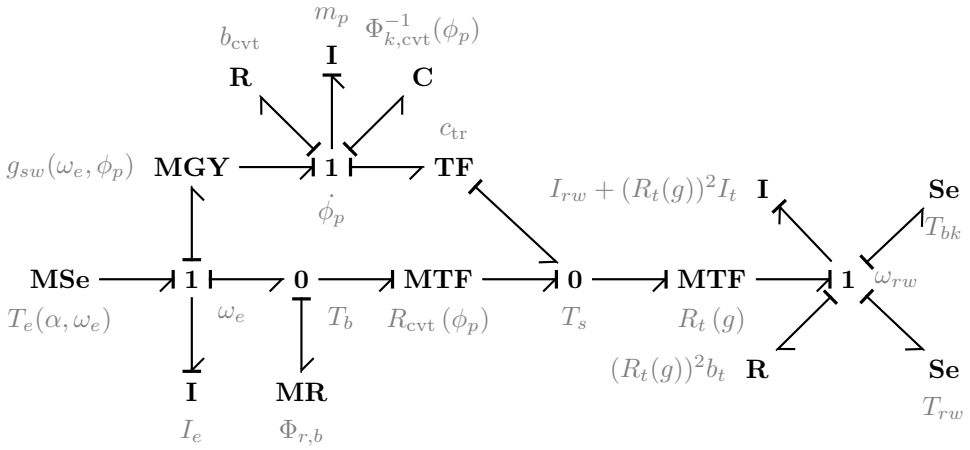


Figure 4.10: Bond graph model of the drive line.

g. It also has two ports: the sources of effort of the brake torque, and rear wheel torque. Alternately sources of efforts can be looked at as two inputs: T_{rw} and T_{bk}, and an output: ω_{rw}.

The inertia and capacitive elements in the model dictates its state variables. All the inertia elements in the model have integral causality, and the flow of the common flow nodes they are connected to therefore becomes state variables: ω_e, $\dot{\phi}_p$, and ω_{rw}. The capacitive element also has integral causality, causing ϕ_p to also become a state variable. By following the causalities and summing together the efforts integrated by the inertias the following set of differential equations can be derived from the bond graph model in fig. 4.10:

$$I_e\dot{\omega_e} = T_e(\alpha, \omega_e) - g_{sw}(\omega_e)\dot{\phi_p} - T_b \tag{4.7}$$

$$m_p\ddot{\phi_p} = g_{sw}(\omega_e)\omega_e - r_{\mathrm{cvt}}\dot{\phi_p} - \Phi_{k,\mathrm{cvt}}(\phi_p) - c_{\mathrm{tr}}R_{\mathrm{cvt}}(\phi_p)T_b \tag{4.8}$$

$$\left(I_{rw} + R_t(G)^2 I_t\right)\dot{\omega_w} = R_t(g)R_{cvt}T_b - (R_t(G))^2 b_t\omega_w - T_{bk} - T_{rw} \tag{4.9}$$

$$T_b = \Phi_{r,b}\left(\omega_e, R_{\mathrm{cvt}}\left(R_t(G)\omega_w - c_{\mathrm{tr}}\dot{\phi_p}\right)\right) \tag{4.10}$$

The derivation of these equations from the bond graph model are shown in appendix B.

A general structure for the model is now found. To complete the model the different functions and values in the differential equations has to be found.

The axial force on the primary pulley produced by the shift weights is governed by the mass of the shift weights, m_{sw}, the pulley displacement dependant radius of the shift weights, $r_{sw}(\phi_p)$, the derivative of the radius in relation to the displacement, $\frac{d\phi_p}{dr_{sw}}$, defined by the profile the shift weights pushes against, which dictates the ratio between the centripetal force acting on the shift weights and the axial force compressing the pulley, and the square of the angular velocity, ω_e^2:

$$F_{sw} = \left.\frac{d\phi_p}{dr_{sw}}\right|_{\phi_p} m_{sw}r_{sw}(\phi_p)\omega_e^2 \tag{4.11}$$

The gyrator relationship can therefore be expressed as:

$$F_{sw} = \left[\left.\frac{d\phi_p}{dr_{sw}}\right|_{\phi_p} m_{sw}r_{sw}(\phi_p)\omega_e\right]\omega_e = g(\omega_e, \phi_p)\omega_e \tag{4.12}$$

By using some of the knowledge of the tuning goals of the mechanism the expression can be simplified. As the torque produced by the engine is nearly constant for all engine speeds, the produced power will therefore be close linear in respect to the engine speed. A clamping force proportional to the square of the engine speed pushing against linear spring would therefore be undesirable as it results in a very narrow usable shifting band. It would therefore seems likely that the shift weight radius to pulley displacement ratio and its derivative is designed

in such a that the force to engine speed ratio is close to linear. The ratio is therefore assumed to be linear and can therefore be expressed by one parameter:

$$F_{sw} = c_{sw}\omega_e \tag{4.13}$$

As the primary pulley also has to reach a minimum angular velocity before it contacts the belt a minimum value for the engine speed for belt engagement, ω_{be}, is introduced. Below this speed, no forces or torques are transfered via the shift weight system. The final expression for the shift weight gyrator relationship, $g_{\mathrm{sw}}(\omega_e)$ is:

$$g_{\mathrm{sw}}(\omega_e) = \begin{cases} c_{\mathrm{sw}} & \text{if } \omega_e > \omega_{\mathrm{be}} \\ 0 & \text{otherwise} \end{cases} \tag{4.14}$$

The springs in the speed and torque sensing devices is modeled by the nonlinear capacitance $\Phi_{k,\mathrm{cvt}}$. It is assumed linear for ϕ_p in the range $[0, 1]$ with the spring coefficient k_{cvt}. Outside this range the spring constant is multiplied by 10 to model end of travel of the movable pulley sheaves. The produced effort by the spring is:

$$\Phi_{k,\mathrm{cvt}}(\phi_p) = \begin{cases} k_{\mathrm{cvt}}\phi_p & \text{if } 0 \leq \phi_p \leq 1 \\ k_{\mathrm{cvt}}\phi_p + 10k_{\mathrm{cvt}}(\phi_p - 1) & \text{if } 1 < \phi_p \\ 10k_{\mathrm{cvt}}\phi_p & \text{if } \phi_p < 0 \end{cases} \tag{4.15}$$

The torque transfered from the primary pulley to the belt, T_b, is modeled by the nonlinear resistance $\Phi_{r,b}$. Following traditional laws of friction it is proportional to the normal force acting on the belt surface (clamping force of primary pulley), the relative speed between the belt and the pulley as well as other factor such as material properties of the belt and the pulley. The friction model used to model the contact is a slip based model. The slip is calculated from the angular velocity of the engine, ω_e, and the angular velocity of the belt at the primary pulley, ω_b, ω_b is found from angular velocity of the wheels, ω_w, the speed contributed by the torque sensing device at the secondary pulley, and the speed ratio of the transmission, $R_t(g)$ and the CVT, $R_{\mathrm{cvt}}(\phi_p)$:

$$\omega_b = R_{\mathrm{cvt}}(\phi_p)\left(R_t(g)\omega_w - c_t r\dot{\phi_p}\right) \tag{4.16}$$

The slip is defined as:

$$s_b = \frac{\omega_e - \omega_b}{\max\left(|\omega_e|, |\omega_b|\right)} \tag{4.17}$$

The clamping force of the primary pulley was already decided to be proportional to the engine speed, but the same expression for the magnitude is not used. Instead it is included in the coefficient of friction of the belt. The model for the produced torque as a function slip and engine speed is a very simplified three parameter model defined by the maximum coefficient of friction, $\mu_{b,\max}$, the slip producing maximum friction $s_{b,\max}$, and the minimum engagement speed for the belt, ω_{be}:

$$T_b(s_b, \omega_e) = \begin{cases} \mu_b(\omega_e - \omega_{be}) \tanh\left(2\frac{s_b}{s_{b,\max}}\right) & \text{if } \omega_e > \omega_{be} \\ 0 & \text{otherwise} \end{cases} \tag{4.18}$$

The speed ratio of the CVT is proportional with the axial displacement of the pulleys, ϕ_p, as the displacement of the pulleys changes the radii of the pulleys. The relationship is assumed to be linear. The speed ratio is constrained between a upper and a lower bound, the highest possible speed ratio at full down-shift, $R_{\mathrm{cvt},D}$, and the lowest at full up-shift, $R_{\mathrm{cvt},U}$. As the pulley displacement is constrained between 0 and 1 the speed ratio of the CVT, $R_{\mathrm{cvt}}(\phi_p)$, can be expressed as

$$R_{\mathrm{cvt}}(\phi_p) = R_{\mathrm{cvt},D} - \left(R_{\mathrm{cvt},D} - R_{\mathrm{cvt},U}\right)\phi_p \tag{4.19}$$

The transmission is modeled as a simple modulated transformer with a fixed gearing ratio for each gear:

$$R_t(G) = \begin{cases} R_H, & G = H \\ R_L, & G = L \\ R_R, & G = R \\ 0, & G = N \end{cases} \tag{4.20}$$

The gearing ratios includes the ratio provided by the final drive gearing of the differential.

T_{rw} is the sum of torque applied to the rear axle by the rear wheels. T_{bk} is the torque applied to the rear axle by the rear brakes. These will be discussed later in sections 4.4 and 4.5 and are only assumed to inputs here inputs here.

The feedback torque on the primary and speed on the secondary pulley from the shifting mechanism is assumed to be small in comparison to the rest of the torques and speeds that they can be ignored and removed from the model. This makes c_{sw}, c_{tr}, k_{cvt}, r_{cvt}, and m_p unit less. m_p thereby becomes meaningless as it only scales the acceleration of the pulley displacement and is therefore removed. To make the parameters less sensitive to changes in other parameters, c_{sw} and c_{tr} are replaced by k_{cvt} divided by two new parameters, u_{cvt} and d_{cvt}.

$$c_{\text{sw}} = \frac{k_{\text{cvt}}}{u_{\text{cvt}}} \tag{4.21}$$

$$c_{\text{tr}} = \frac{k_{\text{cvt}}}{d_{\text{cvt}}} \tag{4.22}$$

The result of this is that u_{cvt} controls the engine speed required to overcome the spring tension, d_{cvt} controls the sensitivity of the torque sensing in comparison to the spring, and k_{cvt} only controls the rate of change of the speed and not the speed ratio itself.

In addition to eqs. (4.15) and (4.18) to (4.20) the model of the engine, CVT and transmission can be summarized with the following set of differential equations:

$$\begin{bmatrix} \dot{\omega_e} \\ \ddot{\phi_p} \\ \dot{\phi_p} \\ \dot{\omega_{rw}} \end{bmatrix} = \begin{bmatrix} \frac{1}{I_e}\left[T_e(\alpha, \omega_e) - T_b\right] \\ \frac{k_{\text{cvt}}}{u_{\text{cvt}}}\omega_e - \frac{k_{\text{cvt}}}{d_{\text{cvt}}} R_{\text{cvt}}(\phi_p) T_b - \Phi_{k,\text{cvt}}(\phi_p) - b_{\text{cvt}}\dot{\phi_p} \\ \dot{\phi_p} \\ \frac{1}{I_{rw} + R_t(g)^2 I_t}\left[R_t(G)\left(R_{\text{cvt}} T_b - b_t R_t(G)\omega_w\right) - T_w\right] \end{bmatrix} \tag{4.23}$$

4.4 Brake system

The brake system is modeled as a torque proportional to a unitless brake "pressure", ρ_{bk}, with the opposite sign of the axle the brakes are applied to.

$$T_{bk}(\omega, \rho_{bk}) = -\operatorname{sign}(\omega) T_{bk,\max} \rho_{bk} \tag{4.24}$$

Due to the magnitude of the brake torque the sign function, is approximated with tanh as this gives better numerical stability near zero angular velocity.

$$T_{bk}(\omega, \rho_{bk}) = -\tanh(\omega) T_{bk,\max} \rho_{bk} \tag{4.25}$$

ρ_{bk} is produced by delaying the brake input, β, with a time constant, τ_{bk}.

$$\dot{\rho_{bk}} = \frac{1}{\tau_{bk}}\left[\beta - \rho_{\text{bk}}\right] \tag{4.26}$$

The delay takes into account both the slow electrical actuator that actuates the brake pedal as well as any delay in the actual system.

The brake torque is distributed to the front and rear axle according to the brake balance parameter, B_{bk}:

$$T_{fw,bk} = B_{bk} T_{bk}(\omega_{fw}, \rho_{bk}) \tag{4.27}$$

$$T_{rw,bk} = (1 - B_{bk}) T_{bk}(\omega_{rw}, \rho_{bk}) \tag{4.28}$$

$$\tag{4.29}$$

4.5 Tire model

As discussed in section 2.6.1 there are numerous tire models described in the literature. The most common is Pacejka's *magic formula* tire model and variations of it, which is an empirical model which relate the coefficient of friction of the tire to the slip. The simplest formulation contains four parameters that has to be determined experimentally. The *magic formula* model is deemed unsuitable because it contain too many parameters to be estimated directly without a proper tire testing rig. Other alternatives are more physically based models such as LuGre brush model, but these also require a large amount of parameters.

One possible simplified model that tries to parametrize a typical slip to friction coefficient curve is presented Rudd in [22], where it was used to estimate the tire properties for an anti skid system for aircrafts during braking. The friction produced by the tire ground contact is only described with two parameters, the maximum coefficient of friction, $\mu_{t,\max}$, and the normalized slip producing the maximum friction, $s_{t,\max}$.

$$\mu(s) = 2\mu_{t,\max}\frac{s}{s_{t,\max}\left(1+\frac{s}{s_{t,\max}}^2\right)} \tag{4.30}$$

Another possible two parameter model is the "tanh" model proposed in [10]:

$$\mu(s) = \mu_{t,\max}\tanh\left(2\frac{s}{s_{t,\max}}\right) \tag{4.31}$$

fig. 4.11 shows a plot comparing these two simplified models to a curve produced by Paicejka's magic tire formula for a tire on fry asphalt. Both models of diverge from the typical curves produced by mode complex models at higher slip values. Rudd's model produces a unrealistically low friction coefficient at high slip, while the tanh model has the same friction for all slip values over the peak. As high slip values only occurs during extreme maneuvers: when locking the wheels during braking, spinning the driving wheels during acceleration, or on surfaces with very little friction and these conditions are to avoided anyway, these simplified models should be sufficient. As the both produce similar result in the typical slip region either could be used for normal driving conditions, but tanh model is chosen for its simpler structure.

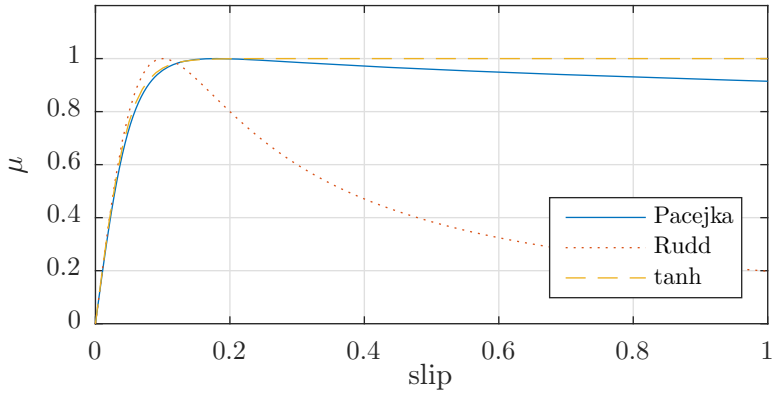


Figure 4.11: Comparison of simplifications of magic formula tire model for the tire dynamics with $\mu_{\max} = 1$ and $s_{\max} = 0.1$ and a curve generated by Pacejka's *magic formula* for a tire on dry asphalt.

The relative slip used in the tire force calculation is defined as:

$$s_w = \frac{\omega_w r_w - v_x}{\max(|\omega_w r_w|, |v_x|)} \tag{4.32}$$

Where v_x is the speed of the vehicle along the longitudinal direction and v_w is the speed of surface of the tire relative to the vehicle. v_w is calculated using the effective radius, r_{eff}:

$$v_w = \omega_w r_{\text{eff}} \tag{4.33}$$

The effective radius is the radius observed by rolling the tire and measuring the distance covered. The directly measured radius of the wheel is generally not the same as the effective radius.

The calculated slip should always be in the range $\langle -1, 1 \rangle$. A slip value of 0 means that there is no sliding between the tire and the ground and a slip value of ± 1 means that the tire is slipping in either direction. As the velocity of the vehicle and tire approaches zero the definition no longer holds as the delimiter becomes zero. This is not a problem when studying a vehicle that always is in motion, but creates problems when investigating stopping and starting. To correct this a minimum value for the delimiter, $v_{\min}$, is added to avoid that the slip becomes singular. In addition to this eq. (4.32) gives slip values with absolute values larger than one when the two velocities have different signs. As this can happen, for example when accelerating when rolling backwards, the slip is constrained to ± 1 by setting the slip equal to the sign of the speed of the wheels when the signs differ. This addition results in the following equation for the slip:

$$s_w = \begin{cases} \operatorname{sign}(v_w) & \text{if } \operatorname{sign}(v_w) \neq \operatorname{sign}(v_x) \\ \frac{\omega_w r_w - v_x}{\max(|\omega_w r_w|, |v_x|, v_{\min})} & \text{otherwise} \end{cases} \tag{4.34}$$

There exists methods for determining a value for $v_{\min}$ that gives the most accurate response of the system, like the method discussed by Lee and Yoo in [17], but this approach introduce additional parameters and computational complexity. A constant value of $v_{\min} = 1$ has shown to give good results and was therefore chosen. This makes the model underestimate the slip when both the speed of the tire contact point and the vehicle is below $1\,\mathrm{m\,s^{-1}}$. This is not regarded as a problem as this gives starts and stops with less oscillations, which is a problem with these kinds of slip based friction models.

The front and rear tires are assumed to have equal coefficient of friction. Including this factor and the normal force acting on the axles the complete expression for the tractive force produced by the tires during static condition is

$$F_{w,x,\mathrm{stat}} = \mu(s_w) F_{w,z} = \mu_{t,\max} \tanh(2 \frac{s_w}{s_{t,\max}}) F_{w,z} \tag{4.35}$$

To account for the dynamic behaviour of the tire, a first order time delay is introduced with time constant τ_t, is introduced:

$$F_{w,x} = \frac{1}{\tau_t} \left(F_{w,x,\mathrm{stat}} - F_{w,x} \right] \tag{4.36}$$

The torque, T_t, applied to the wheel due to the traction force is

$$T_t = r_{\mathrm{dyn}} F_{w,x} \tag{4.37}$$

where r_{dyn} is the dynamic tire radius which is the radius of the tires under load, from axle center to the ground. r_{dyn} generally changes during different load conditions, but this is not modeled, and the dynamic radius is assumed constant.

The rolling resistance is assumed to only have a static component proportional to the normal force applied to the tire and one parameter, b_{rr}. As the rolling resistance is due to deformation of the tire it is inversely proportional to the tire pressure and the width of the tire. As $F = \rho a$, a lower pressure tire has to deflect more to increase the contact patch to the size required to support the applied load. A wider tire has to deform less as the contact patch is wider to begin with. As the tires fitted to the vehicle are wider in the rear but run higher pressure in the front, it is assumed that these effects cancels each other out and that the rolling resistance for a given normal force is equal for the front and the rear tires. The torque produced by the rolling resistance for a given normal load is therefore modeled as:

$$T_{rr} = -\operatorname{sign}(\omega_w) b_{rr} F_{w,z} \tag{4.38}$$

The sign function is again approximated with the tanh function.

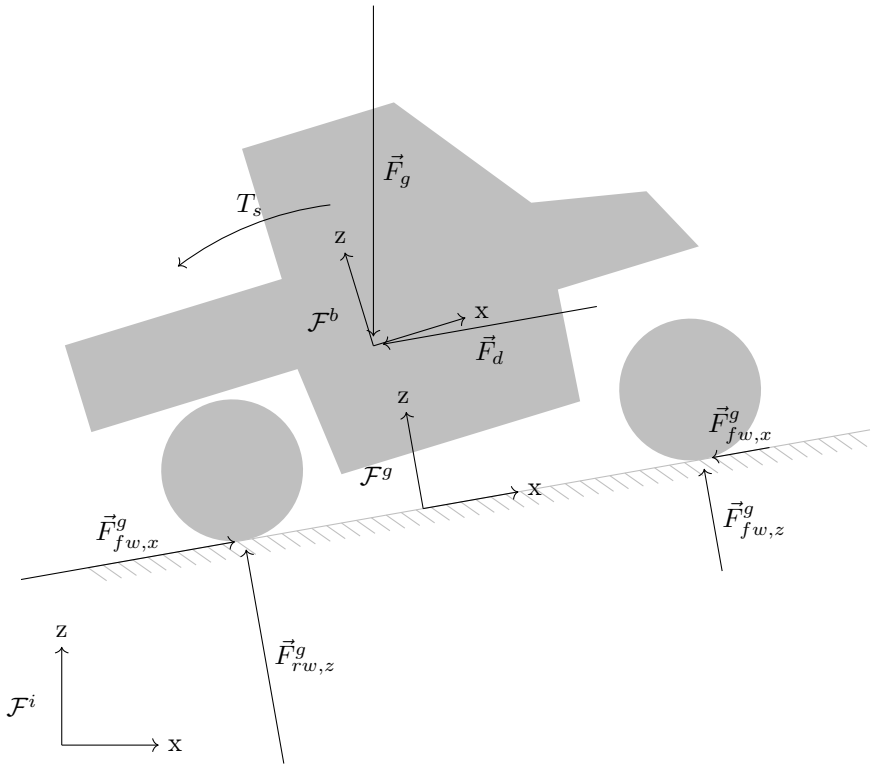


Figure 4.12: Diagram of the frames use to describe the vehicle dynamics and forces acting on the vehicle.

The tire model for a single axle can be summarized with the following set of equations:

$$\begin{bmatrix} \dot{\omega}_w \\ \dot{F}_{w,x} \end{bmatrix} = \begin{bmatrix} \frac{1}{I_w}\left[r_{\text{dyn}}F_{w,x} - \tanh(\omega_w)b_{rr}F_{w,z} + T_a\right] \\ \frac{1}{\tau_t}\left[F_{w,x,\text{stat}} - F_{w,x}\right] \end{bmatrix} \tag{4.39}$$

$$s_w = \frac{\omega_w r_{\text{eff}} - v_x}{\max(|\omega_w r_{\text{eff}}|, |v_x|, v_{\min})} \tag{4.40}$$

$$F_{w,x,\text{stat}} = \mu_{t,\max}\tanh(2\frac{s_w}{s_{t,\max}})F_{w,z} \tag{4.41}$$

Tire normal force, $F_{w,z}$, the vehicle longitudinal velocity, v_x, and the torque applied to the axle, T_a, are input variables. For the front axle T_a is only the brake torque, while for the rear axle it is also the torque applied from the drive line.

4.6 Vehicle dynamic model

The forces acting on the vehicle is gravity, $\vec{F}_g$, drag, $\vec{F}_d$, and the tire forces, $\vec{F}_{fw}$ and $\vec{F}_{rw}$. These forces as well as the reference frames used to describe the model is shown in fig. 4.12. As only the longitudinal dynamics are under consideration OLAV is assumed to only move in two dimensions horizontal and vertical. The axes are labeled x, for horizontal, and z, for vertical. Three frames are used for the model are the inertia frame, $\mathcal{F}^i$, the ground fixed frame, $\mathcal{F}^g$, with x axis parallel to the ground surface as experienced by the wheels and z axis pointing up, and the body frame, $\mathcal{F}^b$, centered in the vehicles center of mass with x pointing forward and z up and intersecting the origin of the ground frame. The wheels are assumed fixed in the ground frame while the rest of the vehicle is assumed fixed in the body frame. The vehicle suspension is assumed fixed vertically, but free to move around the pitch axis. The pivot point of the suspension is assumed to be the origin of the ground frame. This assumed location of the roll center may not coincide with the vehicles true roll center, but simplifies the calculations greatly.

The equations for the linear motion of the body frame in the inertia frame is

$$m_v \dot{\vec{v_b}} = \vec{F_g} + \vec{F_d} + \vec{F_{rw}} + \vec{F_{rw}} \tag{4.42}$$

Which can be decomposed in the axes of the ground frame:

$$m_v(\dot{v^{ig}_{b,x}}) = F^g_{g,x} + F^g_{w,x} + F^g_d \tag{4.43}$$

$$m_v(\dot{v^{ig}_{b,z}}) = F^g_{g,z} + F^g_{fw,z} + F^g_{rw,z} \approx 0 \tag{4.44}$$

As the ground frame is fixed to the ground the vertical forces in $\mathcal{F}^g$ must sum to zero:

$$\sum_z F^g = F^g_{g,z} + F^g_{fw,z} + F^g_{rw,z} = 0 \tag{4.45}$$

and the horizontal forces sum to the tire tractive forces:

$$\sum_x F^g = F^g_{fw,x} + F^g_{rw,x} = 0 \tag{4.46}$$

The equation of motion of the body frame around the pitch axis is:

$$I_{v,\theta}(\dot{\omega^i_b}) = T_s + r^b_{bg} \times \sum \vec{F^g} \tag{4.47}$$

where T_s is the torque produced by the suspension between the ground frame and the body frame and θ^i_b is the orientation of the Body frame in the inertia frame, ω_b is the angular velocity, $\vec{r^b_{bg}}$ is the position of the ground frame in the body frame, and $\sum \vec{F^g}$ are the forces acting on the ground frame. The cross products in eq. (4.47) can be written as

$$I_{v,\theta}(\dot{\omega^i_b}) = T_s + r^b_{bg,x} \sum_z \vec{F^g} + r^b_{bg,z} \sum_x \vec{F^g} \tag{4.48}$$

As the angle between the ground plane is small, $r^b_{bg,x} << r^b_{bg,z}$ and $||r^b_{bg,z}|| \approx ||\vec{r^b_g z}||$, the equation can be simplified to:

$$I_{v,\theta}(\dot{\omega^i_b}) = T_s + \left(F^g_{fw,x} + F^g_{fw,x}\right) ||\vec{r^b_{bg}}|| \tag{4.49}$$

The suspension is modeled as an rotational spring and damper between the ground and the body frame. This can be expressed as:

$$T_s = k_s \left(\theta^i_g + \theta^g_{b,0} - \theta^i_b\right) + b_s \left(\omega^i_g - \omega^i_b\right) \tag{4.50}$$

where k_s and b_s are the spring and damper coefficients, and $\theta^g_{b,0}$ is the orientation of the body frame in the ground frame when no torque is imparted by the spring.

To calculate the traction force from the wheels, $F_{fw,x}$ and $F_{rw,x}$, the normal forces of the front and rear wheels, $F_{fw,z}$ and $F_{rw,z}$ are required. As the wheels are assumed fixed in the ground frame the torque balance around the ground frame must be approximately zero:

$$\vec{F^g_{fw}} \times \vec{r^g_{fc}} + \vec{F^g_{rw}} \times \vec{r^g_{rc}} - T_s \approx 0 \tag{4.51}$$

where $\vec{r^g_{fc}}$ and $\vec{r^g_{fc}}$ are the positions of the front and rear tire contact points in the ground frame. Due to the geometry the cross products can be written as:

$$T_s = F^g_{fw,z} r^g_{fc,x} - F^g_{rw,z} r^g_{rc,x} \tag{4.52}$$

Using the assumption of zero vertical motion in the ground frame, eq. (4.44) can be rearranged to:

$$F^g_{rw,z} = -F^g_{g,z} - F^g_{fw,z} \tag{4.53}$$

which then can be used to eliminate $F^g_{rw,z}$ from eq. (4.52):

$$T_s = F_{fw,z} r^g_{fc,x} - (-F^g_{g,z} - F^g_{fw,z}) r^g_{rc,x} \tag{4.54}$$

By rearranging this equation, $F^g_{fw,z}$ can be found:

$$F^g_{fw,z} = \frac{T_s - F^g_{g,z} r^g_{rc,x}}{r^g_{fc,x} + r^g_{rc,x}} \tag{4.55}$$

Equation (4.53) can then be used to find $F^g_{rw,z}$.

The normal forces applied to the tires can obviously never be negative and the sum of the normal forces can never be larger than the component of gravity parallel to the ground frames z-axis. There therefore has to be set a maximum and minimum value for the suspension torque. This implies that the following two inequalities must be satisfied:

$$F^g_{fw,z} = \frac{T_s - F^g_{g,z} r^g_{rc,x}}{r^g_{fc,x} + r^g_{rc,x}} \geq 0 \tag{4.56}$$

$$F^g_{rw,z} = -F^g_{g,z} - F^g_{fw,z} \geq 0 \tag{4.57}$$

Solving the inequalities in relation to T_s gives the following constraint for T_s:

$$F^g_{g,z} r^g_{rc,x} \leq T_s \leq -F^g_{g,z} r^g_{rc,x} + \tag{4.58}$$

The drag force F_d is assumed oppositely directed and proportional to the squared of the velocity of the body frame in the inertia frame:

$$\vec{F^i_d} = -b_{ar} \frac{\vec{v^i_b}}{||\vec{v^i_b}||} ||\vec{v^i_b}||^2 \tag{4.59}$$

As the velocity of the body frame is parallel with the x axis of the ground frame the drag force only has a component along the ground frames x-axis:

$$F^g_{d,x} = -b_{ar} \operatorname{sign}(v^{ig}_{b,x})(v^{ig}_{b,x})^2 \tag{4.60}$$

As the speed of the body frame is assumed zero along the ground frames z-axis, the position of the vehicle in the inertia frame is described by:

$$\dot{\vec{r}^i_b} = R^i_g \vec{v^{ig}_b} \tag{4.61}$$

where R^i_g is the rotation matrix defining the orientation of the ground frame in the inertia frame. Decomposed in the two axes of the inertia frame the position is defined by:

$$\dot{r}^i_{b,x} = \cos(\theta_g) v^{ig}_{b,x} \tag{4.62}$$

$$\dot{r}^i_{b,z} = \sin(\theta_g) v^{ig}_{b,x} \tag{4.63}$$

The vehicle mass is described with two parameters: $m_{v,e}$, which is the mass of the vehicle including all equipment permanently fixed to the vehicle, and $m_{v,p}$, which is the weight of the payload, generally any occupants in the cabin.

$$m_v = m_{v,e} + m_{v,p} \tag{4.64}$$

$m_{v,p}$ is assumed to be added at the vehicles center of mass as it is roughly centered in the cabin. Variation in mass due to fuel load is not taken into account, as the fuel capacity of 37.8 l is small in comparison to the vehicle's mass which is higher than 1000 kg.

The terrain is parametrized by the orientation of ground frame, $\theta_g(r_x)$, and its derivative in relation to the horizontal displacement, $\frac{d\theta_g}{dr_x}(r_x)$. The angular velocity of the ground frame, ω_g, is then dependant on the position in the terrain and the horizontal speed of the vehicle:

$$\omega_g = \dot{\theta}_g = \frac{d\theta_g}{dt} = \frac{d\theta_g}{dr_x}\frac{dr_x}{dt} = \frac{d\theta_g}{dr_x}\cos(\theta_g) v_x \tag{4.65}$$

When driving in relatively smooth terrain ω_g will be very small and may possibly be ignored.

The model for the vehicle dynamics can be summed up with the following vector differential equation with simplified notation:

$$\begin{bmatrix} \dot{\omega_b} \\ \dot{\theta_b} \\ \dot{v_x} \\ \dot{r_x} \\ \dot{r_z} \end{bmatrix} = \begin{bmatrix} \frac{1}{I_{v,\theta}} \left[T_s(\theta_b, \omega_b, \theta_g, \omega_g) + F_{fw,x}(s_{fw}, F_{fw.z}) + F_{rw,x}(s_{rw}, F_{rw.z}) \right] \\ \omega_b \\ \frac{1}{m_v} \left[\left(F_{fw,x}(s_{fw}, F_{fw.z}) + F_{rw,x}(s_{rw}, F_{rw,z}) \right) r_{cg} + F_{g,x} + F_d \right] \\ \cos(\theta_g) v_x \\ \sin(\theta_g) v_x \end{bmatrix} \tag{4.66}$$

$$m_v = m_{v,e} + m_{v,p} \tag{4.67}$$

$$F_{g,x} = -\sin(\theta_g) m_v g \tag{4.68}$$

$$F_{g,z} = -\cos(\theta_g) m_v g \tag{4.69}$$

$$F_d = -b_{ar} \operatorname{sign}(v_x) (v_x)^2 \tag{4.70}$$

$$T_s = k_s \left(\theta_g + \theta_{v,0} - \theta_v \right) + b_s \left(\omega_g - \omega_b \right) \tag{4.71}$$

$$T_s \in [F_{g,z} r_r, -F_{g,z} r_f] \tag{4.72}$$

$$F_{fw,z} = \frac{T_s - F_{g,z} r_r}{r_f + r_r} \tag{4.73}$$

$$F_{rw,z} = -F_{g,z} - F_{fw,z} \tag{4.74}$$

The frame postscript is dropped from forces, the position, and velocity. As only the length of radius vector matter, $r^g_{fc,x}$, $r^g_{rc,x}$, and r^b_{bg} are simplified to r_f, r_r and r_{cg}. The orientation of the body frame, θ_b, is changed to θ_v, where "v" is short for vehicle. The static offset between ground and vehicle orientation is changed to $\theta_{v,0}$. The wheel traction forces $F_{fw,x}$ and $F_{rw,x}$ are calculated by the tire model presented in section 4.5.

4.7 Complete vehicle model

The complete vehicle model has the following state vector

$$\underline{x} = \begin{bmatrix} \omega_e & \dot{\phi}_p & \phi_p & \rho_b & \omega_{fw} & \omega_{rw} & F_{fw,x} & F_{rw,x} & \omega_v & \theta_v & v_x & r_x & r_z \end{bmatrix}^T \tag{4.75}$$

The states are summarized in table 4.1. The inputs of the model is throttle, α, brake, β, gear, G, the inclination of the terrain, θ_g, and its derivative in relation to the horizontal position $\frac{d\theta_g}{dr_x}$.

$$\underline{u} = \begin{bmatrix} \alpha & \beta & G & \theta_g & \frac{d\theta_g}{dr_x} \end{bmatrix} \tag{4.76}$$

The models presented in sections 4.2 to 4.6 can be combined to a complete state space model of the vehicles longitudinal dynamics:

$$\begin{aligned} \dot{\underline{x}}(t) &= \underline{f}(\underline{x}(t), \underline{u}(t), \underline{\gamma}) \\ &= \begin{bmatrix} \frac{1}{I_e} \left[T_e \left(\alpha, \omega_e \right) - T_b \right] \\ \frac{k_{\text{cvt}}}{u_{\text{cvt}}} \omega_e - \frac{k_{\text{cvt}}}{d_c v t} \left(R_{\text{cvt}} T_b \right) - b_{\text{cvt}} \dot{\phi}_p - \Phi_{k,\text{cvt}}(\phi_p) \\ \dot{\phi}_p \\ \frac{1}{\tau_b} \left[\beta - \rho_b \right] \\ \frac{1}{I_{fw}} \left[B_{bk} T_{bk}(\omega_{fw}, \rho_{bk}) + T_{rr}(\omega_{fw}, F_{fw,z}) - r_{\text{dyn}} F_{fw,x} \right] \\ \frac{1}{I_{ra}} \left[R_{\text{cvt}} R_t(G) T_b + (1 - B_{bk}) T_{bk}(\omega_{rw}, \rho_{bk}) + T_{rr}(\omega_{rw}, F_{rw,z}) - r_{dyn} F_{rw,x} \right] \\ \frac{1}{\tau_t} \left[F_{fw,x,\text{stat}} - F_{rw,x} \right] \\ \frac{1}{\tau_t} \left[F_{rw,x,\text{stat}} - F_{fw,x} \right] \\ \frac{1}{I_{v,\theta}} \left[(F_{fw,x} + F_{rw,z}) r_{cg} + k_s \left(\theta_g + \theta_{v,0} - \theta_v \right) + b_s \left(\omega_g - \omega_v \right) \right] \\ \omega_v \\ \frac{1}{m_v} \left[F_{G,x} + F_d + F_{fw,x} + F_{rw,x} \right] \\ v_x \cos(\theta) \\ v_x \sin(\theta) \end{bmatrix} \end{aligned} \tag{4.77}$$

Symbol	Description	unit
ω_e	Angular velocity of engine.	$\mathrm{rad\,s^{-1}}$
$\dot{\phi}_p$	Displacement speed of CVT pulleys.	
ϕ_p	Displacement of CVT pulleys. Proportional to the speed ratio of the CVT.	
ρ_b	Delayed brake input.	
ω_{fw}	Angular velocity of front wheels.	$\mathrm{rad\,s^{-1}}$
ω_{rw}	Angular velocity of rear wheels.	$\mathrm{rad\,s^{-1}}$
$F_{fw,x}$	Tractive force of front wheels.	N
$F_{rw,x}$	Tractive force of rear wheels.	N
ω_v	Angular velocity of vehicle body around pitch axis.	$\mathrm{rad\,s^{-1}}$
θ_v	Pitch angle of vehicle in the inertia frame.	rad
v_x	Speed of the vehicle along the ground.	$\mathrm{m\,s^{-1}}$
r_x	Horizontal position of the vehicle in the inertia frame.	m
r_z	Vertical position of the vehicle in the inertia frame.	m

Table 4.1: The model states and their units.

The complete vehicle model contains 42 parameters. The parameters are summarized in table 4.2. Some of these are known, while others can be estimate with reasonable accuracy from the vehicle specifications, while others have to be estimated experimentally.

4.8 Measurement model

The measured states of the model is the engine speed, ω_e, the speed of the rear wheels, ω_{rw}, the speed in the body frame, v_x, the altitude, r_z, and the pitch of the vehicle body, θ_v:

$$\underline{z} = \begin{bmatrix} \omega_w & \omega_{rw} & v_x & r_z & \theta_v \end{bmatrix}^T \tag{4.78}$$

These measurements are provided by the OLAVs instrumentation and navigation system. The sensors are sampled in discrete time and are assumed to have additive noise. This results in the following measurement model:

$$\underline{z}_k = h(\underline{x}(t_k)) + \underline{w}_k = \begin{bmatrix} \omega_e(t_k) \\ \omega_{rw}(t_k) \\ v_x(t_k) \\ r_z(t_k) \\ \theta_v(t_k) \end{bmatrix} + \underline{w}_k \tag{4.79}$$

The measurement model may possibly be an over simplification, particularly for the altitude measured by the navigation system, as it has a bias which drifts slightly over time and the noise is therefore not truly zero mean Gaussian noise. This is compensated for by increasing the covariance of the noise slightly. The measurement noise covariance this consideration as well as the approximate magnitude observed in the measured data, as well as the standard deviation reported by the navigation system:

$$R = \mathrm{diag}\begin{bmatrix} 50 & 5\text{E}{-2} & 1\text{E}{-3} & 1\text{E}{-1} & 1\text{E}{-5} \end{bmatrix} \tag{4.80}$$

The horizontal distance travelled is not included in the measurements model. This is partly because it would complicate the measurement model greatly as the true path of the vehicle is not a straight line like assumed in the model. The position estimate from the navigation system also has the same problem as the altitude, it drifts. Considering the time span of the measurement series and the assumed accuracy of the velocity estimate provided by the navigation system the integration of the velocity should be sufficient.

Parameter	Description
$m_{v,e}$	Total mass of vehicle without any occupants or additional equipment.
$m_{v,p}$	Payload mass. Includes personnel and none standard equipment.
$I_{v,\theta}$	Moment of inertia of the vehicle around the pitch axis
k_s	Spring coefficient of suspension.
b_s	Damper coefficient of suspension.
$\theta_{v,0}$	Pitch of vehicle relative to ground at steady state.
r_f	Distance along longitudinal axis from center of gravity to rear wheel contact patch.
r_r	Distance along longitudinal axis from center of gravity to front wheel contact patch.
r_{cg}	Vertical distance from ground to center of gravity.
b_{ar}	Drag coefficient.
I_e	Moment of inertia of engine including the primary cvt pulley.
$P_{e,\max}$	Peak power of engine at full throttle.
b_e	Viscous friction coefficient of engine.
$\omega_{P_{e,\max}}$	Angular velocity of engine at peak power.
$\omega_{e,\text{idle}}$	Engine idle speed.
$\omega_{e,\max}$	Angular velocity for rev limiter.
$p_{e,\text{idle}}$	Proportional gain of idle controller.
$v_{\max}$	Speed limiter set point.
ω_{be}	Engine speed needed to engage belt.
u_{cvt}	Coefficient controlling the relationship between primary pulley angular velocity and shifting speed.
d_{cvt}	Coefficient controlling the relationship between torque on the secondary pulley and shifting speed.
k_{cvt}	Shifting spring coefficient.
b_{cvt}	Viscous friction coefficient of shifting.
$R_{\text{cvt},U}$	Speed ratio of CVT at full up-shift.
$R_{\text{cvt},D}$	Speed ratio of CVT at full down-shift.
$\mu_{b,\max}$	Maximum friction between belt and the primary pulley.
$s_{b,\max}$	Slip between belt and primary pulley at maximum friction.
I_t	Moment of inertia of the gear box including secondary pulley.
b_t	Transmission friction.
R_H	Speed ratio of transmission in high range.
R_L	Speed ratio of transmission in low range.
R_R	Speed ratio of transmission in reverse.
I_{fw}	Moment of inertia of front wheels, drive shafts, brake disks, and differential.
I_{rw}	Moment of inertia of rear wheels, drive shafts, brake disks, and differential.
b_{rr}	rolling resistance coefficient.
$\mu_{t,\max}$	Maximum friction produced by the tires.
$s_{\mu_{t,\max}}$	Tire slip producing maximum friction.
τ_t	Time constant for dynamic tire forces.
r_{eff}	Effective tire radius.
r_{dyn}	Static tire radius.
$T_{bk,\max}$	Maximum braking torque.
τ_{bk}	Time constant for brake system.
B_{bk}	Front to rear brake torque distribution.

Table 4.2: List of parameters in the complete vehicle model.

4.9 Simulation of the model

Figures 4.13 and 4.14 on the facing page and on page 46 shows simulations of two scenarios: coasting down an incline in neutral and driving up an incline with varying throttle and brake setting. The model was simulated using MATLAB's *ODE45* differential equation solver. The parameters used for the simulation are shown in table 4.3. Both scenarios gives behaviour that fits well with the observed and expected vehicle behaviour. The chosen brake torque, $T_{bk} = 5000\,\mathrm{N\,m}$, is probably set too high as front wheels lock up immediately during heavy braking.

The inclination for the coast down simulation was:

$$\theta_g(r_x) = -0.15e^{\frac{-r_x}{200}} \tag{4.81}$$

giving the following expressions for $r_z(r_x)$ and $\frac{d\theta_g(r_x)}{dr_x}$:

$$\begin{aligned} r_z(r_x) &= \int \sin\left(-0.15e^{\frac{-r_x}{200}}\right) dr_x \\ &= 200\,\mathrm{sinint}\left(0.15e^{\frac{-r_x}{200}}\right) \end{aligned} \tag{4.82}$$

$$\frac{d\theta_g(r_x)}{dr_x} = -\frac{0.15}{200}e^{\frac{-r_x}{200}} \tag{4.83}$$

The inclination for the driving up hill simulation was:

$$\theta_g(r_x) = 0.15e^{\frac{r_x-200}{200}} \tag{4.84}$$

giving the following expressions for r_z and $\frac{d\theta_g}{dr_x}$:

$$\begin{aligned} r_z(r_x) &= \int \sin\left(-0.15e^{\frac{r_x-200}{200}}\right) dr_x \\ &= 200\,\mathrm{sinint}\left(0.15e^{\frac{r_x-200}{200}}\right) \end{aligned} \tag{4.85}$$

$$\frac{d\theta_g(r_x)}{dr_x} = \frac{0.15}{200}e^{\frac{r_x-200}{200}} \tag{4.86}$$

Figures 4.15 and 4.16 on page 47 and on page 48 shows the simulations shown in figs. 4.13 and 4.14 on the facing page and on page 46 sampled with the measurement model including additive Gaussian noise.

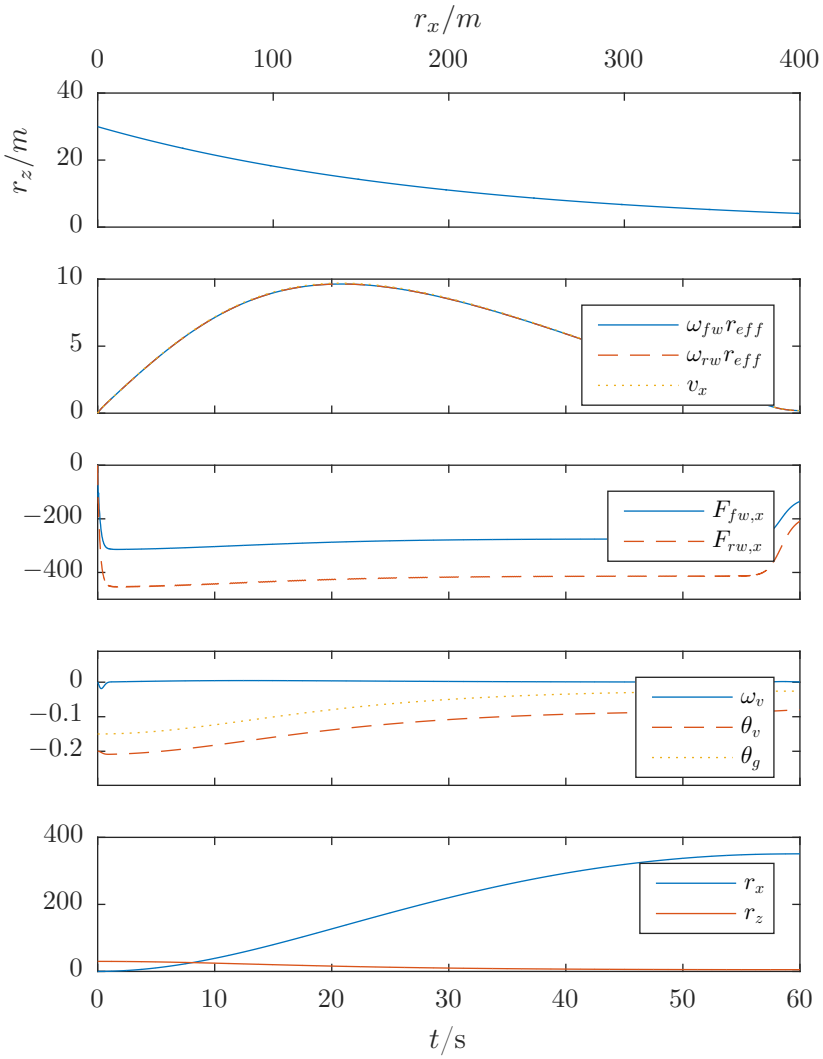


Figure 4.13: Simulation of vehicle coasting down an incline in neutral. First subplot shows the terrain. Model parameters are shown in table 4.3 on page 47 and the deterministic terrain model in eq. (4.81) on page 44. States not relevant when coasting in neutral are omitted.

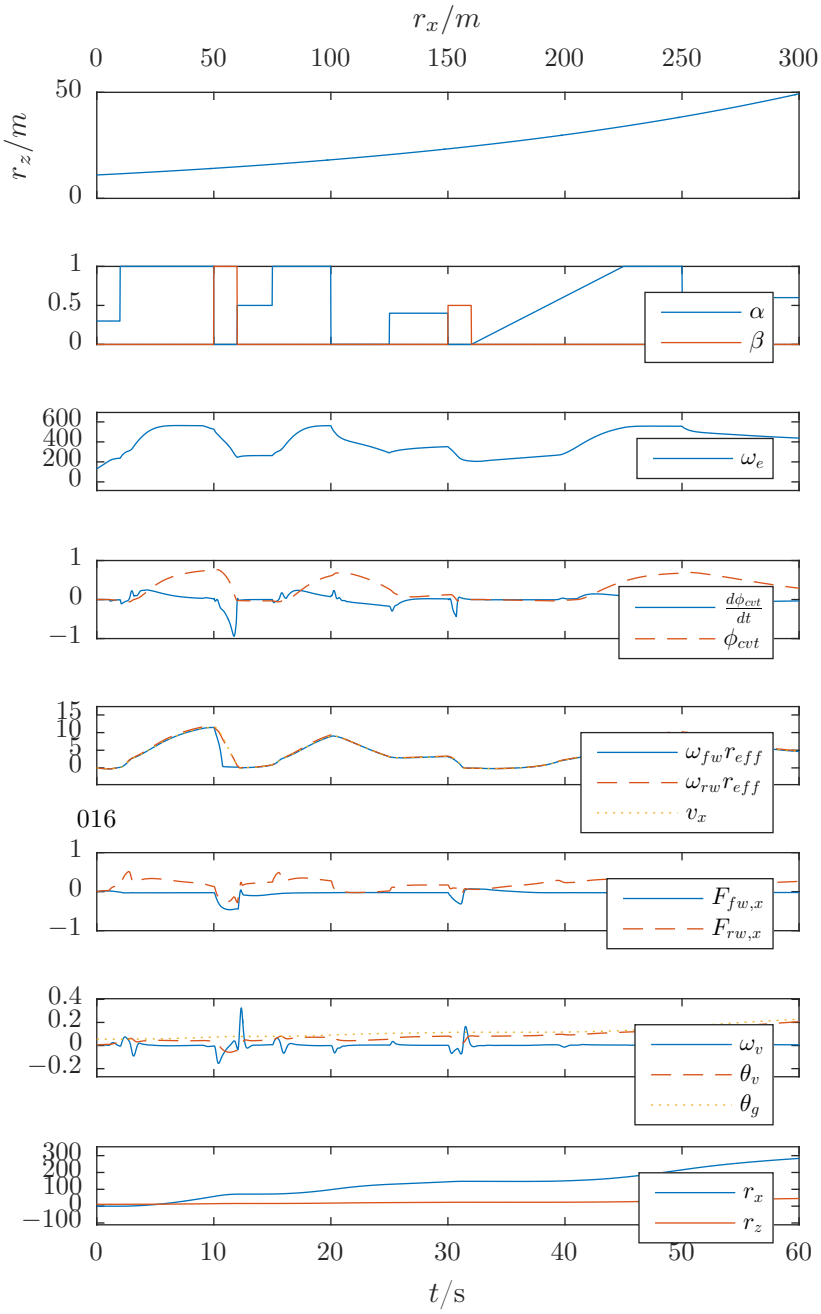


Figure 4.14: Simulation of vehicle driving up an incline with different throttle and brake inputs with transmission in high range. First subplot shows the terrain. Model parameters are shown in table 4.3 on the facing page and the deterministic terrain model in eq. (4.84) on page 44. The delayed brake signal, ρ_{bk} is omitted from the plot.

Parameter	Value	Unit	Parameter	Value	Unit
$m_{v,e}$	1040	kg	b_{cvt}	50	
$m_{v,p}$	160	kg	$R_{\text{cvt},U}$	0.76	
$I_{v,\theta}$	1093	$\text{kg}\,\text{m}^2$	$R_{\text{cvt},D}$	3.83	
k_s	50	$\text{kN}\,\text{rad}^{-1}$	$\mu_{b,\max}$	0.50	
b_s	10	$\text{kN}\,\text{s}\,\text{rad}^{-1}$	$s_{b,\max}$	0.20	
$\theta_{v,0}$	−0.05	rad	I_t	0.025	
r_f	1.26	m	b_t	0	
r_r	0.79	m	R_L	24.59	
r_{cg}	0.60	m	R_H	10.4	
b_{ar}	2.0	$\text{N}\,\text{s}^2\,\text{m}^{-1}$	R_R	−22.92	
I_e	0.2	$\text{kg}\,\text{m}^2$	I_{fw}	4.2	$\text{kg}\,\text{m}^2$
$P_{e,max}$	33.6	kW	I_{rw}	4.7	$\text{kg}\,\text{m}^2$
$\omega_{P_{e,max}}$	733	$\text{rad}\,\text{s}^{-1}$	b_{rr}	0.02	
b_e	0.01	$\text{N}\,\text{m}\,\text{s}\,\text{rad}^{-1}$	$\mu_{t,max}$	0.7	
$\omega_{e,\max}$	759	$\text{rad}\,\text{s}^{-1}$	$s_{\mu_{t,max}}$	0.15	
$\omega_{e,\text{idle}}$	132	$\text{rad}\,\text{s}^{-1}$	τ_t	0.2	s
$p_{e,\text{idle}}$	0.001		r_{eff}	0.35	m
$v_{\max}$	11.2	$\text{m}\,\text{s}^{-1}$	r_{dyn}	0.33	m
$\omega_{\text{engagement}}$	209	$\text{rad}\,\text{s}^{-1}$	$T_{bk,\max}$	5	$\text{kN}\,\text{m}$
u_{cvt}	300	$\text{rad}\,\text{s}^{-1}$	τ_{bk}	0.5	s
d_{cvt}	200	$\text{N}\,\text{m}$	B_{bk}	0.6	
k_{cvt}	100				

Table 4.3: Model parameters used for simulation.

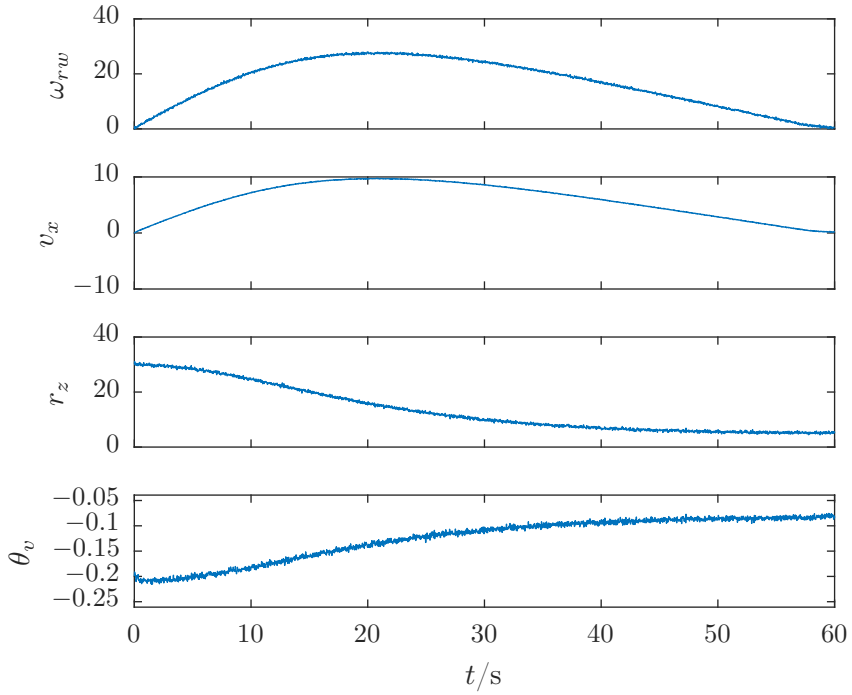


Figure 4.15: Simulated measurements from simulation of vehicle coasting down an incline in neutral.

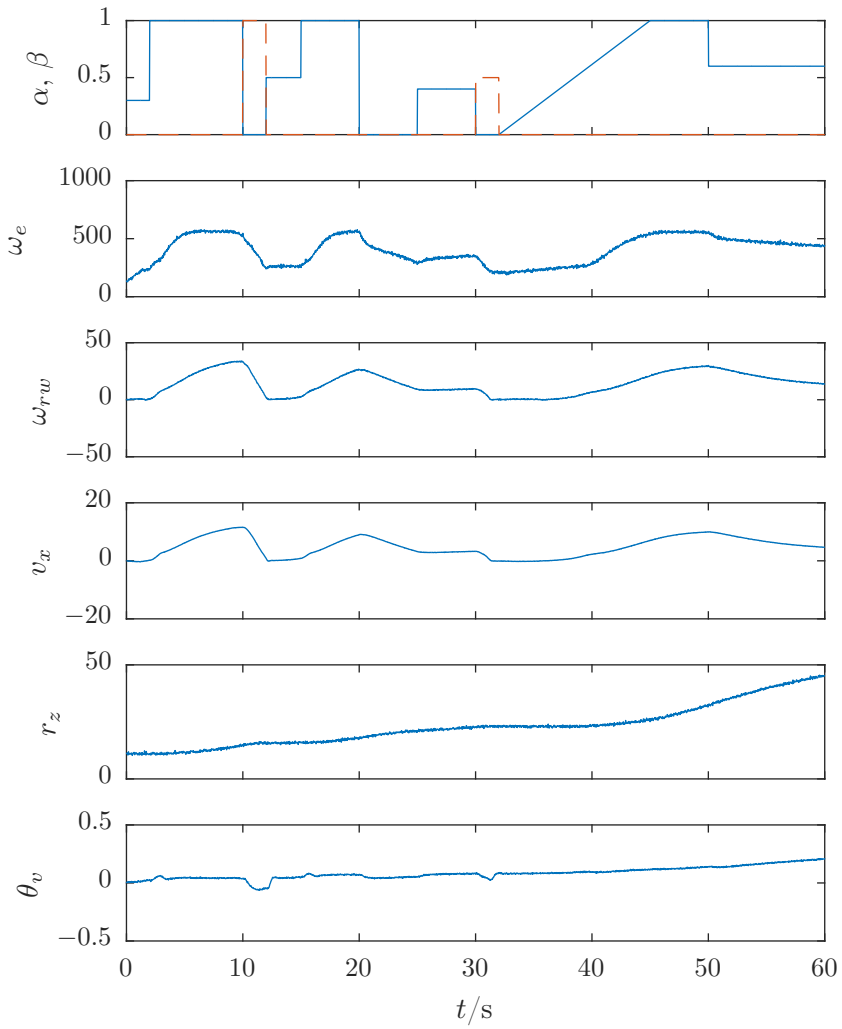


Figure 4.16: Simulated measurements from simulation of vehicle driving up an incline with different throttle and brake inputs with transmission in high range.

Chapter 5

Parameter estimation algorithm

In this chapter the parameter estimation algorithm is presented. This includes the filter models used for state estimation, the criterion function, and the optimization strategy. The MATLAB is not presented in detail, but the source code is presented in appendix: A.

Due to the fact that the inclination of the ground is neither known a priori or measured it is necessary to use a likelihood and state estimation based parameter estimation scheme. Due to the number of parameters in the model, a two stage parameter estimation algorithm using measurements from two different experiments is proposed. It is assumed that some of the parameters can be estimated accurately from observations of the vehicle while it is coasting in neutral down a hill. While the rest can be estimated by driving the vehicle using the throttle and brake actuators.

5.1 Coast down experiment filter model

As coasting in neutral removes any influence from the engine, drive line, and brakes, the states modeling theses are removed from the model. As the only forces on the tires are the tractive forces and the rolling resistance the produced slip is assumed to be negligible and the wheel speed and vehicle velocity is directly coupled through the effective tire radius and the tractive force is assumed equal to the force generated by the rolling resistance torque. This entails a significant reduction in tractive force which significantly reduces the dynamics experienced by the suspension. The suspension model is therefore reduced to only the static offset between the ground and vehicle pitch. As the terrain profile, parametrized by the ground inclination, θ_g and its derivative is unknown, it is added as an additional state to the model. The inclination of the ground is modeled as integrated zero mean Gaussian noise scaled by the absolute value of the vehicles velocity. This results in the following expression for the inclination model:

$$\omega_g = \dot{\theta_g} = (|v_x| + 0.1)v_{\theta_g} \tag{5.1}$$

where v_{θ_g} is the zero mean Gaussian noise process. 0.1 is added to the absolute value of the vehicle velocity so that there is still some uncertainty when the vehicle is not moving. This results in the following state vector:

$$\underline{x} = \begin{bmatrix} v_x & r_x & r_z & \theta_g \end{bmatrix}^T \tag{5.2}$$

and state space model:

$$\dot{\underline{x}} = \underline{f}_{cd}(\underline{x}(t), \underline{v}(t), \underline{\gamma}_{cd}) = \begin{bmatrix} 1/m\left[F_{g,x} + F_d + F_{rr}\right] + v_1 \\ v_x \cos(\theta_g) + v_2 \\ v_x \sin(\theta_g) + v_3 \\ (|v_x| + 0.1)v_4 \end{bmatrix} \tag{5.3}$$

$$m = m_v + \frac{1}{(r_{\text{eff}})^2} I_w \tag{5.4}$$

$$F_{g,x} = -m_v g \cos(\theta_g) \tag{5.5}$$

$$F_{g,z} = -m_v g \sin(\theta_g) \tag{5.6}$$

$$F_d = \text{sign}(v_x) b_{ar} v_x^2 \tag{5.7}$$

$$F_{rr} = r_{\text{dyn}} \,\text{sign}(v_x) b_{rr} F_{g,z} \tag{5.8}$$

$$\underline{v}_{cd} = \begin{bmatrix} v_1 & \dots & v_4 \end{bmatrix}^T \tag{5.9}$$

The continuous model is discretized by integrating from t_k to t_{k+1} using fourth order Runge Kutta with a time step $dt = 1/120\,\text{s}$. The noise is assumed to have a constant value during the prediction step, given the following discrete model:

$$\underline{x}_{k+1} = \underline{f}'_{cd}(\underline{x}_k, \underline{v}_k, \underline{\gamma}_{cd}) = \text{RK4}_{\underline{x}(t_k)}^{\underline{x}(t_{k+1})}\left(\underline{f}_{cd}(\underline{x}(t), \underline{v}_k, \underline{\gamma}_{cd})\right), \quad \underline{x}_k = \underline{x}(t_k) \tag{5.10}$$

The measured states is the vehicle speed, both through the navigation system as well as the wheel speed sensor, the altitude, and the vehicle pitch, giving the following measurement vector:

$$\underline{z} = \begin{bmatrix} \omega_w & v_x & r_z & \theta_v \end{bmatrix}^T \tag{5.11}$$

and measurement model:

$$\underline{z}_k = h_{cd}(\underline{x}(t_k), \underline{w}_k, \underline{\gamma}_{cd}) = \left.\begin{bmatrix} v_x/r_{\text{eff}} \\ v_x \\ r_z \\ \theta_v + \theta_{v,0} \end{bmatrix}\right|_{t=t_k} + \underline{w}_{cd,k} \tag{5.12}$$

The parameters included in the coast down model is the effective tire radius, r_{eff}, rolling resistance coefficient, b_{rr}, drag coefficient, b_{ar}, moment of inertia of the wheels, I_{fw} and I_{rw}, the mass of the vehicle and payload, and the static offset between the ground and the vehicle pitch, $\theta_{v,0}$. r_{eff}, b_{rr}, b_{ar}, and $\theta_{v,0}$ are the parameters which are assumed to be possible to estimate with good accuracy with this model. These parameters are organized in the parameter vector $\underline{\gamma}_{cd}$:

$$\underline{\gamma} = \begin{bmatrix} r_{\text{eff}} & b_{rr} & b_{ar} & \theta_{v,0} \end{bmatrix}^T \tag{5.13}$$

The process noise, $\underline{v}_{cd}$, and measurement, $\underline{w}_{cd}$, vectors are assumed to be zero mean Gaussian noise, with covariance matrix Q_{cd} and R_{cd} respectively. Q_{cd} is set to:

$$Q_{cd} = \text{diag}\left\{1\text{E}{-3} \quad 1\text{E}{-3} \quad 1\text{E}{-3} \quad 1\text{E}{-5}\right\} \tag{5.14}$$

As the only state which is actually assumed to be stochastic is the ground inclinations, θ_g, the covariance of the other states are set rather arbitrarily. The covariance of the θ_g was set to provide good tracking of a moderately uneven terrain.

As the suspension dynamics are ignore, the noise of the pitch measurement is increased by two orders of magnitude over the original measurement model to compensate for the inaccuracy of the model. This results in the following measurement noise covariance matrix:

$$R_{cd} = \text{diag}\left\{5\text{E}{-2} \quad 1\text{E}{-3} \quad 1\text{E}{-1} \quad 1\text{E}{-3}\right\} \tag{5.15}$$

5.2 Full model filter model

The filter model for the complete vehicle longitudinal dynamics uses the complete model from chapter 4, but extended with the same terrain model from the coast down filter model. This gives the following state vector:

$$\underline{x} = \begin{bmatrix} \omega_e & \dot{\phi}_p & \phi_p & \rho_b & \omega_{fw} & \omega_{rw} & F_{fw,x} & \dots \\ & \dots & F_{rw,x} & \omega_v & \theta_v & v_x & r_x & r_z & \theta_g \end{bmatrix}^T \tag{5.16}$$

The terrain inputs are removed from the input vector as they are now model by the stochastic inclination model. This reduces the input vector to:

$$\underline{u} = \begin{bmatrix} \alpha & \beta & g \end{bmatrix} \tag{5.17}$$

The model has the structure:

$$\dot{\underline{x}} = f_{fm}(\underline{x}(t), \underline{u}(t), \underline{v}(t), \underline{\gamma}, \hat{\underline{\gamma}}_{cd}) = \begin{bmatrix} & & v_1(t) \\ f_m(\underline{x}_m(t), \underline{u}_m(t), \underline{\gamma}, \hat{\underline{\gamma}}_{cd}) + & & \vdots \\ & & v_{13}(t) \\ & (|v_x| + 0.1)v_{14}(t) & \end{bmatrix} \tag{5.18}$$

where f_m is the vehicle model, $\underline{x}_m$ is the unextended state, and $\underline{u}_m$ is the original input vector including the terrain from the filter model. The filter model is a function of both the parameters found by the coast down model,$\underline{\gamma}_{cd}$, as well as the other model parameters, $\underline{\gamma}$.

This continuous model is discretized by integrating from t_k to t_{k+1} using fourth order Runge Kutta with time step $dt = 1/200\,\text{s}$. The time step is a compromise between accuracy and reasonable computation time. The noise and the input variables is assumed to have a constant value during the prediction step. This results in the following structure for the discretized model:

$$\begin{aligned} \underline{x}_{k+1} &= \underline{f}'_{fm}(\underline{x}_k, \underline{u}_k, \underline{v}_k, \underline{\gamma}, \underline{\gamma}_{cd}) \\ &= \text{RK4}_{\underline{x}(t_k)}^{\underline{x}(t_{k+1})} \left(\underline{f}_{fm}(\underline{x}(t), \underline{u}_k, \underline{v}_k, \underline{\gamma}, \underline{\gamma}_{cd}) \right), \quad \underline{x}_k = \underline{x}(t_k) \end{aligned} \tag{5.19}$$

The measurement vector is the same as the originally proposed model:

$$\underline{z} = \begin{bmatrix} \omega_e & \omega_{rw} & v_x & r_z & \theta_v \end{bmatrix}^T \tag{5.20}$$

and with the same measurement model:

$$\underline{z}_k = h(\underline{x}(t_k)) = \left. \begin{bmatrix} \omega_e \\ \omega_w \\ v_x \\ r_z \\ \theta_v \end{bmatrix} \right|_{t=t_k} + \underline{w}_k \tag{5.21}$$

Process noise covariance matrix was set to:

$$Q_{fm} = \text{diag} \begin{Bmatrix} 1\text{E}2 & 1 & 1\text{E}{-2} & 1\text{E}{-2} & 1\text{E}{-2} & 1\text{E}{-2} & 1\text{E}2 & \dots \\ \dots & 1\text{E}2 & 1\text{E}{-2} & 1\text{E}{-4} & 1 & 1\text{E}{-5} & 1\text{E}{-5} & 1\text{E}{-4} \end{Bmatrix} \tag{5.22}$$

The covariance was chosen in an effort to account for the assumed inaccuracy of the model, where the engine speed and traction force are assumed to be the most inaccurate. Measurement noise was set to the same as for the original model in chapter 4:

$$R_{fm} = \text{diag} \begin{Bmatrix} 5\text{E}1 & 5\text{E}{-2} & 1\text{E}{-3} & 1\text{E}{-1} & 1\text{E}{-5} \end{Bmatrix} \tag{5.23}$$

As the full model contains all the model parameter theoretically all of the parameters can be estimated. The exact parameters that will be estimated will be decided later.

5.3 Negative log likelihood criterion

As both of the filter models are nonlinear, a UKF will be used for state estimation and for calculating the log likelihood from measurements of the vehicle. The UKF was chosen over the EKF both because it generally performs better and because it does not require the computation of the models Jacobian, which for the full model would be a fairly difficult. As described the continues time models are integrated using fourth order Runge Kutta integration so that the regular formulation of the UKF with discrete time models can be used. The implementation of the UKF is based on the algorithm presented in section 2.2 although the state is not augmented with the measurement noise as it is additive for both of the filter models.

As it is possible for some parameter combinations to cause the UKF to diverge, resulting in a non positive definite covariance, which again causes MATLAB to throw an error when trying to take the matrix square root, the criterion function is set up too catch the error and set the likelihood to $-\infty$ instead of crashing the program. As it may be desirable to use a couple of different measurement series instead of one long measurement series for the parameter estimation, the criterion function is set up to take multiple measurement series and sum the log likelihood from these together. This results in the following criterion:

$$\mathcal{J}(\underline{\gamma}) = \begin{cases} \infty & \text{if UKF throws an error} \\ \sum_{i=1}^{N_{ds}} -\xi_{\text{UKF}}(\mathcal{Z}_N^i, \underline{\gamma}) & \text{otherwise} \end{cases} \tag{5.24}$$

5.4 Optimization strategy

The parameter optimization starts by estimating the parameters in the coast down filter model. The estimation is done by optimizing the negative log likelihood, calculated by the UKF using the coast down filter model, using MATLAB's *fmincon* algorithm, which is a gradient based algorithm. The parameter space is bounded to reasonable limits. To improve the chance of finding a global minimum the optimization is started from 20 staring points uniformly drawn from the space of allowable parameters. The default parameters are used for *fmincon*.

When the parameter estimation using the coast down model is finished the full filter model is used to estimate a subset of the remaining parameters. Due to the number of parameters in the full model experiment a gradient based algorithm is assumed to be to slow and unlikely to find a global minimum. Especially since the gradient of the criterion has to bes estimated numerically. MATLAB's *Particle swarm optimization* was therefore instead used as it allows to search through a larger portion of the parameters space with fewer criterion function evaluations. The default number of particles, 10 times the number of parameters is used, and the optimization is allowed to continue until there has been either 20 iteration without improvement or for a total of 100 iterations.

After the particle swarm optimization has ended and hopefully found a value close to the global maximum the resulting parameters are further optimized using *fmincon*. Where available the computations are run in parallel.

Chapter 6

Parameter estimation

In this chapter some of the more practical aspects of the parameter estimation are presented. First the parameters that are known a priori will be determined and then reasonable ranges for the parameters that will be estimated are set. The parameter estimation algorithm will be tested on simulated measurements to give an indication of the expected performance of the estimator and to uncover any obvious flaws

6.1 Choosing parameters to be estimated

The model developed in chapter 4 has 42 parameters. Some of these are known parameters from the vehicles specifications and other can be measured easily. The remaining parameters must either be estimated if an accurate value is assumed to necessary or can alternatively be approximated using educated guesses if they are assumed to be less important for the accuracy of the model.

6.1.1 Known and approximated parameters

The mass of OLAV was measured with a vehicle scale that had a claimed accuracy of 20 kg. The total mass was measured by placing the whole vehicle without any occupants on the scale. In addition to this the weight of the vehicle on each axle and individual wheel was measured, as well as the total mass with two occupants manning the vehicle during the while obtaining the measurements used for parameter estimation. These measurements weight measurements are summarized in table 6.1. From these measurements it can be be deduced that $m_{v,e} = 1040\,\text{kg}$ and $m_{v,p} = 160\,\text{kg}$. The position of the center off mass in the relation to the axles was estimated as follow:

$$r_{fw} = \frac{\frac{m_{\text{front center}}}{m_{\text{middle center}}} + \left(1 - \frac{m_{\text{rear center}}}{m_{\text{middle center}}}\right)}{2}\text{wheelbase} \tag{6.1}$$

$$r_{rw} = \text{wheelbase} - r_{fw} \tag{6.2}$$

The vehicle specification specifies the wheelbase to 2.05 m, which results in $r_{fw} = 1.26\,\text{m}$ and $r_{rw} = 0.79\,\text{m}$, which seams reasonable given the rear mounted engine and all the instrumentation and batteries placed at the rear of the vehicle. The height of the center of mass is set to $r_z = 0.60\,\text{m}$. This is just an educated guess based on the vehicle geometry and not based on any measurements. The occupants are not assumed to affect this as they are place more or less exactly in the estimated center of mass. The moment of inertia around the pitch axis, $I_{v,\theta}$, was approximated by:

$$I_{v,\theta} = (m_{v,e})\left(\frac{\text{wheelbase}}{2}\right)^2 \tag{6.3}$$

which gives $I_{v,\theta} = 1093\,\text{kg}\,\text{m}^2$.

	Left side	Center	Right side
Front axle	180 kg	400 kg	200 kg
Both axles	500 kg	1040 kg	520 kg
Rear axle	300 kg	640 kg	320 kg
With two occupants		1200 kg	

Table 6.1: Measurements of axle weights and total vehicle wight with and without occupants.

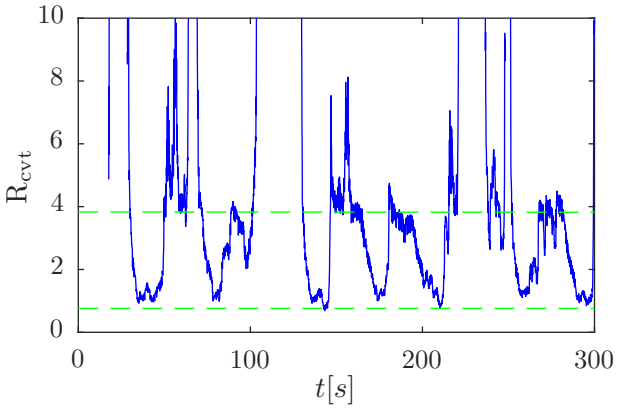


Figure 6.1: Measured speed ratio of the CVT. Dashed lines shows the assumed endpoints for the speed ratio of the CVT, 0.76 and 3.83. Measurement does not take into account belt slip.

Very little information exist about the torque and power characteristics of the engine. There exists no information about the position of the power peak. It is assumed to at 7000RPM or733 $\mathrm{rad\,s^{-1}}$, right before the rev limiter engages, as this gives a relatively flat torque curve which is what the service manual for the vehicle describes [7]. The engine idle speed is specified to be 1250RPM, or 131 $\mathrm{rad\,s^{-1}}$, and the rev limiter is specified to be 7250RPM or 759 $\mathrm{rad\,s^{-1}}$. The moment of inertia of the engine is estimated by assuming a 10 kg flywheel with 0.20 m radius. This gives $I_e = 10\,\mathrm{kg}(0.15\,\mathrm{m})^2/2 = 0.11\,\mathrm{kg\,m^2}$. The optional speed limiter activates at 25 miles per hour or 11.2 $\mathrm{m\,s^{-1}}$ according to the service manual [7], $v_{\max} = 11.2\,\mathrm{m\,s^{-1}}$ătherefore chosen. The idle controller gain was set to $p_{e,\mathrm{idle}} =$ 1E−3, but is assumed to not be very critical.

$s_{b,\max} = 0.2$ was chosen for the slip between the belt and the pulley producing maximum friction in the CVT. The CVT seems to engage the belt at approximately $\omega_{be} = 209\,\mathrm{rad\,s^{-1}}$ (2000RPM). The speed ratios of the CVT at maximum up-shift and down-shift are not specified in the service manual, neither where any other public specifications for this found. There does exist public specifications for the Polaris P90 CVT on the frequently asked questions page of the Polaris SAE sponsorship program webpage [21]: 3.82 : 1 maximum down-shift and 0.76 : 1 up-shift. It is possible that these ratios are shared between many of Polaris's CVT-s and adapted to the specific vehicle by tuning the shifting mechanism's and transmission ratios. The CVT speed ratio can be roughly estimated with:

$$R_{\mathrm{cvt}} = \frac{\omega_e}{R_H \omega_w} \tag{6.4}$$

although this not take into account belt slip. A plot comparing the estimated speed ratio and the specifications of the P90 is shown fig. 6.1. It shows that $R_{\mathrm{cvt},U} = 0.76$ and $R_{\mathrm{cvt},D} = 3.83$ are very likely candidates when taking the belt slip into consideration as the belt will slip significantly at lower engine speeds and thereby making the estimated speed ratio larger.

The speed ratios of the gears in the transmission are specified in the service manual [7]. They are 10.4, 25.59, and −22.92 for high range, low range, and reveres respectively. These speed ratios include the speed ratio provided by the final drive gearing of the rear differential. The inertia of the transmission is approximated by a 5 kg disc with 0.10 m radius giving

$I_t = 5\,\mathrm{kg}(0.10\,\mathrm{m}^2/2 = 0.025\,\mathrm{kg\,m}^2$. The friction in the transmission governed by b_t is assumed to be zero. This does not reflect reality, but it is assumed that the model can compensate for this with a lower engine power estimate.

The dynamic tire radius is assumed to be equal to the specified static radius of the tires, 0.330 m. Due to the aggressive tire pattern on the off-road tires fitted to the vehicle the true static radius is larger than the specifications. The critical slip for tires on dry asphalt is typically 0.1 to 0.2. The critical slip is therefore assumed to be $s_{t,\max} = 0.15$. The time constant for the tires was set to $\tau_t = 0.2\,\mathrm{s}$ which gave well behaved simulations. Higher values gave very unrealistic overshots in the traction force due to the slow build up in forces, while lower values caused a lot of high frequency oscillations, particularly at low speeds where the slip can change rapidly. The combined moment of inertia of the front wheels is assumed to be $I_{fw} = 20\,\mathrm{kg} * (0.33\,\mathrm{m})^2 + 2\,\mathrm{kg\,m}^2 = 4.2\,\mathrm{kg\,m}^2$, i.e. as two disks with mass 20 kg and radius 0.33 m plus an additional $2\,\mathrm{kg\,m}^2$ to account for the rest of the rotating mass of the front axle. The rear is approximated similarly: $I_{rw} = 25\,\mathrm{kg} * (0.33\,\mathrm{m})^2 + 2\,\mathrm{kg\,m}^2 = 4.8\,\mathrm{kg\,m}^2$.

As the spring and friction parameters of the CVT, k_{cvt} and b_{cvt}, only govern the speed of the shifting, they are set to values which gave reasonable results during simulation: $k_{\mathrm{cvt}} = 100$ and $b_{\mathrm{cvt}} = 50$.

The brake balance, B_{bk}, is usually biased towards the front wheels on most cars. As the Polaris Ranger has brakes with two pistons on the front brakes and one on the rear brakes this seems to apply to OLAV as well. The brake balance was therefore set to $B_{bk} = 0.6$, i.e 60 % to the front. The time constant for the brake system was set to $\tau_{bk} = 0.5\,\mathrm{s}$

Summary of the 30 parameters which was presented here and are assumed known in the model is shown in table 6.2.

6.1.2 Parameters to be estimated

Excluding the 30 parameters set in section 6.1.1 there are 12 parameters that has to be estimated. Reasonable bounds has to be set to make it feasible to estimate these parameters. The bounds chosen for the parameters are summarized in table 7.1.

The effective tire radius ,r_{eff}, should be somewhere between the static and the dynamic radius. The tire specification specifies the tires as 26 inches in diameter, or 0.33 m in radius, although it is not specified if this includes the fairly aggressive tire pattern, so it may actually be higher. 0.40 m was therefore chosen as the upper bound. The lower bound was set to 0.30 m.

The rolling resistance coefficient b_{rr} could potentially be zero, but never higher than one divided by the dynamic tire radius as the resulting force would be higher than the normal force experienced by the tire. A more realistic estimate would probably be that the torque would result in no more than 10 % of the normal load or $b_{rr} = 0.1/0.33 \approx 1$ where 0.33 is the assumed dynamic tire radius, r_{dyn}. The bound for b_{rr} is therefore set to $0\,\mathrm{N\,m\,rad}^{-1}\,\mathrm{s}$ to $0.3\,\mathrm{N\,m\,rad}^{-1}\,\mathrm{s}$. The drag coefficient, b_{ar}, can also be zero and a reasonable upper bound of ten results in 1000 N of drag force at $10\,\mathrm{m\,s}^{-1}$.

The maximum power is claimed by a Polaris sales brochure for the Ranger XP 900 [18] to be 60hp or approximately 45 kW. It is not specified if this is the horsepower as measured at the flywheel or the wheels, but as it is not specified it is most likely measured at the flywheel as it is always higher. The vehicle registration paper on the other hand specifies it to 17 kW, although this is probably the maximum nominal power and not the peak power. The range for $P_{e,\max}$ was therefore set to 17 kW to 55 kW. The engine friction parameter, b_e, is assumed to be in the range $1\mathrm{E}{-4}\,\mathrm{N\,m\,s\,rad}^{-1}$ to $0.2\,\mathrm{N\,m\,s\,rad}^{-1}$, giving friction torque between 0.05 N m to 200 N m when the engine is at $500\,\mathrm{rad\,s}^{-1}$ and zero throttle.

The engine speed dependant shifting parameter of the CVT, u_{cvt}, is the engine speed minus the CVT engagement speed necessary to completely overcome the pre-load spring and reach maximum up-shift assuming no torque load. u_{cvt} therefore has to be less than $\omega_{e,\max} - \omega_{\mathrm{engagement}} = 550$ as it is possible to reach maximum up-shift, but larger than zero due to the structure of the model. The range for u_{cvt} is set to 1 to 550. Torque dependant down-shift parameter d_{cvt} also has to be larger than zero due to the model structure. 1000 was chosen as a reasonable upper limit, resulting in the range 1 to 1000 for d_{cvt}.

The possible range for the tire coefficient of friction, μ_t, is between 0 assuming that the

Parameter	Value	Source
$m_{v,e}$	$1040\,\mathrm{kg}$	Measured with vehicle scale.
$m_{p,e}$	$160\,\mathrm{kg}$	Measured with vehicle scale.
$I_{v,\theta}$	$1092\,\mathrm{kg\,m^2}$	Estimated from measured mass and vehicle dimensions.
r_{fw}	$1.26\,\mathrm{m}$	Estimated from measurements of axle weights.
r_{rw}	$0.79\,\mathrm{m}$	Estimated from measurements of axle weights.
r_{cg}	$0.60\,\mathrm{m}$	Guess based on vehicle geometry.
b_s	$50\,\mathrm{kN\,m\,s\,rad^{-1}}$	Gave reasonable results during simulation.
I_e	$0.2\,\mathrm{kg\,m^2}$	Guess based on engine size.
$\omega_{P_{e,\max}}$	$733\,\mathrm{rad\,s^{-1}}$	Guess based on engine size and engine speed limit.
$\omega_{e,\max}$	$759\,\mathrm{rad\,s^{-1}}$	Vehicle specification.
$\omega_{e,\mathrm{idle}}$	$131\,\mathrm{rad\,s^{-1}}$	Vehicle specification.
$p_{e,\mathrm{idle}}$	$1e-3$	Gave reasonable results during simulation.
$v_{\max}$	$11.2\,\mathrm{m\,s^{-1}}$	Vehicle specification.
ω_{be}	$209\,\mathrm{rad\,s^{-1}}$	Based of observation.
$s_{b,\max}$	0.2	Guess.
$R_{\mathrm{cvt},D}$	3.83	Specification for similar a CVT.
$R_{\mathrm{cvt},U}$	0.76	Specification for similar a CVT.
k_{cvt}	100	Gave reasonable response during simulation.
b_{cvt}	50	Gave reasonable response during simulation.
I_t	$0.075\,\mathrm{kg\,m^2}$	Approximation based on transmission size.
b_t	$0\,\mathrm{N\,m\,s\,rad^{-1}}$	Friction in transmission modelled by less engine power.
R_L	25.69	Vehicle specification.
R_H	10.4	Vehicle specification.
R_R	22.92	Vehicle specification.
I_{fw}	$4.2\,\mathrm{kg\,m^2}$	Approximated from wheel weight and radius.
I_{rw}	$4.8\,\mathrm{kg\,m^2}$	Approximated from wheel weight and radius.
$s_{\mu_{t,max}}$	0.15	Typical value for tire on dry asphalt.
τ_t	$0.2\,\mathrm{s}$	Gave reasonable response during simulations.
r_{dyn}	$0.33\,\mathrm{m}$	Guess based on tire geometry.
τ_{bk}	$0.5\,\mathrm{s}$	Gave reasonable response during simulations.
B_{bk}	0.6	Typical value of brake torque applied to the front wheels.

Table 6.2: List of assumed known parameters.

Parameter	Lower bound	Upper bound
r_{eff}	$0.300\,\text{m}$	$0.400\,\text{m}$
b_{rr}	$0\,\text{N m s rad}^{-1}$	$1\,\text{N m s rad}^{-1}$
b_{ar}	$0\,\text{N m}^{-2}\,\text{s}^2$	$10\,\text{N m}^{-2}\,\text{s}^2$
$\theta_{v,0}$	$-0.2\,\text{rad}$	$0.2\,\text{rad}$
$P_{e,max}$	$17\,\text{kW}$	$55\,\text{kW}$
b_e	$1\text{E}{-4}\,\text{N m s rad}^{-1}$	$0.2\,\text{N m s rad}^{-1}$
u_{cvt}	1	550
d_{cvt}	1	1000
$\mu_{b,\max}$	0.01	10
$\mu_{t,max}$	0.5	1.0
$T_{bk,\max}$	$1\,\text{kN m}$	$5\,\text{kN m}$
k_s	$20\,\text{kN m rad}^{-1}$	$120\,\text{kN m rad}^{-1}$

Table 6.3: List of parameters that will be estimated and plausible bounds for the parameter values.

tire is really bad to about 2 assuming it is the best racing slick in the world, although both 0 and 2 are extremely unlikely. The range was therefore set to a more reasonable range of 0.5 to 1.

The friction coefficient of the CVT belt is not actually a friction coefficient as it also influences the normal force produced given the engine speed. The range 0.01 i to 10 is assumed to cover the possible value for $\mu_{b,\max}$.

The range for maximum brake torque, $T_{bk,\max}$ was set to 1 kN m to 5 kN m. The resulting brake force is calculated by $F_{bk} = T_{bk,\max}/r_{dyn}$. This causes between 3 kN to 15 kN of braking force assuming the tires can provide the needed traction.

Assuming a maximum traction force of 5 kN the resulting torque applied to the vehicle body is approximately $r_{cg}F_{w,x} = 0.60 \times 10\,\text{kN m} = 3000\,\text{N m}$. If this results in pitching of the vehicle body in the range 0.05 rad to 0.3 rad the suspension spring constant, k_s, must be between $20\,\text{kN m rad}^{-1}$ to $120\,\text{kN m rad}^{-1}$.

6.2 Testing the parameter estimation algorithm with simulated data

The verify that the parameter estimation algorithm converges to a optimal result it was tested on simulated data. The simulations used are the same as the ones presented in section 4.9, which contains no measurement noise except for the deterministically generated inclination.

6.2.1 Tuning the unscented Kalman filters

The α tuning parameter for the UKF was determined by testing the filters on the simulated data with different values of α between $1e-3$ and 1. The criterion used for deciding the optimum α was the weighted mean squared error between the estimated state and the true simulated stated.

$$\mathcal{J}_{WMSE}(\alpha) = \frac{1}{N}\sum_{k=1}^{N}\left(\underline{\hat{x}}_k(\alpha) - \underline{x}_k\right)^T W \left(\underline{\hat{x}}_k(\alpha) - \underline{x}_k\right) \tag{6.5}$$

The weight matrix for finding alpha for the coast down filter was set to:

$$W_{cd} = \text{diag}\left\{2 \quad 1 \quad 2 \quad 4\right\} \tag{6.6}$$

placing special emphasis on correct estimation of the inclination of the ground, θ_g, and less emphasis on correct estimation of the horizontal position. The optimum optimum α decided

for the coast down filter model was 1E−3. For the full model the following weight matrix was used:

$$W_{cd} = \text{diag}\left\{1\text{E}{-1} \quad 1 \quad 2 \quad 1 \quad 1 \quad 1 \quad 1\text{E}{-3} \quad 1\text{E}{-3} \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 2\right\} \tag{6.7}$$

Again, a little extra emphasis is put on correct inclination estimation. Both engine speed and the tire traction forces are weighted much lower than the other states because they can have much higher values than the other states. Both of the weight matrices where normalized so that the sum of their diagonal elements are one. The weight matrices where normalized, so that the sum of all elements are equal to one. The weighted mean square error was calculated for 100 evenly spaced values between 1E−3 to 1. The result is shown in fig. 6.2.

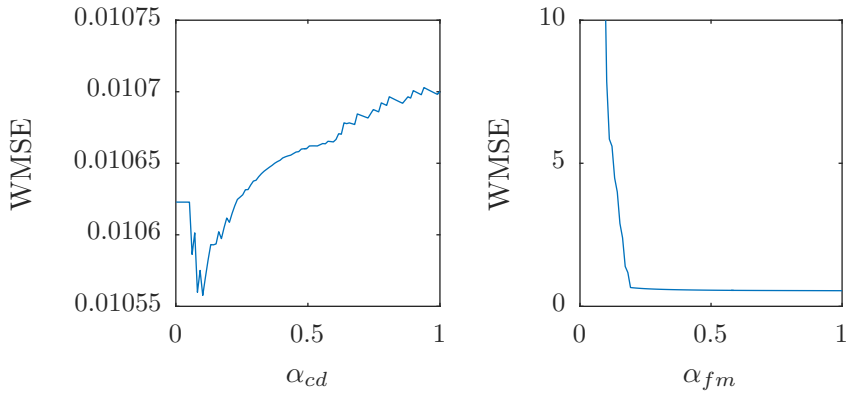


Figure 6.2: Weighted mean square error of estimated and true system state as a function of α for the coast down and full model filters applied to simulated data without process noise. The $WMSE$ for the full model filter continuous to raise to 7.5E9 for $\alpha = 1\text{E}{-3}$. The plot is cut off at $WMSE = 10$ as the higher values clearly are sub optimal choices for α.

As the datasets used to tune the filters do not contain any process noise except for the generated terrain, the only state that will deviate significantly from the filter prediction is the ground inclination, θ_g, which is modeled as a random walk by the filters. As the inclination changes slowly, one would expect to see fairly similar values for the mean squared error for all values of α as the spread of the sigma points should not significantly change the ability of the filter to track the state. The major exception to this is filter models becomes unstable due to the spread of the sigma points.

The weighted mean square error of the state estimate produced by the coast down filter is more or less constant for all values of α. This is the expected result. α for the coast down filter was chosen to be 0.1.

The weighted mean square error of the state estimate produced by the full model on the other hand varies significantly with different values of α. For values ≈ 0.25 and smaller the weighted mean square error raises sharply to a maximum value of 7.5E9 at $\alpha = 1\text{E}{-3}$. A possible explanation for this is that the response of the model becomes to fast for the numerical solver used to discretize the model when the sigma points are placed far from the mean value. This results in an unstable solution for some of the sigma points. Particularly the tire model is assumed to cause problems as the slip, and with it the generated traction forces, can change rapidly, especially at low vehicle speeds where small changes in tire speeds creates large changes in slip.

One possible way to rectify this other than to raise the α is to improve the discretization of the model used for the state prediction making it better suited to handle the stiffness of the system. This can either be done by decreasing the time step, by using a variable time step solver, or by using a implicit solver better suited to the stiff problem. There was not time to implement variable step length or a implicit in the time frame of this thesis, and the solvers provided by MATLAB are to slow for the order of problem: solving a 14 state space equation for $2 * (14 + 14) + 1$ sigma points. Decreasing the time step of the fixed time step Runge Kutta integration routine is not desirable either as the UKF already uses

	−50 %	−10 %	−1 %	1 %	10 %	50 %
r_{eff}	−5.5E5	−1.2E4	−2.1E2	2.5E1	−6.6E3	−1.1E5
b_{rr}	−2.2E3	−8.5E1	−8.0E−1	−9.2E−1	−8.7E1	−2.1E3
b_{ar}	−7.9E1	−2.9	4.7E−3	−6.9E−2	−3.6	−8.3E1
$\theta_{v,0}$	−9.5E1	−9.5	−7.4E−1	6.9E−1	4.8	−2.3E1

Table 6.4: Parameter sensitivity for coast down filter with Runge Kutta based discretization of model. Relative change of log likelihood in percent when changing one parameters from known correct values by ±1 %, ±10 %, and ±50 % from the true parameter value used to generate the dataset. Filter tuning parameter $\alpha = 0.1$.

approximately 1 s to estimate the state from 3 s of measurements, already resulting in fairly long processing time for the parameter estimation.

It was therefore decided to set α sufficiently high to avoid the problem. The α for the full model filter was therefore set to 0.4, giving a bit of margin over the point where the filter model appears to become unstable.

6.2.2 Parameter sensitivity

To find out if the parameter estimation is likely to give good results a rudimentary sensitivity analysis was performed. The filters were tested on the simulated data with the same parameters used during the simulation to give a baseline for the log likelihood. Each of the model parameters was then changed one by one ±1 %, ±10 %, and ±50 % from their known correct values and the log likelihood reevaluated. This is a test to see if changing the parameters to known wrong values decreases the log likelihood as it should. It can also be used to decide which parameters to estimate based on the relative change of likelihood and estimating the parameters that changes it the most.

The results for the two filter models are shown in tables 6.4 and 6.5. Parameters that did not change the value of the log likelihood are excluded (speed ratios of the gears not used etc.) As should be apparent the estimator is unlikely to find the exact parameters used for the simulation. Many of the parameters gives a higher log likelihood for the changed values than the known correct ones. Small variations when changing the parameters with ±1 % or for some parameters even ±10 %, are probably to be expected as the simulated dataset is only 60 s long and has a fairly limited dynamic range. A longer measurement series with more varied inputs would likely provide better accuracy.

The coast down model has good results for r_{eff}, b_{rr}, and b_{ar}, with lower log likelihood when changed ±10 % and ±50 %. $\theta_{v,0}$ on the other hand shows a 4.8 % higher log likelihood when being increased by 10 %. This could mean that the assumption that the suspension dynamics can be ignored does not hold true.

The full filter model shows reasonable result for most of the parameters, but some deviates significantly. A possible reason for the inaccuracy is the stiffness of the system and the sub optimal numerical solver used which was discussed in section 6.2.1. Particularly the parameters which influences the acceleration of the wheels hints at this. Increasing the wheel and transmission inertia, I_{fw}, I_{rw}, and I_t, all results in a higher log likelihood, even at a 50 % increase. Although the log likelihood also increases then the values for I_{rw} and I_t are lowered. These parameters are already assumed to be known a priori and are not part of the parameters that will be estimated.

Because the simulated dataset spends no time at idle, the parameters controlling the idle speed, $\omega_{e,\text{idle}}$, and gain of the idle controller, p_{idle}, does not change the likelihood. The same is true for the max engine speed, $\omega_{e,\max}$, which only influences the likelihood when reduced significantly.

6.2.3 Parameters estimated from simulated data

The complete parameter estimation algorithm was tested on the two simulated dataset. The estimation took 9 h and 45 min running on a 4 core 2.93 GHz Intel Xenon X5570. The

	−50 %	−10 %	−1 %	1 %	10 %	50 %
$m_{v,e}$	−1.6E2	−1.4	5.0E−2	−1.8E−1	−1.8	−3.9E1
$m_{v,p}$	−3.1E−1	1.0E−1	1.2E−1	9.4E−3	−5.5E−2	−9.0E−1
r_f	−5.4E1	−1.1	−2.0E−2	1.5E−1	9.7E−1	−4.3
r_r	−9.0	1.0	1.6E−1	−1.4E−2	−1.0	−1.7E1
r_{cg}	−6.2E1	−2.2	7.6E−1	9.8E−1	−6.4E−1	−2.9E1
$I_{v,\theta}$	−1.2	−1.4E−1	5.8E−2	1.1E−2	9.4E−2	−6.4E−1
k_s	−5.9E1	−7.7E−1	1.1	8.7E−1	−1.8	−3.0E1
b_s	−3.9	−1.9E−1	−1.7E−2	4.9E−2	8.0E−1	−1.2
I_e	−4.9E1	−1.2	−3.4E−2	1.0	2.0E−1	−2.1E1
$P_{e,\max}$	−4.3E2	−1.7E1	5.9E−1	2.1E−1	−1.3E1	−3.8E2
b_e	−6.0E−1	9.4E−1	6.5E−2	1.1E−1	7.2E−1	−2.9E−1
$\omega_{e,\text{idle}}$	0.0	0.0	0.0	0.0	0.0	0.0
$\omega_{P_{e,\max}}$	−3.8E2	−1.1E1	9.5E−1	7.7E−1	−1.1E1	−1.8E2
$\omega_{e,\max}$	−1.2E3	0.0	0.0	0.0	0.0	0.0
$v_{\max}$	−7.5E2	−3.9E1	−3.2E−1	−2.5E−1	−1.6	−1.6
$p_{e,\text{idle}}$	0.0	0.0	0.0	0.0	0.0	0.0
ω_{be}	−1.2E3	−5.0E1	4.0E−1	5.2E−1	−4.3E1	−9.6E2
$R_{\text{cvt},U}$	−2.4E3	−4.0E1	6.6E−1	−2.1E−1	−2.5E1	−3.0E2
$R_{\text{cvt},D}$	−1.2E1	−7.5E−1	−1.1E−2	1.6E−1	−3.1E−1	−1.7E1
R_H	−2.1E3	−2.7E1	7.0E−1	−5.3E−2	−1.4E1	−1.8E2
u_{cvt}	−2.0E3	−3.5E1	−4.0E−1	−1.5E−1	−2.4E1	−3.7E2
d_{cvt}	−4.9E2	−7.0	8.2E−2	−1.3E−1	−6.0	−8.2E1
k_{cvt}	−4.8	−1.5E−1	3.1E−2	4.9E−2	−6.4E−2	−1.2
b_{cvt}	−4.7	−1.5E−1	9.0E−2	3.4E−2	−1.1E−2	−1.8
$\mu_{b,\max}$	−7.4E1	−9.5E−2	−1.7E−1	7.1E−2	−1.4	−1.8E1
$s_{b,\max}$	−2.8	−8.6E−2	1.4E−1	6.9E−2	5.0E−2	−1.3
I_t	8.0E−1	1.0E−1	1.7E−2	5.8E−2	8.3E−2	2.3E−1
r_{dyn}	−1.8E2	−1.0	7.0E−2	8.3E−1	−2.8	−5.5E1
I_{fw}	−5.2	−1.2	9.1E−2	1.1E−1	1.2	1.9
I_{rw}	4.6E−1	3.7E−2	6.0E−2	4.3E−2	1.5E−1	1.3E−1
$\mu_{t,\max}$	−2.7E2	−2.6	2.9E−2	6.6E−2	−1.7	−3.5E1
$s_{\mu_{t,\max}}$	−3.8E1	−1.1	−4.0E−2	1.1E−1	−1.5E−1	−2.9E1
τ_t	−2.3	6.7E−1	1.1E−1	9.7E−2	2.9E−2	−2.1
$T_{bk,\max}$	−4.4E1	−2.9E−1	9.7E−1	5.0E−2	−1.4	−1.9E1
τ_{bk}	−1.0E1	−4.6E−1	1.1E−1	1.0	9.0E−1	−2.0
B_{bk}	−2.2E1	−1.4	5.0E−2	−1.8E−2	−3.3	−8.3E1

Table 6.5: Parameter sensitivity for full model filter with Runge Kutta based discretization of model. Relative change of log likelihood in percent when changing one parameters from known correct values by ±1 %, ±10 %, and ±50 % from the true parameter value used to generate the dataset. Filter tuning parameter $\alpha = 0.4$.

	Estimated	True	Difference/%
r_{eff}	3.52E−1	3.50E−1	6.42E−1
b_{rr}	1.98E−2	2.00E−2	−1.08
b_{ar}	2.05	2.00	2.56
$\theta_{v,0}$	−5.80E−2	−5.00E−2	1.59E1
$P_{e,\max}$	3.41E4	3.36E4	1.50
b_e	7.04E−3	1.00E−2	−2.96E1
u_{cvt}	3.00E2	3.00E2	4.87E−2
d_{cvt}	2.04E2	2.00E2	2.04
$\mu_{b,\max}$	5.09E−1	5.00E−1	1.85
$\mu_{t,\max}$	6.39E−1	7.00E−1	−8.69
T_{bk}	4.73E3	5.00E3	−5.32
k_s	4.07E4	5.00E4	−1.87E1

Table 6.6: Parameters estimated from simulated data, their known true values, and the estimation error in percent.

estimated parameters as well as their known true values are shown in table 6.6.

Most values are reasonably close to their true values considering the small datasets and the already discussed problems with the full model filter. The largest errors are the engine friction parameter, b_e, which is estimated 30 % to low and the suspension spring parameters, k_s, which is estimated 19 % to low. As these parameters are estimated with the full model filter with the parameters estimated by the coast down filter, it is possible that these errors are sequential errors caused by the error in the estimate of the steady state vehicle pitch offset, $theta_{v,0}$, which was estimated 16 % from the true value.

6.2.4 State estimation and simulation using estimated parameters

The model was tested with the parameters estimated with the simulated data both for simulation and state estimation. The result of the state simulation and estimation in addition to originally simulated state are shown in fig. 6.3 on the following page. The reference state simulation used is the same used to generate the measurements used to estimate the parameters originally presented in section 4.9.

Even with the slightly wrong parameters, the UKF tracks the known true state perfectly, although the rejection of the measurement noise is not perfect. The simulation using the estimated parameters also follows the state closely. As the original simulation was generate without process noise and the same data as used for the parameter estimation was used, this is the most ideal scenario imaginable, and can therefore only be used as an upper bound for the performance that can be expected.

6.3 Experimental data used for parameter estimation

The cost down model is given two 60 s long measurement series of the vehicle coasting down a hill. The measurement series are shown in fig. 6.4. Both series are from the same hill. The hill has three speed bumps as well as some other irregularities which can be clearly seen by the measurement of the vehicle pitch, θ_v.

Due to the time required to estimate the parameters the full filter model was also only given two 60 s measurement series. These measurement series are shown in fig. 6.5. One series consists of driving up the hill, and the other down the hill, both with relatively varied throttle and brake inputs.

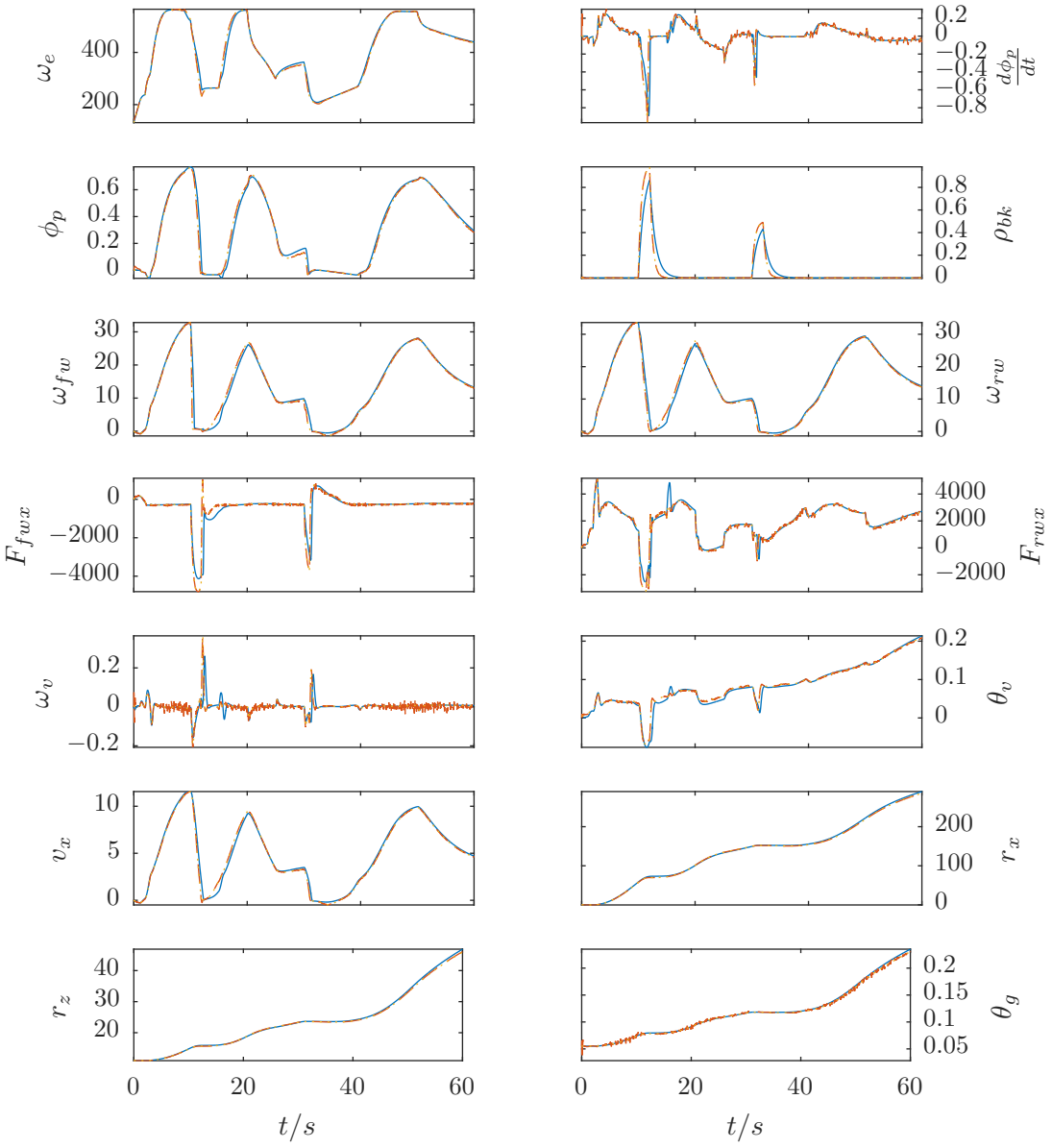


Figure 6.3: State simulation and estimation using model with estimated parameters from simulated data. The whole lines is the simulation using the estimated parameters, the dashed is the estimated state, and the dotted is the original simulated state used to generate the measurements for parameter and state estimation.

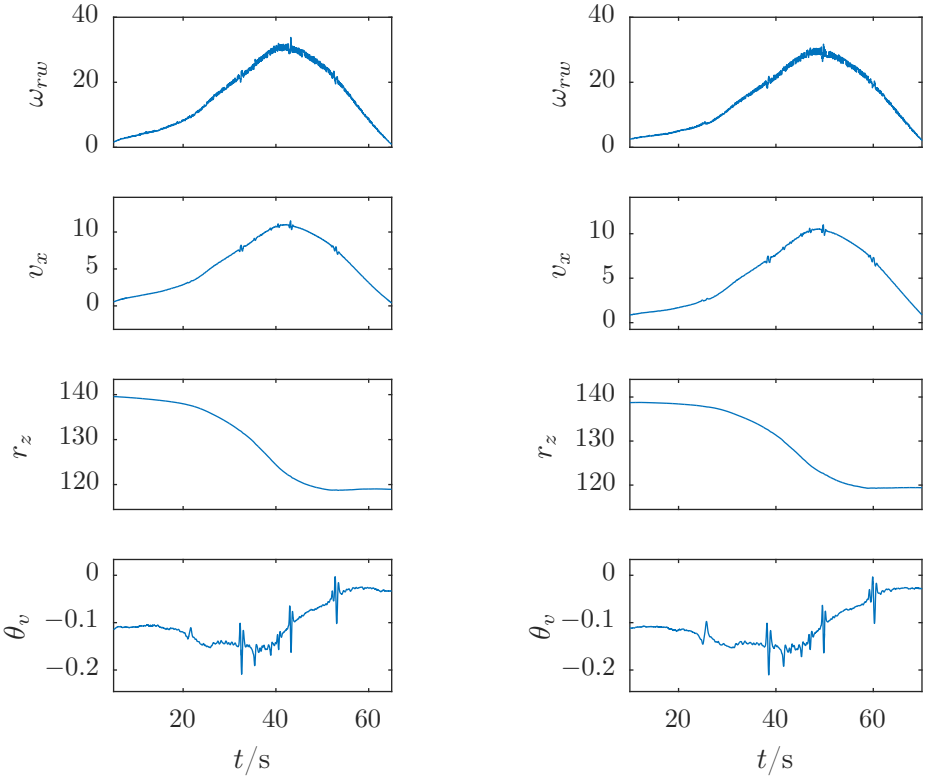


Figure 6.4: Datasets used for estimating parameters using the coast down filter model.

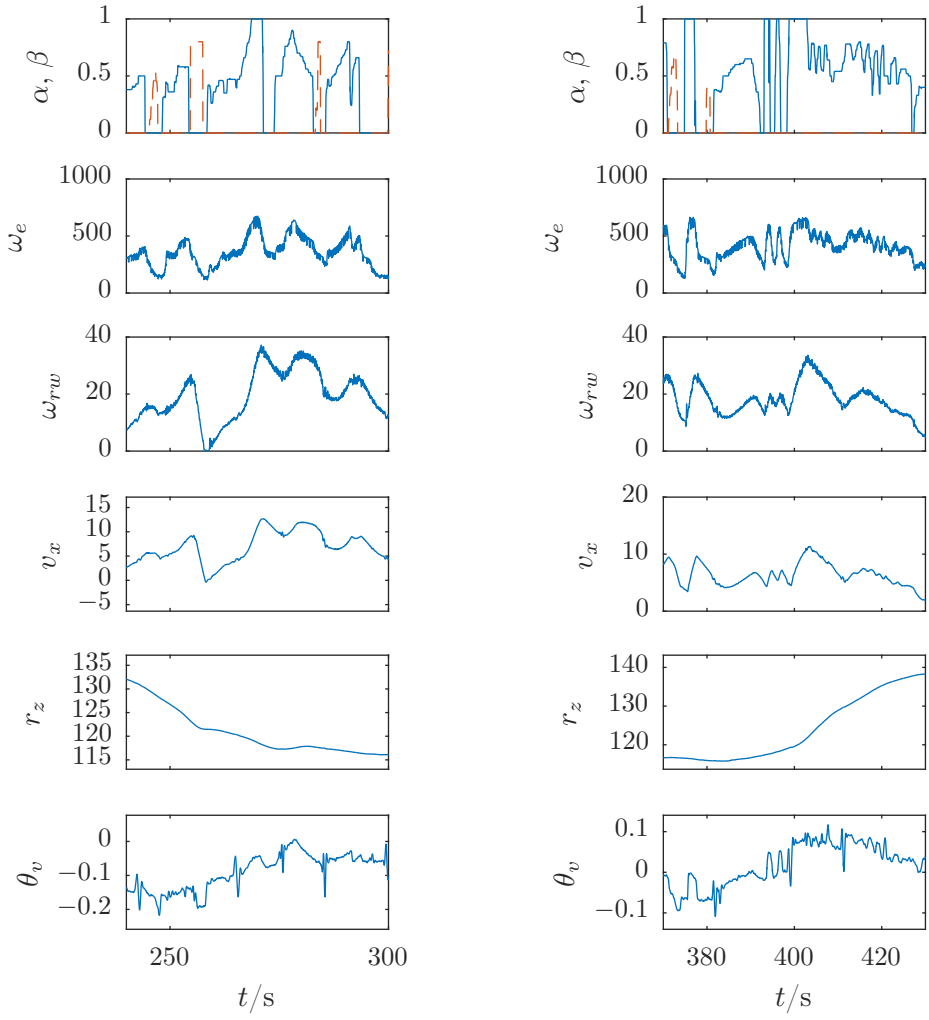


Figure 6.5: Datasets used for parameter estimation using the full vehicle filter model. First subplot shows throttle and brake inputs. The dashed line is the brake input.

Chapter 7

Estimation results and model performance

In this chapter the parameters estimated from measurements of the vehicle will be presented. The performance of the model will be test by comparing simulations of the model and observations of the vehicle as well as showing and example of the estimated state when used for state estimation.

7.1 Estimated parameters

Estimating the parameters from the datasets presented in section 6.3 took 16 h and 30 min. The estimated parameters are shown in table 7.1. As the parameter estimation algorithm estimated biased parameters when tested on simulated data these parameters are likely also biased.

The parameters, generally looks reasonably close to what was expected. The major exception is the peak engine power, $P_{e,\max}$, which was estimated to 54.8 kW which is very close to the upper bound set for it. It is unlikely that it truly is this high, which hints at that the engine model is probably inaccurate. The estimated coefficient of friction for the tires, $\mu_{t,\max}$, estimated to 0.678 seams reasonable give that they are off-road tire and thereby not optimized for asphalt. It is hard to say anything about the remaining parameters, but they seem reasonable given that they are placed well inside their respective upper and lower bounds.

r_{eff}	3.56E−1
b_{rr}	1.56E−2
b_{ar}	2.55
$\theta_{v,0}$	−4.69E−2
$P_{e,\max}$	5.48E4
b_e	1.15E−1
u_{cvt}	2.76E2
d_{cvt}	1.97E2
$\mu_{b,\max}$	2.79E−1
$\mu_{t,\max}$	6.78E−1
T_{bk}	3.40E3
k_s	6.43E4

Table 7.1: Parameters estimated from measurements of the vehicle.

7.2 Simulations using the estimated parameters

To evaluate how well the model represents the dynamics of the vehicle, simulated measurements generated by the model are compared with recorded measurements of the vehicle. The ground inclination, θ_g, and terrain altitude, r_z, used for the simulation is estimated using the UKF, with the full filter model using the estimated parameters, from the dataset the simulation is compared to. The estimated θ_g and r_z are placed in a look-up table index by the estimated r_x. $\frac{d\theta_g}{dr_x}$ is set to zero.

As the terrain is estimated using the model, the results will likely be better than what can be achieved if the terrain was measured using some other method, as the estimated terrain may compensate for inaccuracies in the model. The initial conditions used for the simulations is the state estimated by the UKF after five seconds. The throttle and brake inputs from the simulation is the same as the recorded in the dataset and are applied at the same time in the simulation. The model will be tested on data recorded from the same scenarios as used for the parameter estimation: coasting down a hill in neutral as well as regular driving.

7.2.1 Coasting down hill

The comparison between the measurements simulated by the model with estimated parameters and measurements recorded from the vehicle when coasting down a hill is shown in fig. 7.1 on the next page. The simulation does not correspond particularly well with the observations from the vehicle: the model accelerates much slower than the recorded measurements, although the maximum speed achieved is relatively similar. As the parameter estimation algorithm is know to produce a biased estimate of the static input between vehicle pitch and ground inclination, $\theta_{v,0}$, the comparison was repeated with $\theta_{v,0}$ adjusted by 16 %, to $\theta_{v,0} = -3.94\text{E}{-2}$, in both the filter model used to estimate the terrain and the simulation model. The results of this comparison is shown in fig. 7.2 on page 68. This drastically improved the results almost matching the observations perfectly.

7.2.2 Driving

The simulation of normal driving of the vehicle was first tested against the best case scenario, i.e one of the datasets used for estimating the parameters. The comparison is shown in fig. 7.3 on page 69. The simulated follows the general trend of the measurements from the dataset quite well, but there is generally a quite large difference between the simulated and measured behaviour. Particularly the response during deceleration, both when using the brakes and when just using engine braking, is exaggerated. The engine speed on the other hand is relatively equal in both the simulation and measurements. The acceleration also seem to be to high, particularly at low throttle settings. This could be caused by errors in the engine, CVT and or the brake model.

Comparing the model to a new set of measurements is shown in fig. 7.4 on page 70. It shows more of the same behaviour: acceleration is too high at low and medium throttle as well as too high deceleration when braking. The model with the current set of assumed and estimated parameters generally does not correspond well to the observed vehicle behaviour.

7.3 State estimation

Figure 7.5 on page 72 shows state estimation with the UKF using the model with estimated parameters as filter model. The state is estimated from the observed measurements shown in fig. 7.4 on page 70. The estimated state looks reasonable, with no obvious discrepancies, like the front wheels suddenly stopping or turning at a speed very different than that of the front wheels. The brake pressure, ρ_{bk}, are estimated as always being slightly active. Although this is reasonable, as the brake pads are likely to always be slightly in contact with the brake discs, this was not the intended behaviour of the model. The covariance of the state estimate, represented by ± 1 standard deviation, is estimated unreasonably low.

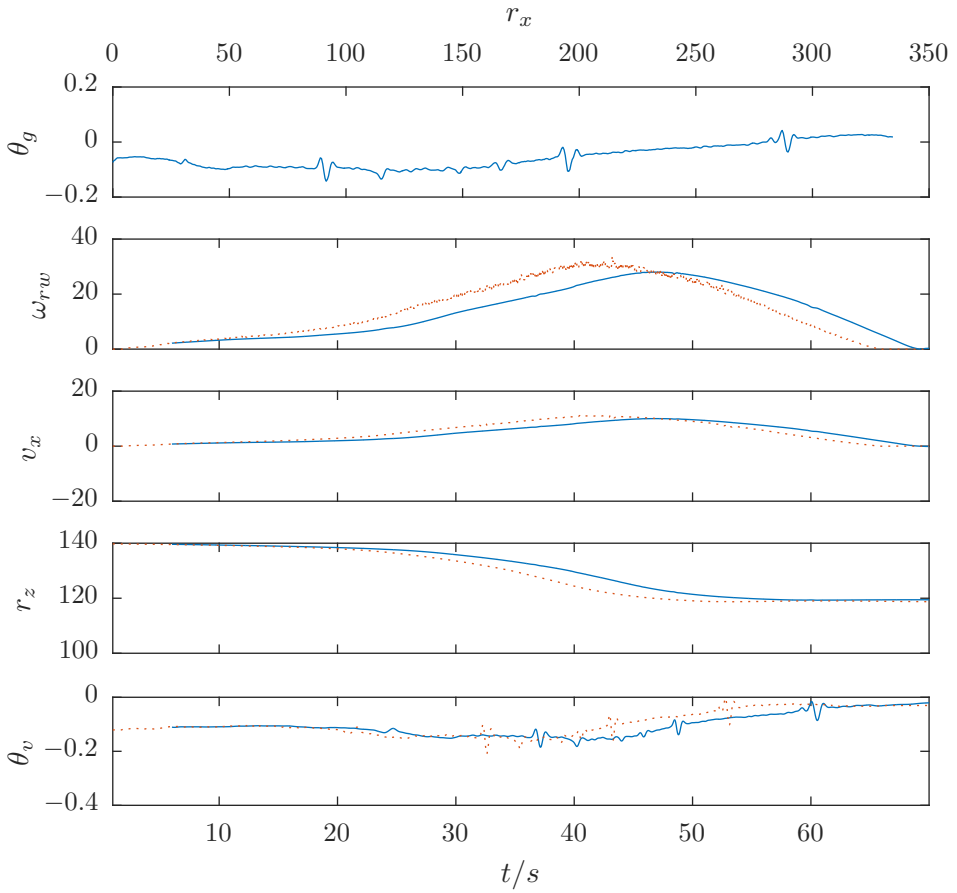


Figure 7.1: Comparison of simulations of model with estimated parameters and observations of the vehicle coasting down a hill. First subplot shows the inclination estimated by the UKF and the remaining subplots compares simulated and observed measurements. Dotted lines are observations from the vehicle.

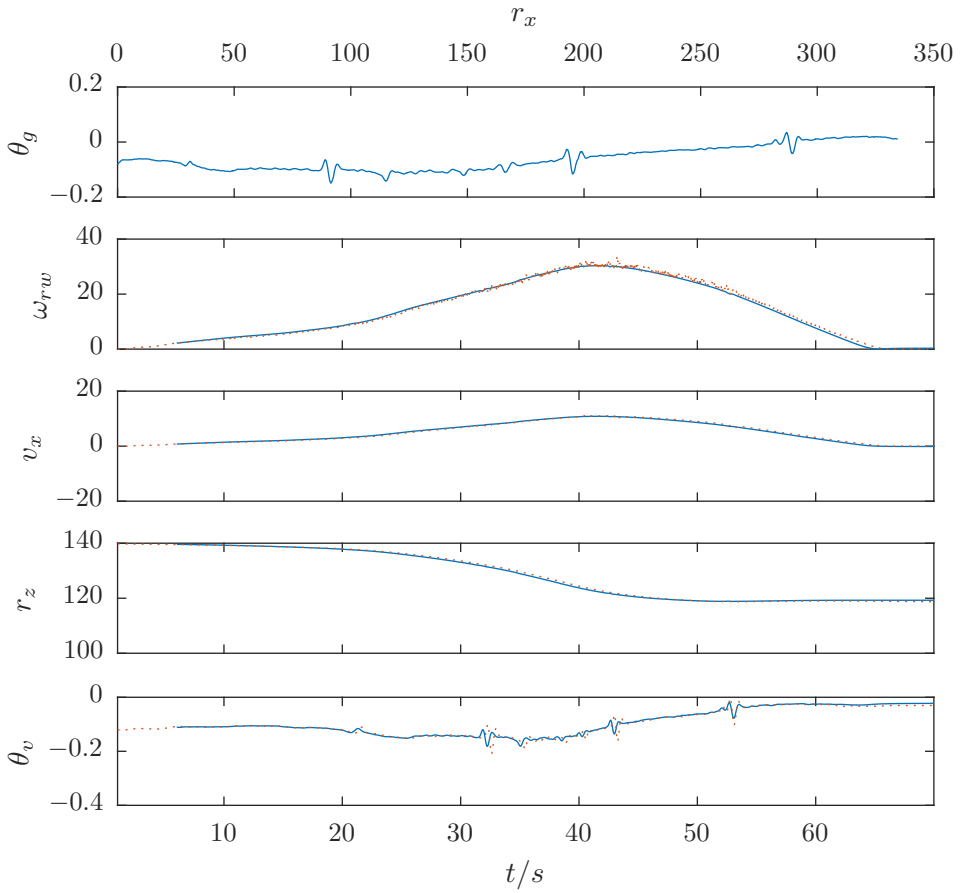


Figure 7.2: Comparison of simulations of model with $\theta_{v,0}$ adjusted for estimation bias and observations of the vehicle coasting down a hill. First subplot shows the inclination estimated by the UKF and the remaining subplots compares simulated and observed measurements. Dotted lines are observations from the vehicle.

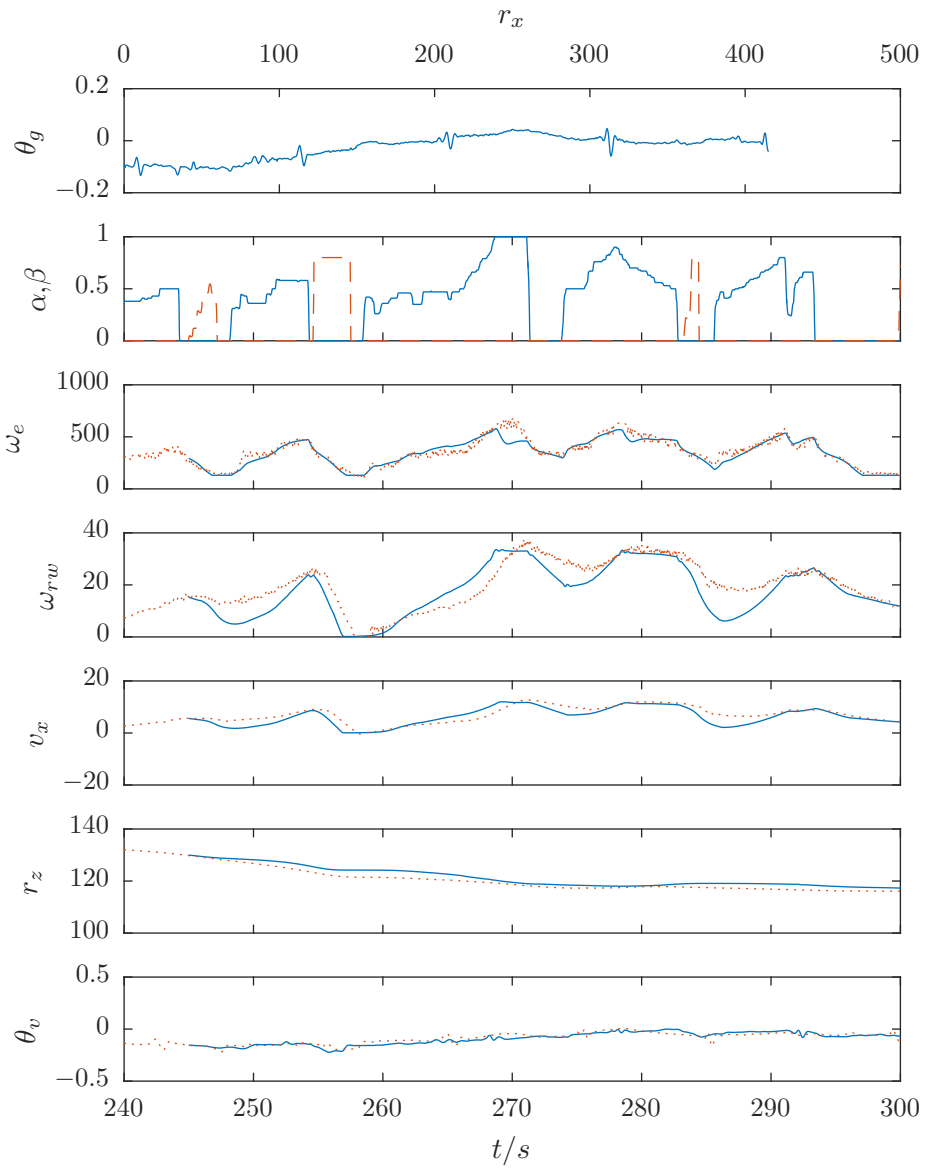


Figure 7.3: Comparison of simulations of model with estimation parameters and observations of the vehicle driving. First subplot shows the inclination estimated by the UKF, the second subplot the throttle and brake inputs, and the remaining subplots compares simulated and observed measurements. Dotted lines are observations from the vehicle. The observation of the vehicle is one of the datasets used while estimating the parameters.

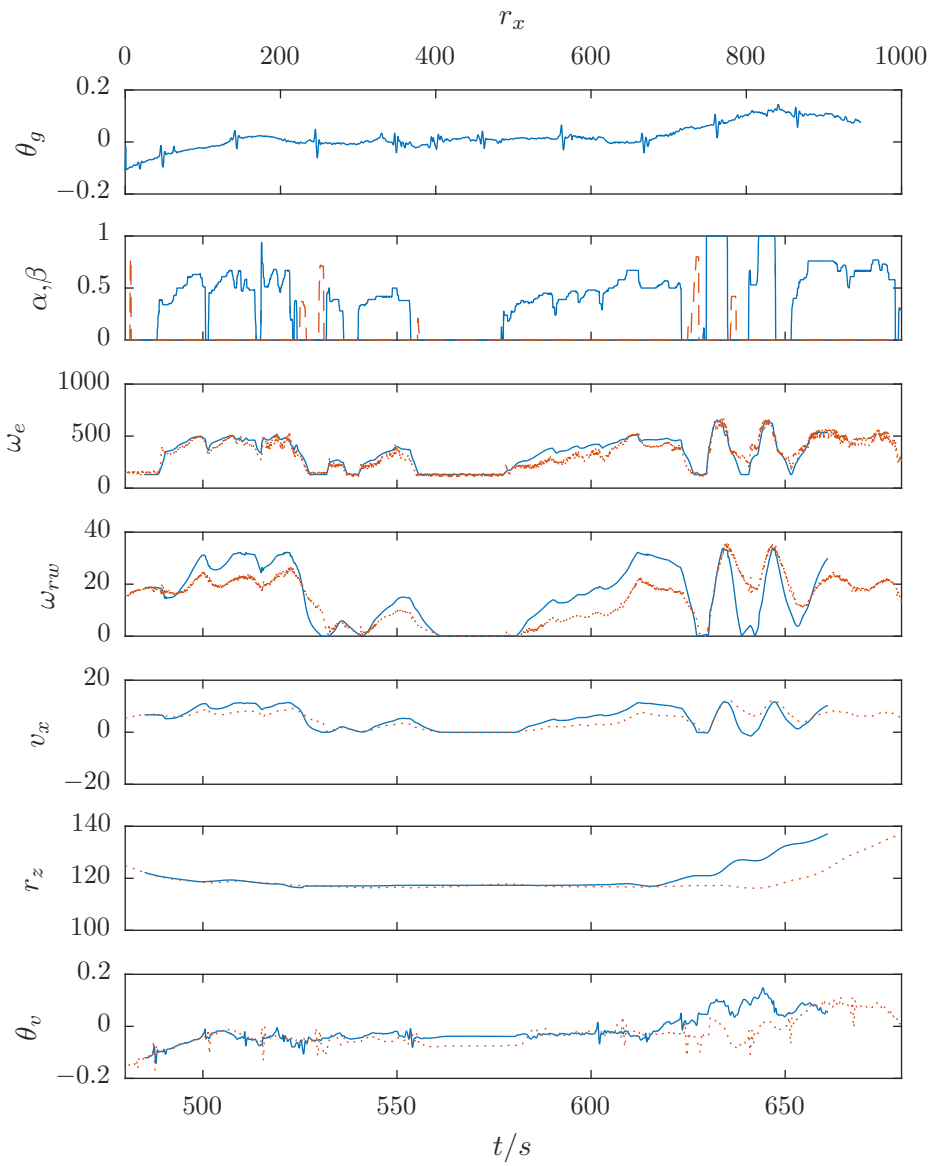


Figure 7.4: Comparison of simulations of model with estimation parameters and observations of the vehicle driving. First subplot shows the inclination estimated by the UKF, the second subplot the throttle and brake inputs, and the remaining subplots compares simulated and observed measurements. Dotted lines are observations from the vehicle.

This is partly explained by the low covariance of the measured states, but likely also caused by too low process noise in the filter model.

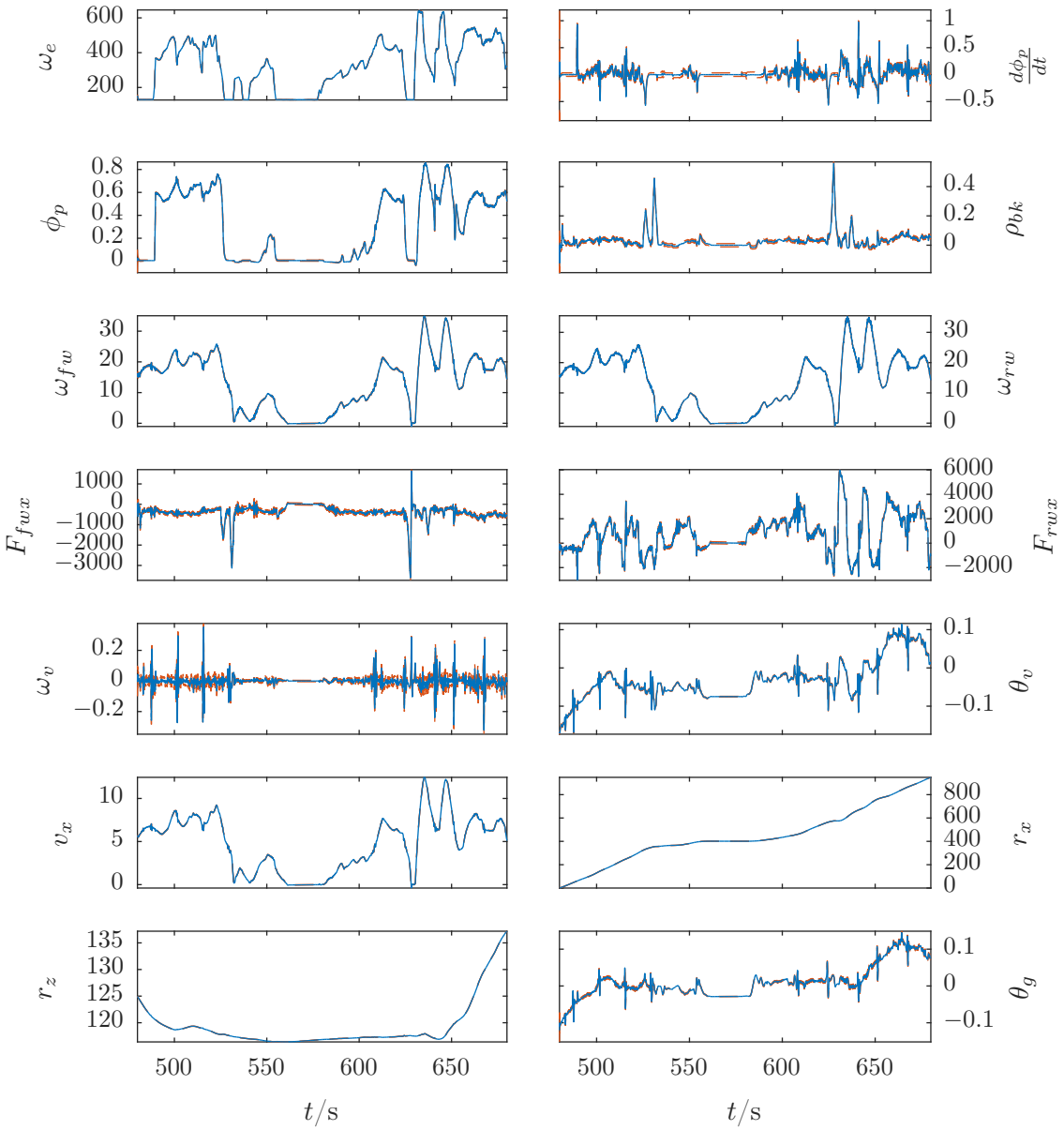


Figure 7.5: State estimate of the dataset in fig. 7.4 using the full model with estimated parameters as filter model. Whole lines are the estimated means and the dashed lines are the estimated covariance represented by the mean ± 1 standard deviation calculated from the diagonal of the estimated covariance matrix.

Chapter 8

Discussion

As the simulation results shown in chapter 7 show, the model and estimated parameters does not conform well with the observed vehicle behaviour. When comparing the modeled and observed behaviour when coasting in neutral, there was a quite substantial difference in the speed, but in this scenario it was shown that the inaccuracy was primarily caused by biased estimation of the static offset between vehicle pitch and ground inclination. By correcting this parameter with the estimation offset found when testing the parameter estimation on simulated data, the resulting simulation was improved dramatically. For regular driving using the throttle and brake the overall shape of the simulated response was consistent with the observations, but both acceleration and deceleration is exaggerated. The error was too large to be described by inaccurate parameter estimation alone and is believed to also be caused by modeling errors. Using the model as a filter model for state estimation, even with slightly wrong parameters, worked well on simulated data. State estimation with measurements from the vehicle gives reasonable results, even with the moderately large model errors, although these results are hard to verify.

Unfortunately, as a lot of time was spent on developing and implementing the model, state estimator and parameter estimation algorithm, the presented results where found close to the deadline for the thesis, leaving no time to improve upon them. A lot of attention was spent on the development of the presented model, and presented model is only the final iteration of many incremental attempts. The resulting model is more complex than initially expected, but it is believed that a model of this order is necessary to accurately represent the dynamics of the vehicle. The complexity of the model resulted in slow state estimation and parameter estimation. Early attempts at the parameter estimation took many days to complete and the final parameter estimation algorithm presented in the this thesis is the result of reducing the number of parameters to the absolute minimum number that is assumed to be sufficient to give good results. Despite this, estimation of the parameters still takes about 16 h to complete. This meant that iterative improvements where slow, which is why attempts to improve the final results further have not been made.

8.1 Vehicle dynamics model

One of the problems encountered with the vehicle model was the stiffness of the equations, i.e sudden and rapid changes significantly faster the overall dynamics. Particularly this applies to the speed of the wheels, as the tire traction forces can build up very quickly, especially at low speeds, as small changes in wheel speed results in large changes in the calculated slip. This leads to numerical instability of the model with the fixed step Runge Kutta method used for discretization of the model. This was part of the motivation for adding a time delay to the developed traction forces at the cost of adding two extra states, as it made the stiffness manageable.

One of the assumed inaccuracies in the model is the engine torque model. The model was an attempt at describing the engine torque with as few parameters as possible, which lead to some large assumptions. The full throttle power and torque is similar to what marketing material from Polaris describe, but nothing is known about shape at other throttle settings.

This lead to a rather rash assumption about its shape: that it can be described by a linear combination of the full throttle curve and viscous friction. This is likely why the both the maximum torque provided by the engine and the friction seems to be overestimated by the parameter estimation, as the vehicle spends most of the time at around half throttle where the model is least accurate.

A new model was developed for the CVT. This was done because the models described in the literature either was overly complex or did not describe the mechanical control system, which meant that they required large alterations to be used in the vehicle model anyway. The overall structure of the model is assumed to describe the CVT reasonably well, although this is not well proven and it is not known if it is the main contributor to the model error. Some of the assumptions made about the speed and torque sensing devices of the CVT are likely wrong. Particularly the speed to up-shift relationship is probably not linear as it is assumed to be.

The terrain is modeled in the filter models as a stochastic process, in the from of integrated zero mean Gaussian noise scaled by the vehicle velocity. As shown by fig. 6.3 on page 62 this works well when the filter model is relatively accurate and the terrain is smooth and changing slowly. The inclination estimated from measurements from the vehicle seems reasonable enough, although the true terrain is unknown so little can be concluded from this. A more thorough investigation using more realistic simulated data therefore has to be performed before anything can be concluded.

8.2 Parameter estimation algorithm and filter models

The maximum likelihood based parameter estimation approach, using a state estimator to calculate the likelihood, is an approach well described in the literature. The UKF was chosen as a state estimator both because it has good estimation performance and because it does not need to evaluate the Jacobian of the state transition model. Two distinct experiments and filter models where used to reduce the time needed to perform the optimization. This reduced the processing time considerably as four of the parameters could be estimated by a much simpler filter model which does not include the drive line model.

The reduced model is used to estimate the effective wheel radius, rolling resistance, drag and the static offset between the ground and sensors measuring the orientation of the vehicle. The tire traction forces are assumed to be so small that no wheel slip is present. This assumption seems to be right as both the effective wheel and rolling resistance was estimated to 0.6 % and 1.1 % of their true values when tested using simulated data. The other assumption which was made was that there is no movement in the suspension. This seems to have been a bad assumption, as the static suspension offset between vehicle and ground is estimated 16 % from its true value suing the same simulated data. As shown in fig. 7.1 and fig. 7.2 in section 7.2.1, where the model with parameters estimated from observed measurement is compared to observed measurements, this seems to influence the model performance quite significantly. With the estimated static suspension offset, the simulated and observed behaviour do not correspond well, but when the suspension offset is reduced by 16 %, the same amount it was estimated wrong by during simulations, it matches almost perfectly.

Estimation of the remaining parameters using the full vehicle filter model also gave non-negligible errors for some of the estimated parameters when tested on simulated data. A rudimentary sensitivity analysis was performed, and showed that many of the parameters gave better results when moved from their known true values. Particularly when moving parameters which dampens the system response when moved the calculated likelihood increased. It is therefore believed that the errors are partially related to the already mentioned stiffness of the vehicle model. As the length of the simulated dataset was only 60 s, it may also be possible that the results would improve with a longer measurement series with more varied dynamics.

The α tuning parameter for the UKF was tuned by plotting the weighted mean squared error between the estimated state and the known state of the estimation and choosing the lowest value which did not significantly increase the weighted mean square error. The coast down filter had almost the same error for all values α, with smallest value at $\alpha = 0.1$. The

error for the full filter model exploded for small α values. This is again assumed to be caused by the model stiffness leading to numerical instability when integrated by the fixed step Runge Kutta method. As the sigma points are placed further from the mean when α is decreased, this results in the sigma points experiencing more aggressive dynamics. For example: if one of the sigma points places the angular velocity of one of the wheels a large distance away from the vehicle speed, large traction force will be calculated, and the model will potentially be numerically unstable for prediction of the sigma point leading to a bad approximation of the mean and covariance of the state.

The particle swarm optimization was chosen for optimizing the parameters in the full filter model. This is not assumed to contribute to the estimation error because the estimation error was consistent with the performed sensitivity analysis, meaning that the optimization algorithm did its job and found the local minimum.

The computation time required to perform the parameter estimation is problematic, as it makes design iterations takes an unreasonable amount of time. For the 8 parameters 100 iterations of the particle swarm optimization algorithm was determined to be sufficient to find the minimum of the negative log likelihood criterion with 10 particles per parameter. Including the initial iterations this is 8080 evaluations of the criterion. For estimating larger sets of parameters, this increases dramatically. Gradient based optimization techniques were not tested extensively, but when tested, required criterion evaluations in the order of thousands to find a local minimum, as the gradient has to be estimated numerically. As many gradient based optimization has to be performed to confirm a global minimum, particle swarm optimization was found to be significantly quicker.

The decision of which parameters should be estimated was primarily based of which parameters could be easily decided to a reasonable amount of accuracy from the vehicle specifications and empirical observations of the vehicle behaviour. To make the parameter estimation take a reasonable amount of time, more of the parameters had to be assumed to be fixed. This resulted in a lot of guesswork related to the parameters. This has without a doubt also impacted the performance of the final model. Even though a parameter sensitivity analysis was performed, it was done late in the process, and therefore not used to chose which parameters should be estimated. Although it did conform reasonably well to the final set of parameters which was chosen to be estimated.

8.3 Further work

As the speed of the parameter estimation has been a hindrance, it is recommended to start further work with the problem by trying to improve the performance of the parameter estimation algorithm. The discretization of the model is a good place to start. It is currently based on fixed step Runge Kutta. A variable time step solver would probably be better suited, as mean time step needed would be decreased. This would also make the discretization handle the slight stiffness of the model better. As steps sizes longer than the time period between the measurements will likely be possible with a variable step solver. It should therefore be considered if the measurement frequency can be reduced from the current 40 Hz to reduce the computation time.

A better approximation of the suspension in the reduced coast down filter model has to be implemented to remove its estimation bias. As the engine and CVT are believed to be the largest sources of error in the vehicle, these have to be investigated further. If an improved discretization scheme does not improve the numerical stability of the model a different tire model has to be considered.

Which parameter should be estimated and which can be set to fixed values should also be investigated further. A more rigorous sensitivity analysis would likely be a good place to start. Potentially also a second order sensitivity analysis to determine any interdependence of the parameters.

It also has to be considered whether some of the parameters in the model should be found using a different method than the proposed parameter estimation. It would for example be beneficial to have accurate measurements of the torque or power produced by the engine at different throttle setting and engine speeds, which can be made by dynamometer. The fixed

suspension offset may also possibly be estimated more accurately by placing the vehicle on a known level surface and measuring the reported pitch of the vehicle.

As the model with the currently estimated parameters is inaccurate when used for simulation, it is therefore likely not very useful for control system development as is. Used for state estimation on the other hand, it seems to work reasonably well. Possible uses are online estimation of some of the parameters that are likely to change, by augmenting the state with the parameters. Potential candidates are the coefficient of friction of the tires, rolling resistance and dynamic tire radius. The tire friction can be used to decide how aggressive the vehicle can be driven, while changes in rolling resistance and dynamic radius of the tires can be used to determine if one or more of the tires have been punctured.

Chapter 9

Conclusion

All the goals presented in the introduction are achieved, although to varying degrees of success:

A. A model of the longitudinal dynamics of the vehicle is developed and presented. The model is based on classical mechanics as well approximations of proven vehicle dynamics modeling techniques. The drive line was modeled using bond graph modeling resulting in a new type of CVT model. Without direct comparison to measurements of the vehicle, it seems to give a realistic representation of the vehicles dynamics.

B. A parameter estimation algorithm is proposed and implemented. The algorithm is based on maximum likelihood estimation, using optimization of the model parameters with the likelihood of the observations as calculated by a UKF based state estimator. Two different filter models and experiments are used to reduce the number of parameters that has to be estimated simultaneously.

C. A UKF based state estimator was developed as part of the parameter estimation algorithm. It manages to observe all states when applied to simulated data, but this was during ideal conditions with the ground inclination being the only process nnoise, and its performance have to be investigated further. It also gave what appears to be a good estimate of the vehicle state when tested on measurements from the vehicle.

D. The parameter estimation algorithm was verified to work on simulated data, although it did not manage to find the exact parameters. The simulations of the model with estimated parameters was compared to measurements of the vehicle. The model was not found to be satisfactory in its current state, but state estimation using the model appears to work well.

Unfortunately the resulting model with the estimated parameters did not give satisfactory results. This is partially believed to be caused by the parameter estimation algorithm, as it failed to give optimal results even for simulated data. The biased parameter estimate is believed to be caused by inaccuracies in the reduced model used to estimate parameters decoupled from the drive line, as well as numerical instability during discretization of the model. Estimations from longer data series is also believed to improve the results, but this was not investigated because of computation time of current implementation of the parameter estimation algorithm. The estimation bias is not believed to be caused by the particle swarm optimization algorithm used to optimize the parameters, as the estimation bias was consistent with the performed sensitivity analysis. It is also a possibility that it is caused by implementation errors.

The sub optimal result is also believed to be caused by errors in the developed model. Due to the time it took to derive and implement the presented model and parameter estimation scheme, the errors in the model is not throughly investigated. The largest sources of error in the model is assumed to be the engine torque model and the CVT model, and these therefore has to investigated further. Due to the stiffness of the used tire model, an alternative model may be beneficial.

The choice of fixed and estimated parameters can not be ruled out as a possible source of the model inaccuracy. Due to the long processing time the parameter estimation algorithm required, the number of estimated parameters is smaller than what was desired. This lead to a lot of guesswork when deciding values for the fixed parameters. A thorougher parameter sensitivity analysis should be performed to decide which parameters should be estimated. A second order sensitivity analysis could be of benefit to investigate the interdependence of the parameters.

The long computation time for both the state estimator and the parameter estimation algorithm has been a problem as it considerably slowed down the development of the model and parameter estimation algorithm. Faster and more efficient implementations of these are therefore highly desirable as it would allow for quicker design iterations.

In its current form, the vehicle model with the estimated parameters does not match the observed vehicle dynamics sufficiently to be useful for developing a better control system. The overall model structure and parameter estimation algorithm is nevertheless believed to be a good approach and have potential to give good results if worked on further.

Appendices

Appendix A

MATLAB code

This appendix presents the MATLAB implementations of the vehicle model, filter models, the UKF, a script for testing the UKF, and the parameter estimation algorithm. Scrips used to plot figures and doing simple calculations, like weighted mean square error for tuning α and the sensitivity analysis, are not included as their implementation are straightforward. These scripts can be provided by the author on request if they should be of interest.

A.1 MATLAB implementation of the filter and simulation models

The MATLAB implementation of the state space model of OLAV, described in chapter 4, is shown in listing A.1. The state transition function and the measurement functions are implemented as nested functions, in a function that sets up all parameters and generates a structure containing handles to f and h as well as other properties needed to use the model. The function implements both the simulation and filter model for the full vehicle model. The models are returned as two separate structures. Listing A.2 on page 85 shows the implementation of the reduced filter model used for the coast down experiment.

Listing A.3 on page 86 contains the assumed true parameters and returns a structure containing them. The function shown in listing A.4 on page 87 is used to modify the parameter structure using an array of parameter names and an array of parameter values. The function shown in listing A.5 on page 87 implements the fourth order Runge Kutta integration used for discretization of the model. The models and Runge Kutta function are implemented to work on vectorized inputs. This means that the UKF can evaluate all the sigma points with using one function call.

Listing A.1: MATLAB implementation of the full mathematical state space model of the vehicle dynamics.

```
1  function [stochastic_model, deterministic_model] = Model(parameter_names,
       parameter_values)
2      % Model dimensions
3      deterministic_model.Nx = 12;
4      deterministic_model.Nu = 5;
5      deterministic_model.Nz = 4;
6      stochastic_model.Nx = deterministic_model.Nx + 2;
7      stochastic_model.Nu = deterministic_model.Nu - 2;
8      stochastic_model.Nz = 5;
9
10     % Names
11     deterministic_model.state_names      = {'$\omega_e$','$\frac{d\phi_p}{dt}$'
           ,'$\phi_p$','$\rho_{bk}$','$\omega_{fw}$','$\omega_{rw}$','$F_{fwx}$',
           '$F_{rwx}$','$\omega_v$','$\theta_v$','$v_x$','$r_x$'};
12     deterministic_model.input_names      = {'Throttle','Brake','Gear','$\
           theta_g$', '$\frac{d\theta_g}{dt}$'};
13     deterministic_model.measurement_names = {'$\omega_e$','$\omega_rw$','$v_x$
           ','$\theta_v$'};
14     stochastic_model.state_names        = [deterministic_model.state_names,'
           $r_z$','$\theta_g$'];
```

```
15      stochastic_model.measurement_names = {'$\omega_e$','$\omega_{rw}$','$v_x$'
            ,'$r_z$','$\theta_v$'};
16      stochastic_model.input_names       = deterministic_model.input_names(1:3);
17
18      % Noise covariances
19      stochastic_model.Q = diag([1e+2, 1, 1e-2, 1e-2, 1e-2, 1e-2, 1e+2, 1e+2, 1e
            -3, 1e-4, 1, 1e-3, 1e-3, 1e-4]);
20      stochastic_model.R = diag([5e+1, 5e-2, 1e-3, 1e-1, 1e-5]);
21
22      deterministic_model.R = diag([5e+1, 5e-2, 1e-3, 1e-5]);
23
24      % Load constants
25      p = Parameters(parameter_names,parameter_values);
26      stochastic_model.p    = p;
27      deterministic_model.p = p;
28
29      % Calculate constant values
30      power_coefficients  = p.P_e_max.*[1/p.omega_e_P_max, 1/p.omega_e_P_max^2,
            -1/p.omega_e_P_max^3];
31      T_e_idle            = power_coefficients*[1; p.omega_e_idle; p.
            omega_e_idle.^2];
32      transmission_ratios = [0, p.R_R, 0, p.R_L, p.R_H];
33      c_sw                = p.k_cvt/p.u_cvt;
34      c_tr                = p.k_cvt/p.d_cvt;
35      m_v                 = p.m_ve + p.m_vp;
36      F_g                 = m_v*9.81;
37
38      % Stochastic process model
39      function dxdt = f(x,u,v,type)
40          M = size(x,2);
41
42          if strcmp(type,'filter')
43              filter = true;
44          else
45              filter = false;
46          end
47
48          omega_e             = x(1,:);
49          pulley_speed        = x(2,:);
50          pulley_displacement = x(3,:);
51          brake_pressure      = x(4,:);
52          omega_fw            = x(5,:);
53          omega_rw            = x(6,:);
54          F_fwx               = x(7,:);
55          F_rwx               = x(8,:);
56          pitch_speed         = x(9,:);
57          pitch               = x(10,:);
58          v_x                 = x(11,:);
59          r_x                 = x(12,:);
60
61          if filter == true
62              r_z                 = x(13,:);
63              inclination         = x(14,:);
64              inclination_speed   = (abs(v_x) + 1e-1).*v(14,:);
65              dxdt                = zeros(14,M);
66              dxdt(13:14,:)       = [v_x.*sin(inclination) + v(13,:) ;...
67                                     inclination_speed                ];
68          else
69              inclination       = u(4);
70              inclination_speed = u(5)*v_x.*cos(inclination);
71          end
72
73          throttle = u(1);
74          brake    = u(2);
75          gear     = u(3);
76
77          % Speed as measured by wheel sensor
78          v_fw = p.r_eff.*omega_fw;
79          v_rw = p.r_eff.*omega_rw;
80          slip_fw = (v_fw - v_x)./max(max(abs(v_fw),abs(v_x)),1);
81          slip_rw = (v_rw - v_x)./max(max(abs(v_rw),abs(v_x)),1);
82
```

```
83          % Idle controller, engine speed limiter, and vehicle speed limiter
84          throttle_idle = (omega_e<p.omega_e_idle).*(p.b_e*p.omega_e_idle/(
                T_e_idle + p.b_e*p.omega_e_idle) + p.p_idle.*(p.omega_e_idle -
                omega_e));
85          throttle = (omega_e<p.omega_e_max).*(v_x < p.v_max).*throttle + (
                throttle - v_x + p.v_max).*(v_x >= p.v_max);
86          throttle = throttle.*(throttle>throttle_idle) + throttle_idle.*(
                throttle<=throttle_idle);
87          throttle = min(max( throttle, 0), 1);
88
89          % Engine torque
90          T_engine = throttle.*(power_coefficients(1) + power_coefficients(2).*
                omega_e + power_coefficients(3).*omega_e.^2) - p.b_e.*(1-throttle)
                .*omega_e;
91
92          % Transmission
93          R_t = transmission_ratios(round(gear));
94          I_dl = p.I_rw + R_t.^2.*p.I_t; % Combined inertia of front wheels,
                rear wheels, and transmission.
95
96          % CVT:
97          R_cvt     = p.R_cvt_D - (p.R_cvt_D - p.R_cvt_U).*(pulley_displacement);
98          R_cvt     = min(max(R_cvt,p.R_cvt_U),p.R_cvt_D);
99          sw        = (omega_e>p.omega_engagement).*(omega_e - p.omega_engagement
                );
100         omega_b   = R_cvt.*R_t.*omega_rw ;
101         slip_b    = (omega_e - omega_b)./max(abs(omega_e),abs(omega_b));
102         T_belt    = p.mu_bm.*sw.*tanh(slip_b./(p.s_bm/2));
103         F_spring = -(p.k_cvt.*(0 <= pulley_displacement) + p.k_cvt*10.*(
                pulley_displacement < 0)).*pulley_displacement...
104                 - p.k_cvt*10.*(1 < pulley_displacement).*(
                        pulley_displacement - 1);
105         F_sw      = c_sw.*sw;
106         F_tr      = - c_tr.*R_cvt.*T_belt;
107
108         % Calculate forces from gravity
109         F_gx = -sin(inclination).*F_g;
110         F_gz = -cos(inclination).*F_g;
111
112         % Suspension pitch torque and tire normal forces
113         T_s_max = - F_gz.*p.r_r;
114         T_s_min = F_gz.*p.r_f;
115         T_s     = p.k_s.*(inclination + p.theta_v0 - pitch) + p.b_s.*(
                inclination_speed - pitch_speed);
116         T_s     = max( min( T_s, T_s_max), T_s_min );
117
118         % Tire forces
119         F_fwz         = (T_s - F_gz.*p.r_r)./(p.r_f + p.r_r);
120         F_rwz         = - (F_gz + F_fwz);
121         F_fwx_static = p.mu_tm.*tanh((2/p.s_tm).*slip_fw).*F_fwz;
122         F_rwx_static = p.mu_tm.*tanh((2/p.s_tm).*slip_rw).*F_rwz;
123
124         % Force from air resistance
125         F_d = -p.b_ar*sign(v_x).*v_x.^2;
126
127         % Axle torques
128         sign_omega_fw = tanh(omega_fw);
129         sign_omega_rw = tanh(omega_rw);
130         T_fw = - p.b_rr.*sign_omega_fw.*F_fwz...
131                 - p.T_bk.*p.B_bk.*sign_omega_fw.*brake_pressure.*(
                        brake_pressure > 0);
132         T_rw = + R_t.*(R_cvt.*T_belt - p.b_t.*R_t.*omega_rw)...
133                 - p.b_rr.*sign_omega_rw.*F_rwz...
134                 - p.T_bk.*(1-p.B_bk).*sign_omega_rw.*brake_pressure.*(
                        brake_pressure > 0);
135
136         % Calculate derivative of x
137         dxdt(1:12,:) = ...
138             [(1/p.I_e).*( T_engine - T_belt)                          ;...
139              ( F_sw - p.b_cvt.*pulley_speed + F_spring + F_tr )       ;...
140              pulley_speed                                             ;...
141              (1/p.tau_bk).*( brake - brake_pressure )                 ;...
```

```
142             (1/p.I_fw).*( T_fw - p.r_dyn.*(F_fwx) )                 ;...
143             (1./I_dl  ).*( T_rw - p.r_dyn.*(F_rwx) )                ;...
144             (1/p.tau_t).*(F_fwx_static - F_fwx)                     ;...
145             (1/p.tau_t).*(F_rwx_static - F_rwx)                     ;...
146             (1/p.I_v_pitch).*( (F_fwx + F_rwx).*p.r_z + T_s )       ;...
147             pitch_speed                                             ;...
148             (1/m_v).*( F_fwx + F_rwx + F_d + F_gx )                 ;...
149             cos(inclination).*v_x                                  ];
150
151         % Add noise
152         if filter == true
153             dxdt(1:12,:) = dxdt(1:12,:) + v(1:12,:);
154         end
155     end
156
157     % Measurement equation
158     function z = h(x)
159         omega_e  = x(1,:);
160         omega_rw = x(6,:);
161         pitch    = x(10,:);
162         v_x      = x(11,:);
163         r_z      = x(13,:);
164
165         z = [omega_e; abs(omega_rw); v_x ; r_z; pitch];
166     end
167
168     % Initial condition function
169     function x0 = x0_fun(z0,u0)
170         omega_e  = z0(1);
171         omega_rw = z0(2);
172         v_x      = z0(3);
173         r_z      = z0(4);
174         pitch    = z0(5);
175
176         brake    = u0(2);
177
178         x0 = [omega_e; 0; 0; brake; omega_rw; omega_rw; 0; 0; 0; pitch; v_x;
                0; r_z; pitch - p.theta_v0];
179     end
180
181     % Add function handles to model structure
182     deterministic_model.f = @(x,u) f(x,u,[],'simulation');
183     deterministic_model.h = @(x,u) x([1,6,11,10],:);
184
185     stochastic_model.f      = @(x,u,v) f(x,u,v,'filter');
186     stochastic_model.fd     = @(x,u,v,dt) RK4(@(x) f(x,u,v,'filter'), x, dt,
            ceil(dt*200));
187     stochastic_model.h      = @h;
188     stochastic_model.x0_fun = @x0_fun;
189
190     % Generate a consistent color scheme for plotting
191     Nxc = max(stochastic_model.Nx, deterministic_model.Nx);
192     Nuc = max(stochastic_model.Nu, deterministic_model.Nu);
193     Nzc = stochastic_model.Nz;
194     if exist('distinguishable_colors','file')
195         % Use distinguishable_colors if available
196         state_colors       = distinguishable_colors(Nxc);
197         input_colors       = distinguishable_colors(Nuc);
198         measurement_colors = distinguishable_colors(Nzc);
199     else
200         state_colors       = hsv(Nxc);
201         input_colors       = hsv(Nuc);
202         measurement_colors = hsv(Nzc);
203     end
204
205     % Store colors in structure
206     deterministic_model.state_colors     = state_colors([(1:7),(9:10)],:);
207     deterministic_model.input_colors     = input_colors((1:deterministic_model.
            Nu),:);
208     stochastic_model.state_colors        = state_colors((1:stochastic_model.Nx)
            ,:);
209     stochastic_model.input_colors        = input_colors((1:stochastic_model.Nu)
```

```
        ,:);
210     stochastic_model.measurement_colors = measurement_colors((1:
            stochastic_model.Nz),:);
211 end
```

Listing A.2: MATLAB implementation of the reduced coast down filter model.

```
1  function ms = ModelCD(parameter_names,parameter_values)
2      % Model dimensions
3      ms.Nx = 4;
4      ms.Nu = 0;
5      ms.Nz = 4;
6
7      % Names
8      ms.state_names       = {'$v_x$','$r_x$','$r_z$','$\theta$'};
9      ms.measurment_names = {'$\omega_w$','$v_x$','$r_z$','$\theta_v$'};
10     ms.input_names       = {};
11
12     ms.dt_max = 1/200;
13
14     % Noise covariance
15     ms.Q = diag([1e-3, 1e-3, 1e-3, 1e-4]);
16     ms.R = diag([3e-2, 1e-3, 1e-2, 1e-3]);
17
18     % Load constants
19     p = Parameters(parameter_names,parameter_values);
20     ms.p = p;
21
22     % Calculate constant values
23     m_v = p.m_ve + p.m_vp;
24     I_w = p.I_fw + p.I_rw;
25     F_g = m_v.*9.81;
26     combined_inertia = m_v + I_w/p.r_eff^2;
27
28     % Process differential equation
29     function dxdt = f(x,u,v)
30         v_x          = x(1,:);
31         r_x          = x(2,:);
32
33         if isempty(v)
34             inclination = u(1,:);
35         else
36             inclination = x(4,:);
37         end
38
39         % Gravity
40         F_gx = -sin(inclination).*F_g;
41         F_gz = -cos(inclination).*F_g;
42
43         % Air resistance
44         F_D = -p.b_ar.*sign(v_x).*v_x.^2;
45
46         % Rolling resistance
47         omega_w = v_x./p.r_eff;
48         F_rr = (-sign(omega_w)).*p.b_rr.*( -F_gz )./p.r_dyn;
49
50         % Differential
51         if isempty(v)
52             dxdt = [1/(combined_inertia).*(F_gx + F_D + F_rr);...
53                     v_x.*cos(inclination)];
54         else
55             dxdt = [1/(combined_inertia).*(F_gx + F_D + F_rr) + v(1,:);...
56                     v_x.*cos(inclination) + v(2,:);...
57                     v_x.*sin(inclination) + v(3,:);...
58                     (abs(v_x) + 0.1).*v(4,:) ];
59         end
60     end
61
62     % Measurement equation
63     function z = h(x)
64
65         v_x          = x(1,:);
```

```
66          r_z         = x(3,:);
67          inclination = x(4,:);
68          omega_w     = v_x./p.r_eff;
69
70
71          pitch = inclination + p.theta_v0;
72
73          z = [omega_w; v_x; r_z; pitch];
74      end
75
76      % Initial condition function
77      function x0 = x0_fun(z0,u0);
78          omega_w = z0(1);
79          v_x     = z0(2);
80          r_z     = z0(3);
81          pitch   = z0(4);
82
83          x0 = [(v_x + omega_w./p.r_eff)./2; 0; r_z; pitch - p.theta_v0];
84      end
85
86      % Add function handles to model structure
87      ms.f       = @f;
88      ms.fd      = @(x,u,v,dt) RK4( @(x) f(x,u,v), x, dt, ceil(dt*200));
89      ms.h       = @h;
90      ms.x0_fun = @x0_fun;
91      ms.f_deterministic = @(x,u) f(x,u,[]);
92
93      % Generate a consistent color scheme for plotting
94      if exist('distinguishable_colors','file')
95          ms.state_colors = distinguishable_colors(ms.Nx);
96          ms.measurement_colors = distinguishable_colors(ms.Nz);
97          ms.input_colors = distinguishable_colors(ms.Nu);
98      else
99          ms.state_colors = hsv(ms.Nx);
100         ms.measurement_colors = hsv(ms.Nz);
101         ms.input_colors = hsv(ms.Nu);
102     end
103 end
```

Listing A.3: MATLAB function that returns a structure with the default model parameters.

```
1   function d = OlavDefaultParameters()
2       % Conversion functions
3       rpm_to_radps = @(rpm) rpm/60*2*pi;
4       mph_to_mps   = @(mph) mph/(60*60)*1609.34;
5       kmph_to_mps  = @(kmh) kmh/3.6;
6       inch_to_m    = @(inch) inch.*0.0254;
7       hp_to_kw     = @(hp) hp*745.7;
8
9       % Calculate weight distribution
10      wheelbase = 2.05;
11      m_middle = 1040;
12      m_front_axle = 400;
13      m_rear_axle = 640;
14      m_unsprung = 150;
15      CG_to_rear_axle = (m_front_axle/m_middle + (1-m_rear_axle/m_middle))/2*
            wheelbase;
16      CG_above_ground = 0.60;
17
18      % Vehicle mass and geometry
19      d.m_ve = m_middle;
20      d.m_vp = 160;
21      d.r_f = wheelbase - CG_to_rear_axle;
22      d.r_r = CG_to_rear_axle;
23      d.r_z = CG_above_ground;
24      d.I_v_pitch = (d.m_ve)*(wheelbase/2)^2;
25      d.k_s = 5.0e+4;
26      d.b_s = 1.0e+4;
27      d.theta_v0 = -5.0e-2;
28      d.b_ar = 2.0;
```

```
29
30      % Engine parameters
31      d.I_e = 10*0.20^2/2;
32      d.P_e_max = hp_to_kw(45);
33      d.b_e  = 1.0e-2;
34      d.omega_e_idle = rpm_to_radps(1250);
35      d.omega_e_P_max = rpm_to_radps(7000);
36      d.omega_e_max = rpm_to_radps(7250);
37      d.v_max = mph_to_mps(25);
38      d.p_idle = 1e-3;
39
40      % CVT and transmission parameters
41      d.omega_engagement = rpm_to_radps(2000);
42      d.R_cvt_D = 3.83;
43      d.R_cvt_U = 0.76;
44      d.R_R = -22.92;
45      d.R_L = 25.59;
46      d.R_H = 10.40;
47      d.u_cvt = 300;
48      d.d_cvt = 200;
49      d.k_cvt = 1.0e+2;
50      d.b_cvt = 5.0e+1;
51      d.mu_bm = 0.5;
52      d.s_bm = 0.20;
53      d.I_t = (5*0.10^2)/2;
54      d.b_t = 0;
55
56      % Wheel and tire
57      d.r_dyn = 0.33;
58      d.r_eff = 0.35;
59      d.I_fw  = 2*20*0.33^2/2 + 2;
60      d.I_rw  = 2*25*0.33^2/2 + 2;
61      d.mu_tm = 0.7;
62      d.s_tm  = 0.15;
63      d.tau_t = 0.20;
64      d.b_rr  = 2.0e-2;
65
66      % Brakes
67      d.T_bk = 5.0e+3;
68      d.tau_bk = 0.50;
69      d.B_bk = 0.6;
70  end
```

Listing A.4: MATLAB function for changing the model parameter structure using name and value pairs.

```
1   function p = parameters(names,values)
2       % Set default values
3       p = OlavDefaultParameters();
4
5       % Update values
6       if not(isempty(names))
7           for i = 1:numel(names)
8               p = setfield(p, names{i}, values(i));
9           end
10      end
11  end
```

Listing A.5: MATLAB function for Runge Kutta integration.

```
1   function x = RK4(f,x,dt,iterations)
2       dt = dt/iterations;
3       for k = 1:iterations
4           k1 = f(x);
5           k2 = f(x + k1.*(dt/2));
6           k3 = f(x + k2.*(dt/2));
7           k4 = f(x + k3.*dt);
8           x  = x + (k1 + 2.*k2 + 2.*k3 + k4).*(dt/6) ;
9       end
10  end
```

A.2 Batch processing UKF

The implemented UKF is for discrete time non-linear time invariant state space model with non additive process noise and additive measurement noise. Noise is assumed Gaussian and time invariant. The filter performs batch processing and calculates the log likelihood of the estimate in addition to the state estimate. The filter model get passed to the UKF as a structure with field containing, f, h, Q, and R, as well as the dimensions of the state, input, and measurement vectors. The f and h function are assumed to handle vectorized arguments meaning that all sigma points can be evaluated in one function call of f and h.

Listing A.6: MATLAB function for batch processing UKF with recursive log likelihood calculation.

```
function [t,x_apost,P_apost,log_likelihood,MSE] = ukf(t,z,u,x0,P0,model,alpha,
    beta,kappa)
    % Dimensions.
    M   = length(t);
    Nx  = model.Nx;
    Nz  = model.Nz;
    Nu  = model.Nu;
    Nxa = Nx+Nx;
    Ni  = 2*Nxa+1;

    % Sigma point weights
    lambda = alpha^2*(Nxa + kappa) - Nxa;
    Wm = [lambda/(Nxa + lambda); 1/(2*(Nxa + lambda)).*ones(Ni-1,1)];
    Wc = [lambda/(Nxa + lambda) + (1 - alpha^2 + beta); 1/(2*(Nxa + lambda)).*
        ones(Ni-1,1)];
    c   = sqrt(Nxa + lambda);

    % Result matrices
    x_apost = zeros(Nx,M);
    P_apost = zeros(Nx,Nx,M);
    z_apri  = zeros(Nz,M);

    % Initialize log likelihood with the constant part
    log_likelihood = -M*Nz*log(2*pi);

    % Initial augmented state vector and covariance matrix
    Xa = [x0; zeros(Nx,1)];
    Pa = blkdiag(P0, model.Q);
    dt = 0;

    % Form initial sigma points
    U  = chol(Pa)';
    Xi = repmat(Xa,[1,Ni]) + c*[zeros(Nxa,1), U, -U];

    % Run M iterations of UKF.
    for k = 1:M
        % Update sigma points with a posteriori state and covariance estimate
        % from previous time step.
        if k > 1
            dt              = t(k) - t(k-1);
            Xa(1:Nx)        = x_apost(:,k-1);
            U(1:Nx,1:Nx)    = chol(P_apost(:,:,k-1))';
            Xi              = repmat(Xa,[1,Ni]) + c*[zeros(Nxa,1), U, -U];
        end

        % Predict sigma points through process and measurement model.
        if dt ~= 0
            % Fetch input for current time step if model includes inputs
            if model.Nu == 0
                uk = [];
            else
                uk = u(:,k);
            end
            % Predict using the discrete function provided by the model
            Xi(1:Nx,:) = model.fd( Xi(1:Nx,:), uk, Xi((Nx+1):(Nx+Nx),:), dt);
        end
        Zi = model.h(Xi(1:Nx,:));
```

```
56
57          % Sum sigma points to a priori estimates of state and measurement
                vector.
58          x_apri       = Xi(1:Nx,1:Ni)*Wm;
59          z_apri(:,k) = Zi*Wm;
60
61          % Calculate covariances from sigma points and predicted state and
62          % measurement vectors.
63          P_apri = zeros(Nx,Nx);
64          Pzz    = model.R;
65          Pxz    = zeros(Nx,Nz);
66
67          for i = 1:Ni
68              P_apri = P_apri + Wc(i)*( ( Xi(1:Nx,i) - x_apri       )*( Xi(1:Nx,i
                    ) - x_apri       )' );
69              Pzz    = Pzz    + Wc(i)*( ( Zi(1:Nz,i) - z_apri(:,k) )*( Zi(1:Nz,i
                    ) - z_apri(:,k) )' );
70              Pxz    = Pxz    + Wc(i)*( ( Xi(1:Nx,i) - x_apri       )*( Zi(1:Nz,i
                    ) - z_apri(:,k) )' );
71          end
72
73          % Calculate Kalman gain.
74          K_k = Pxz/Pzz;
75
76          % Innovation
77          dz = z(:,k) - z_apri(:,k);
78
79          % Update state and covariance estimates.
80          x_apost(:,k)   = x_apri + K_k*dz;
81          P_apost(:,:,k) = P_apri - K_k*Pzz*K_k';
82
83          % Update log-likelihood
84          log_likelihood = log_likelihood - log(det(Pzz)) - dz'*inv(Pzz)*dz;
85      end
86
87      % Divide log likelihood by two
88      log_likelihood = log_likelihood/2;
89  end
```

A.3 Script for testing the UKF and full vehicle filter model

The script shown in listing A.7 runs a Monte Carlo simulation of the UKF using the full filter model. The result of the simulation was used to see if the UKF was well behaved. The simulated system response is generated with the same noise as assumed in the filter model and with a couple of randomly placed throttle and brake input steps. The script does not create plots that are well suited for inclusion in this report, but examples are shown in fig. A.1 on page 92 and fig. A.2 on page 93. The generated plots are only from 10 realizations.

Listing A.7: MATLAB script for testing the UKF and the model.

```
1  % Generate functions from symbolic model
2  total_timer = tic;
3
4
5  % True parameters
6  param       = [   3.0e+4; 2.00e-2; 4.00e+2;  3.00e+2; 0.50e+0;  0.8e+0; 4.09e
       +3;  1.50e+0];
7  param_names = {'P_e_max',   'b_e',  'u_cvt', 'd_cvt', 'mu_bm', 'mu_tm',  'T_bk
       ', 'tau_bk'};
8  param_cd = [0.356; 1.58e-2; 2.3; -4.47e-2];
9  param_cd_names = {'r_eff','b_rr','b_ar','theta_v0'};
10
11 alpha = 5e-2;
12 beta = 2;
13 k = 0;
14
15 fprintf('Generating model...\n')
```

```
16 tic
17 model = Model([param_names,param_cd_names],[param;param_cd]);
18 toc
19
20 % Initial Condition
21 inclination_0 = -0.05;
22 x0 =       [model.p.omega_e_idle; 0;     0;      0.5;     0;     0;     0;     0;
       0; inclination_0 + model.p.theta_v0;     0;     0;     0; inclination_0]
23 P0 = diag([                  1e+1, 1e-2, 1e-2, 1e-2, 1e-2, 1e-2, 1e-2, 1e-2, 1e
       -2,                                1e-2, 1e-3, 1e-3, 1e-3,           1e-3]);
24
25 % Simulation time parameters
26 fs = 500; % State simulation frequency
27 fm = 40; % Measurement frequency
28 t0 = 0; % Start time
29 t1 = 60; % Final time
30
31
32 Mz = (t1-t0)*fm+1;
33 Mmc = 10;
34
35 t = linspace(t0,t1,Mz);
36
37 e = zeros(model.Nx,Mz,Mmc);
38 P = zeros(model.Nx,model.Nx,Mz);
39
40 Q = diag([1e+2, 1e-1, 1e-2, 1e-2, 1e-3, 1e-3, 1e+2, 1e+2, 1e-3, 1e-9, 1e-3, 1e
       -9, 1e-9, 1e-4])
41 R = diag([1e+1, 1e-1, 1e-3, 1e-1, 1e-4]);
42
43 fprintf('Running Monte Carlo simulation (%d iterations)...\n',Mmc);
44 tic()
45 ukf_time = 0;
46 loglikelihood = zeros(Mmc,1);
47 for i = 1:Mmc
48     % Time dependant forcing function
49     tu = sort(t0 + (t1-t0).*rand(8,1));
50     throttle = 0.2 + 0.8*rand(2,1);
51     brake = 0.2 + 0.8*rand(2,1);
52     gear = 4 + round(rand(1));
53     u  = @(t) [throttle(1)*(t>tu(1)).*(tu(2)>t) + throttle(2)*(t>tu(5)).*(tu
           (6)>t); 0.5.*(t<max(tu(1)-2,0)) + brake(1)*(t>tu(3)).*(tu(4)>t) +
           brake(2)*(t>tu(7)).*(tu(8)>t); gear*ones(1,numel(t))];
54     uk = u(t);
55
56     % Pre allocate
57     x_sim = zeros(model.Nx,Mz);
58     z_sim = zeros(model.Nz,Mz);
59
60     % Generate noise for simulation
61     V = chol(model.Q)*randn(model.Nx,Mz);
62     W = chol(model.R)*randn(model.Nz,Mz);
63
64     % Draw initial state
65     x_sim(:,1) = x0 + chol(P0,'lower')*randn(model.Nx,1);
66
67     for k = 2:Mz
68         x_sim(:,k) = model.fd(x_sim(:,k-1), uk(:,k), V(:,k), 1/fm);
69     end
70     z_sim = model.h(x_sim) + W;
71
72     tic()
73     [~,x_est,P_est,loglikelihood(i)] = ukf(t,z_sim,uk,x0,P0,model,alpha,beta,k
           );
74     ukf_time = ukf_time + toc();
75
76     if i == 1
77         figure(1)
78         clf
79         spax(1) = subplot(4,1,1);
80         spax(2) = subplot(4,1,2);
81         spax(3) = subplot(4,1,3:4);
```

```
82          spax(1).ColorOrder = model.input_colors;
83          spax(1).NextPlot = 'replacechildren';
84          input_plot = plot(spax(1),t,uk);
85          grid on
86          spax(1).Title.String = 'Input';
87          legend(spax(1),model.input_names)
88          spax(2).ColorOrder = model.measurement_colors;
89          spax(2).NextPlot = 'replacechildren';
90          measurement_plot = plot(spax(2),t,z_sim);
91          grid on
92          spax(2).Title.String = 'Measurements';
93          legend(spax(2),model.measurement_names);
94          spax(3).ColorOrder = model.state_colors;
95          spax(3).NextPlot = 'replacechildren';
96          state_sim_plot = plot(spax(3),t,x_sim,':');
97          hold on;
98          state_est_plot = plot(spax(3),t,x_est);
99          grid on
100         spax(3).Title.String = 'State simulation and estimate';
101         legend(spax(3),[model.state_names,model.state_names]);
102         hold off;
103         linkaxes(spax,'x');
104     else
105         set(input_plot,        {'YData'}, num2cell(uk,2));
106         set(measurement_plot, {'YData'}, num2cell(z_sim,2));
107         set(state_sim_plot,    {'YData'}, num2cell(x_sim,2));
108         set(state_est_plot,    {'YData'}, num2cell(x_est,2));
109     end
110     drawnow;
111
112     e(:,:,i) = x_est - x_sim;
113     P = P + P_est/Mmc;
114 end
115 toc()
116
117 ukf_faster_than_RT_percent = ((t1-t0))/(ukf_time/Mmc)*100;
118 mean_loglikelihood = mean(loglikelihood);
119 fprintf( 'UKF filter runs %.1f%% faster than real time\n',
        ukf_faster_than_RT_percent);
120 fprintf( 'Mean log likelihood estimated to: %.2e\n', mean_loglikelihood);
121
122 fprintf('Calculating mean and covariance of estimation error...\n')
123 tic()
124 mean_mc = zeros(model.Nx,Mz);
125 cov_mc  = zeros(model.Nx,model.Nx,Mz);
126 Smc     = zeros(model.Nx,Mz);
127 S       = zeros(model.Nx,Mz);
128
129 for k = 1:Mz
130     mean_mc(:,k)   = mean(e(:,k,1:2),3);
131     cov_mc(:,:,k)  = (e(:,k,1) - mean_mc(:,k))*(e(:,k,1) - mean_mc(:,k))' + (e
            (:,k,2) - mean_mc(:,k))*(e(:,k,2) - mean_mc(:,k))';
132     mean_mc(:,k) = mean(e(:,k,:),3);
133
134     for i = 3:Mmc
135         cov_mc(:,:,k) = cov_mc(:,:,k) + 1/(Mmc-1)*(e(:,k,i) - mean_mc(:,k))*(e
                (:,k,i) - mean_mc(:,k))';
136     end
137     Smc(:,k) = abs(sqrt(diag(cov_mc(:,:,k))));
138     S(:,k)   = abs(sqrt(diag(P(:,:,k))));
139 end
140 toc()
141
142 fprintf('Plotting result...\n');
143 figure(2)
144 clf
145 for i = 1:model.Nx
146     spax(i) = subplot(ceil(model.Nx/3),3,i);
147     plot(spax(i),t,mean_mc(i,:),t,Smc(i,:),'r:',t,S(i,:),'r--',t,-Smc(i,:),'r:
            ',t,-S(i,:),'r--');
148     ylabel(model.state_names(i));
149     xlabel('$t$/s')
```

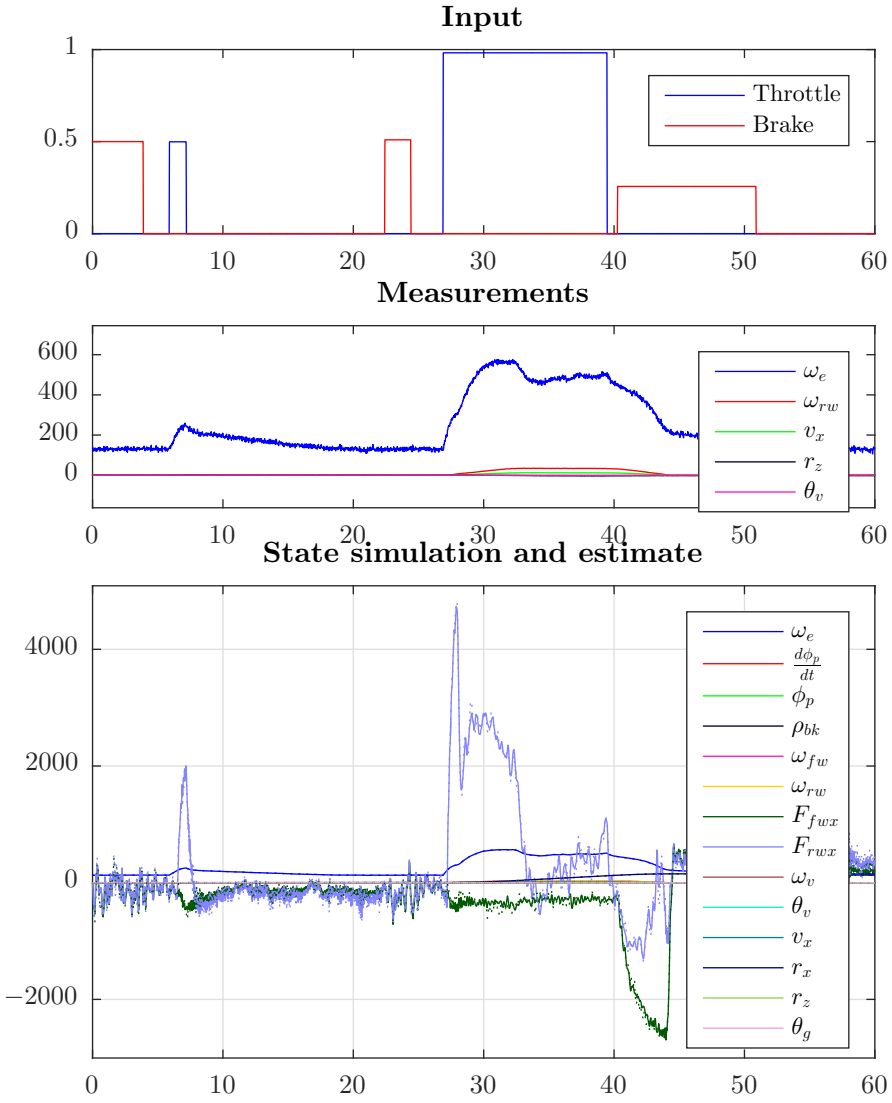


Figure A.1: Plot of a single stochastic simulation and state estimation suing the UKF. Simulated state is dotted lines and the estimated whole lines.

```
150 end
151 linkaxes(spax,'x')
152
153
154 total_time = toc(total_timer);
155 hours = floor(total_time/(60*60));
156 minutes = floor(total_time/60 - hours*60);
157 seconds = round(total_time - minutes*60 - hours*60*60);
158 fprintf('Total runtime: %d:%02d:%02d\n', hours,minutes,seconds);
```

A.4 Matlab code for parameter estimation

The two stages of the parameter estimation algorithm is implemented as two scripts. The script in listing A.8 on page 94 does the initial parameter estimation using the coast down filter model by optimising the parameters from 20 uniformly drawn starting points in the chosen parameter space using *fmincon*. The script choses the parameter estimate with the highest log likelihood, but it should be confirmed that all of the estimates are reasonably close.

Listing A.9 on page 95 shows the script that performs the estimation of the remaining

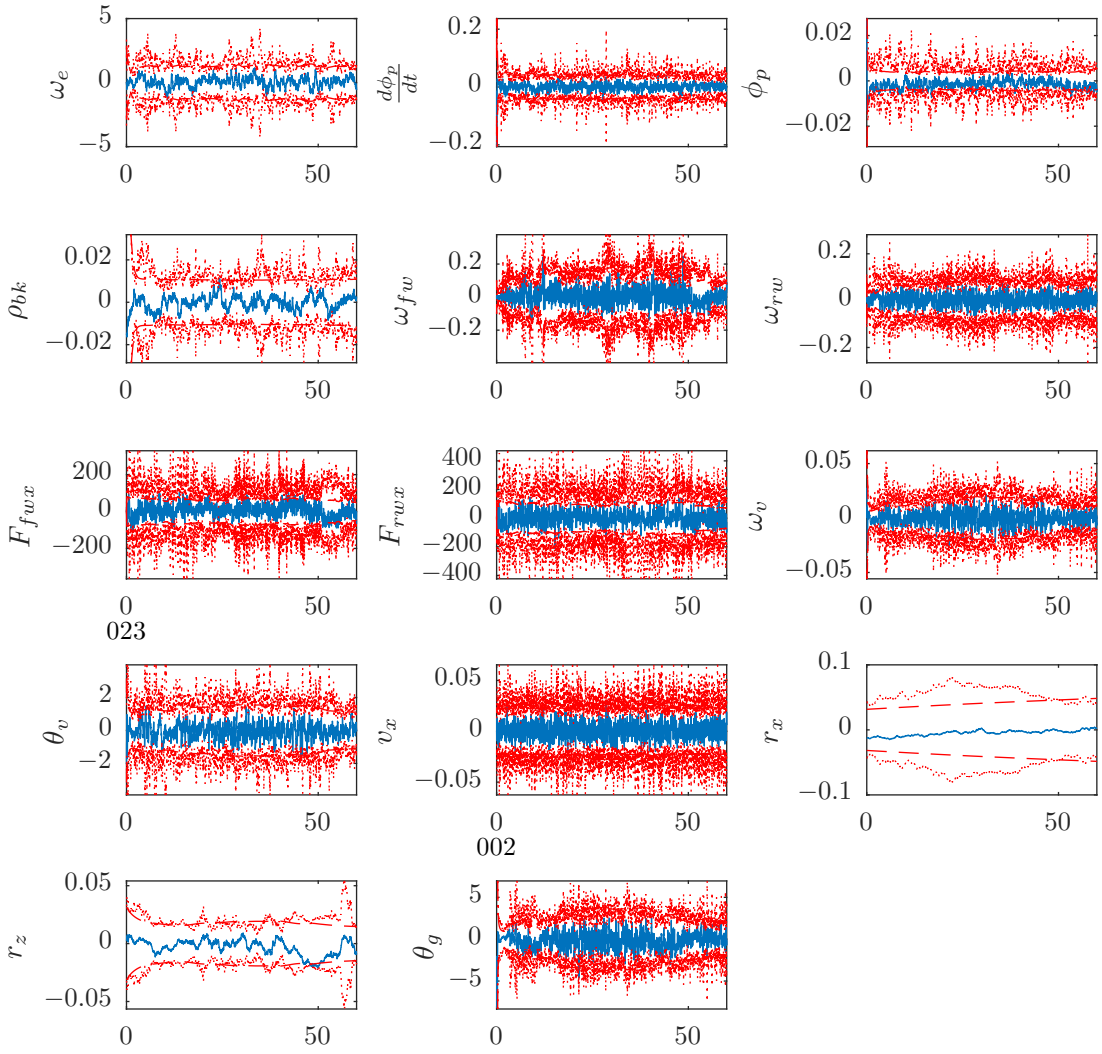


Figure A.2: Plot of estimation error as well as true covariance and covariance computed by the UKF from 10 simulations. The dotted line is one standard deviation and the dashed lines the mean of the standard deviation calculated by the UKF.

parameters using the full filter model. The parameters are optimized using *particleswarm* with a bounded parameter space. After either 100 total iterations or 20 iterations without improvement *particleswarm* is stopped and the optimization is continued by *fmincon*.

The function *LogLikelihood*, shown in listing A.10 on page 96, wraps the previously presented UKF function allowing multiple datasets to be processed. The UKF function is wrapped in a "try catch" block to detect parameters that makes the UKF fail without aborting the parameter estimation. The datasets are represented as data structures with fields containing the time, measurement, and input vectors. The datasets structures are constructed by the function *PrepareDS*, shown in listing A.11 on page 96, from the datasets provided directly by OLAVs instrumentation. *PrepareDS* also re-samples all the measurements to a common time vector.

Listing A.8: MATLAB script for performing the parameter estimation using the coast down filter model.

```
fm = 40;

% UKF tuning parameter bounds
alpha = 0.1;
beta  = 2;
kappa = 0;

disp('Loading data...')
tic()
ds_cd(1) = PrepareDS('../Datasett/2016-04-04-14-05-47.mat', (2:5), (3),
    [5,65],    fm, []); % Neutral coasting down hill.
ds_cd(2) = PrepareDS('../Datasett/2016-04-04-14-08-53.mat', (2:5), (3),
    [10,70],    fm, []); % Neutral coasting down hill.
toc()

% Start pool and disable warnings on workers
poolobject = gcp();
pctRunOnAll warning('off','all');

% Start timer for coast down experiment:
coast_down_timer = tic();

% Parameter bounds
param_cd_ub    = [0.400; 2.0e-1; 1.0e+1;  2.0e-1];
param_cd_lb    = [0.300; 0.0e+0; 0.0e+0; -2.0e-1];
param_cd_names = {'r_eff','b_rr','b_ar','theta_v0'};
Np = numel(param_cd_names);

P0_cd = diag([1e-1,1e-1,1e-1,1e-2]);

Mcd               = 5*Np;
param_cd_0        = zeros(Np, Mcd);
param_cd_est      = zeros(Np, Mcd);
objective_cd_0    = inf(Mcd,1);
objective_cd_est = inf(Mcd,1);

% Problem statement for minimization using fmincon
problem_cd.solver    = 'fmincon';
problem_cd.objective = @(param_values) -LogLikelihood(param_cd_names,
    param_values, @ModelCD, P0_cd, ds_cd, alpha, beta, kappa);
problem_cd.ub        = param_cd_ub;
problem_cd.lb        = param_cd_lb;
problem_cd.nvars     = Np;
problem_cd.options   = optimoptions(problem_cd.solver);
problem_cd.options.Display = 'none';

% Find Mcd possible valid parameter vector for coast down model
fprintf('Searching for %d starting points for coast down experiment:\n', Mcd);
parfor k = 1:Mcd
    while(objective_cd_0(k) == inf)
        param_cd_0(:,k)   = problem_cd.lb + (problem_cd.ub - problem_cd.lb).*
            rand(problem_cd.nvars,1);
        objective_cd_0(k) = problem_cd.objective(param_cd_0(:,k));
    end
    disp(strcat(sprintf('Log likelihood = %.3e, ',-objective_cd_0(k)),
```

```
        PrintParameters(param_cd_0(:,k),param_cd_names)));
52  end
53
54  % Estimate parameter from best staring points
55  disp('Estimating parameters with fmincon from staring points...');
56  disp('Best estimates for parameters in coast down model:');
57  parfor k = 1:Mcd;
58      problem_k    = problem_cd;
59      problem_k.x0 = param_cd_0(:,k);
60      [param_cd_est(:,k), objective_cd_est(k)] = fmincon(problem_k);
61      disp(strcat(sprintf('Log likelihood = %.3e, ',-objective_cd_est(k)),
            PrintParameters(param_cd_est(:,k), param_cd_names)));
62  end
63
64  % Find best estimate
65  [objective_cd_est,i_sort] = sort(objective_cd_est);
66  param_cd_est              = param_cd_est(:,i_sort);
67  param_cd_best_est         = param_cd_est(:,1);
68
69  % Stop timer for coast down experiment
70  coast_down_time = toc(coast_down_timer);
71
72  % Print result
73  disp('Best estimate for parameters in coast down model:')
74  disp(strcat(sprintf('Log likelihood = %.3e, ',-objective_cd_est(1)),
        PrintParameters(param_cd_best_est, param_cd_names)));
75  fprintf('Elapsed time: %s\n', PrintHHMMSS(coast_down_time));
```

Listing A.9: MATLAB script for performing the parameter estimation using the full filter model using the particle swarm optimization algorithm.

```
1   % Global variables for storing particel swarm progress
2   global bestx;
3   global bestf;
4
5   % Measurment sampling frequency
6   fm = 40;
7
8   % UKF tuning parameters
9   alpha_fm = 0.4;
10  beta     = 2;
11  kappa    = 0;
12
13  disp('Loading data...')
14  tic()
15  ds_fm(1) = PrepareDS('../Datasett/2016-04-04-13-30-23',[],[],[240  300],fm,1);
16  ds_fm(2) = PrepareDS('../Datasett/2016-04-04-13-30-23',[],[],[370  430],fm,3);
17  toc()
18
19  % Start pool and disable warnings on workers
20  poolobject = gcp();
21  pctRunOnAll warning('off','all');
22
23  % Start timer for full model parameter estimation
24  full_model_timer = tic();
25
26  % Parameter range and initial estimate.
27  param_ub =    [5.5e+4; 3.0e-1; 5.5e+2;  1.0e+3; 1.0e+1; 1.0e+0; 5.0e+3;  1.2e
        +5];
28  param_lb =    [1.7e+4; 1.0e-3; 1.0e+0;  1.0e+1; 1.0e-2; 2.0e-1; 1.0e+3;  2.0e
        +4];
29  param_0  =    [4.0e+4; 2.0e-2; 5.0e+2; 5.00e+2; 8.0e+0; 0.8e+0; 2.00e+3; 5.0e
        +4];
30  param_names = {'P_e_max','b_e','u_cvt','d_cvt','mu_bm','mu_tm','T_bk','k_s'};
31  Np = numel(param_names);
32
33  P0_fm = diag([5e+1, 1e-2, 1e-2, 1e-2, 1e-2, 1e-2, 1e3, 1e3, 1e-2, 1e-2, 1e-3,
        1e-3, 1e-3, 1e-3]);
34
35  % Problem statement for minimization using particle swarm
36  problem.solver    = 'particleswarm';
```

```
37 problem.objective = @(param) -LogLikelihood([param_names,param_cd_names],[
       param(:);param_cd_best_est(:)], @Model, P0_fm, ds_fm, alpha_fm, beta,
       kappa);
38 problem.ub          = param_ub;
39 problem.lb          = param_lb;
40 problem.nvars       = Np;
41 problem.options     = optimoptions(problem.solver);
42 problem.options.Display             = 'iter';
43 problem.options.UseParallel         = true;
44 problem.options.SwarmSize           = 10*problem.nvars;
45 problem.options.StallIterLimit      = 20;
46 problem.options.MaxIter             = 100;
47 problem.options.InitialSwarm        = param_0';
48 problem.options.OutputFcns          = {@(optimValues, state)
       ParameterEstimationOutputFunction(optimValues, state, param_names)};
49
50 [param_best_est_psw, objective_best_psw] = particleswarm(problem);
51
52 % Print result
53 full_model_time = toc(full_model_timer);
54 disp('Parameters for full model estimated with particle swarm:')
55 disp(strcat(sprintf('Log likelihood = %.3e, ',-objective_best_psw),
       PrintParameters([param_best_est_psw], param_names)));
56 fprintf('Elapsed time: %s\n', PrintHHMMSS(full_model_time));
57
58 disp('Try to further optimize the parameters using fmincon');
59 full_model_timer = tic();
60 problem_loc.solver    = 'fmincon';
61 problem_loc.objective = @(param) -LogLikelihood([param_names,param_cd_names],[
       param(:);param_cd_best_est(:)], @Model, P0_fm, ds_fm, alpha_fm, beta,
       kappa);
62 problem_loc.x0        = param_best_est_psw;
63 problem_loc.ub        = param_ub;
64 problem_loc.lb        = param_lb;
65 problem_loc.nvars     = Np;
66 problem_loc.options   = optimoptions(problem_loc.solver);
67 problem_loc.options.Display     = 'iter';
68 problem_loc.options.UseParallel = true;
69
70 [param_best_est, objective_best] = fmincon(problem_loc);
71 param_est  = param_best_est;
72
73 % Print result
74 full_model_time = toc(full_model_timer);
75 disp('Parameters for full model estimated with particle swarm:')
76 disp(strcat(sprintf('Log likelihood = %.3e, ',-objective_best),PrintParameters
       (param_best_est, param_names)));
77 fprintf('Elapsed time: %s\n', PrintHHMMSS(full_model_time));
```

Listing A.10: MATLAB function for computing the log likelihood of an arbitrary number of measurement series using a UKF.

```
1  function log_likelihood = LogLikelihood(param_names, param_values, model_fun,
       P0, datasets, alpha, beta, kappa)
2      try
3          log_likelihood = 0;
4          model = model_fun(param_names,param_values);
5          for ds = datasets
6              x0 = model.x0_fun(ds.z(:,1),ds.u(:,1));
7              [~,~,~,log_likelihood_ds] = ukf(ds.t, ds.z, ds.u, x0, P0, model,
                   alpha, beta, kappa);
8              log_likelihood = log_likelihood + log_likelihood_ds;
9          end
10     catch
11         log_likelihood = -inf;
12     end
13 end
```

Listing A.11: MATLAB function extracting the relevant measurements from the datasets provided by OLAV.

```
function ds = PrepareDS(filename, z_mask, u_mask, t_span, fm, h)
    % Load complete dataset from file.
    load(filename);

    % Construct time series from relevant data.
    z_nav = timeseries([navdata.x_velocity, navdata.height, navdata.pitch]',
        navdata.time);
    z_vs  = timeseries([2*pi/60*vehicleStatus.engine_rpm, vehicleStatus.
        axleRadPrSec]', vehicleStatus.time);
    u     = timeseries([vehicleStatus.current_throttle_command/100,
        vehicleStatus.current_break_command/100, vehicleStatus.urrent_gear]',
        vehicleStatus.time);
    t     = navdata.time(navdata.time <= vehicleStatus.time(end));

    % Cut time to provided time span.
    if( not(isempty(t_span)) )
        if( t_span(2) < t_span(1) )
            error('Start time larger than end time.');
        elseif ( t(1) >= t_span(1) )
            error('Start time outside dataset');
        elseif ( t(end) <= t_span(2) )
            error('End time outside dataset');
        end
        t = t( (t_span(1) <= t) & (t <= t_span(2)) );
    end

    if isempty(fm)
        % Use navdata's time vector as common time vector.
        ds.t = t;
    else
        % Use provided frequency to construct time vector.
        M = floor((t(end)-t(1))*fm);
        ds.t = t(1):1/fm:t(1) + M/fm;
    end

    % Re-sample ans concatenate measurements
    ds.z = [z_vs.resample(ds.t).data(:,:); z_nav.resample(ds.t).data(:,:)];
    ds.u = u.resample(ds.t).data(:,:);


    % Store data names
    ds.measurement_names = {'$\omega_e$','$\omega_{rw}$','$v_x$','$r_z$','$\
        theta_v$'};
    ds.input_names       = {'$\alpha$','$\beta$','G'};

    % Set brake to zero when vehicle is in neutral as this can only be done in
    % manual mode which does not report brake actuator position accurately.
    % Also chop of any parts where the vehicle was not in neutral.
    if ds.u(3,1) == 3
        mask = ds.u(3,:) == 3;
        ds.t = ds.t(mask);
        ds.z = ds.z(:,mask);
        ds.u = ds.u(:,mask);
        ds.u(2,:) = zeros(1,numel(ds.u(2,:)));
    end

    % Remove unwanted measurements
    if not(isempty(z_mask))
        ds.z = ds.z(z_mask,:);
        ds.measurement_names = ds.measurement_names(z_mask);
    end

    % Remove unwanted inputs
    if not(isempty(u_mask))
        ds.u = ds.u(u_mask,:);
        ds.input_names = ds.input_names(u_mask);
    end

    % Store number
    ds.Nz = numel(ds.measurement_names);
    ds.M  = numel(ds.t);
```

```
67      ds.Nu = size(ds.u,2);
68
69      % Plot to provided axis
70      if not(isempty(h))
71          figure(h);
72          PlotDS(ds)
73      end
74  end
```

Appendix B

Derivation of the drive line equations

This appendix presents the derivation of the equation of motion for the drive line from the bond graph model of the drive line presented in section 4.3.4. The bond graph is shown in fig. 4.10 on page 33. Figure B.1 shows the same bond graph model, but simpler notation and numbered bonds, making the derivation of the equations of motion easier to follow. All elements are assumed to not be modulated, as modulation does not change to structure of the equations. The sources of effort representing the tire and brake torques are represented as a single source.

The bond graph model has four state variables: the momentums p_1, p_2, and p_4, caused by the inertia elements connected to bond 1, 2, and 4, and the displacement q_3 caused by the capacitive element connected to bond 3. Starting with p_1 its derivative is the sum of the efforts going to the common flow node it is connected to:

$$\dot{p}_1 = e_0 - e_8 - e_9 \tag{B.1}$$

Following the causality of the efforts gives the following:

$$e_0 = Se_0 \tag{B.2}$$

$$e_8 = e_6 \tag{B.3}$$

$$e_9 = r_9 f_9 = r_9 f_2 = r_9 \frac{p_2}{i_2} \tag{B.4}$$

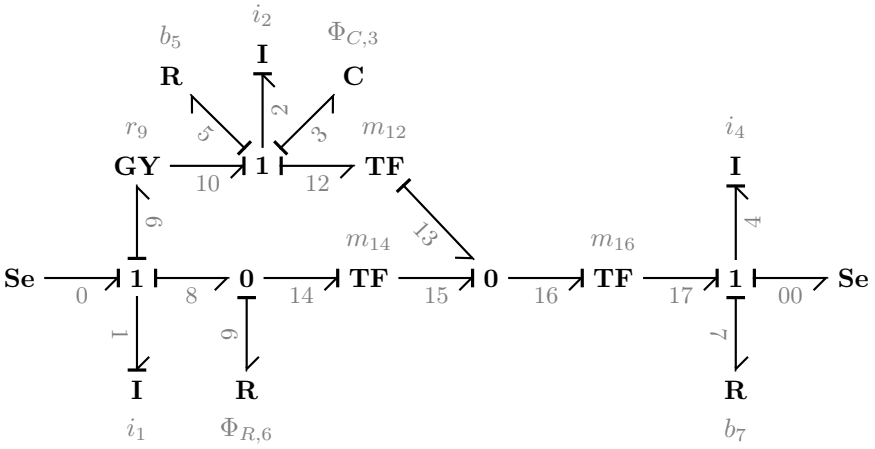


Figure B.1: Numbered version with simplified notation of the drive line bond graph originally presented in fig. 4.10 on page 33.

Resulting in the following expression for $\dot{p}_1$:

$$\dot{p}_1 = Se_0 - e_6 - r_9 \frac{p_2}{i_2} \tag{B.5}$$

Although e_6 is still not completely determined. This is because it will be reused later and it is therefore easier to consider it separately. e_6 is caused by the non linear resistance $\Phi_{b,6}$, with f_6 as input. Because of the structure of the non linear in the drive line model, the resistance is defined[1]:

$$e_6 = \Phi_{R,6}(f_6) := \Phi_{R,6}(f_8, f_{14}) \tag{B.6}$$

The two flows f_8 and f_{14} are:

$$f_8 = f_1 = \frac{p_1}{i_1} \tag{B.7}$$

$$f_{14} = m_{14} f_{15} = m_{14}(f_{13} - f_{16}) = m_{14}\left(m_{12}\frac{p_2}{i_2} - m_{16}\frac{p_4}{i_4}\right) \tag{B.8}$$

The same procedure is performed for p_2 and p_4 resulting in:

$$\dot{p}_2 = e_{10} - e_5 - e_3 - e_{12} = r_9\frac{p_1}{i_2} - b_2\frac{p_2}{i_2} - \Phi_{C,3}(q_3) - m_{12}m_{14}e_6 \tag{B.9}$$

$$\dot{p}_4 = e_{00} - e_7 + e_{17} = Se_{00} - b_7\frac{p_4}{i_4} + m_{16}m_{14}e_6 \tag{B.10}$$

$\dot{q}_3$ is found similarly, but by summing the flows instead of efforts. As it is connected to a common flow node this is simply:

$$\dot{q}_3 = f_3 = f_2 = \frac{p_2}{i_2} \tag{B.11}$$

Rewriting the equations in therms of the flows instead of the momentums gives:

$$i_1\dot{f}_1 = Se_0 - e_6 - r_9 f_2 \tag{B.12}$$

$$i_2\dot{f}_2 = r_9 f_1 - b_2 f_2 - \Phi_{C,3}^{-1}(q_3) - m_{12}m_{14}e_6 \tag{B.13}$$

$$\dot{q}_3 = f_2 \tag{B.14}$$

$$i_4\dot{f}_4 = m_{16}m_{14}e_6 - b_7 f_4 - Se_{00} \tag{B.15}$$

$$e_6 = \Phi_{R,6}(f_1, m_{14}\left(m_{12}f_2 - m_{16}f_4\right)) \tag{B.16}$$

To arrive at the equations presented in section 4.3.4 the appropriate values for the different elements, as presented in the original bond graph model shown in fig. 4.10 on page 33, are simply substituted into the equations derived here.

[1] This breaks the rules of the bond graph as it should have been a function of the sum: $f_8 - f_{14}$, and in hindsight it may have been more correct to represent the belt pulley contact with a more specialized two port element than a common effort junction and a resistive element.

Glossary

AUV autonomous underwater vehicle. 1

CVT continuously variable transmission. ix, 1, 2, 17, 19–21, 23, 29, 30, 32, 34, 35, 42, 43, 54–57, 66, 74, 75, 77

EKF extended Kalman filter. 7, 52

FFI the Norwegian Defence Research Establishment. v, 1, 23, 25

GNSS global navigation satellite system. 1, 25

IMU inertial measurement unit. 25

OLAV off-road light autonomous vehicle. ix, xiii, 1, 2, 23, 25, 27, 38, 42, 53, 55, 81, 94, 96

PLC programmable logic controller. 25

PVT Polaris Variable Transmission. 19, 23

ROS the Robotic Operating System is a framework for writing software for robots and robot like systems. It provides facilities for creating software that spans multiple executables and computers by providing a flexible message, configuration, initialization and monitoring services. It also includes an extensive repository of easy to interface programs that completes common tasks, like interfacing senors, actuators, simulation, indoor navigation etc. The standard distribution of ROS includes C++ libraries and Python modules for integrating software with ROS.[1] . 1, 26

UGV unmanned ground vehicle. v, 1

UKF unscented Kalman filter. x, xiii, 3, 5, 7, 8, 10, 52, 57, 58, 61, 66–70, 74, 77, 81, 88, 89, 92–94, 96

Bibliography

[1] *About ROS*. URL: http://www.ros.org/about-ros/ (visited on 01/19/2016).

[2] Robert Grover Brown and Patrick Y.C. Hwang. *Introduction to Random Signals and Applied Kalman filtering*. 4th edition. John Wiley & Sons, Inc., 2011.

[3] MARCO Cammalleri. "A new approach to the design of a speed-torque-controlled rubber V-belt variator." In: *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering* 219.12 (2005), pp. 1413–1427.

[4] Marco Cammalleri and Francesco Sorge. "Approximate Closed-Form Solutions for the Shift Mechanics of Rubber Belt Variators." In: *ASME 2009 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers. 2009, pp. 187–196.

[5] Joakim Carlsson and Carl Nordheim. "A Parameter Estimation method for Continuous Time dynamical Systems based on the Unscented Kalman Filter and Maximum Likelihood." MA thesis. Chalmers University of Technology, 2011.

[6] Massimo Guiggiani. *The Science of Vehicle dynamics*. Springer Netherlands, 2014.

[7] Polaris Sales Inc. *2013 RANGER XP 900 Service Manual*. 2012.

[8] The Mathworks Inc. *Particle Swarm Optimization Algorithm*. URL: http://se.mathworks.com/help/gads/particle-swarm-optimization-algorithm.html (visited on 05/15/2016).

[9] The Mathworks Inc. *What Is Particle Swarm Optimization?* URL: http://se.mathworks.com/help/gads/what-is-particle-swarm-optimization.html (visited on 05/15/2016).

[10] Reza N. Jazar. *Vehicle Dynamics: Theory and Application*. 2nd. Springer, 2014.

[11] S. J. Julier, J. K. Uhlmann, and H. F. Durrant-Whyte. "A new approach for filtering nonlinear systems." In: *American Control Conference, Proceedings of the 1995*. Vol. 3. June 1995, 1628–1632 vol.3. DOI: 10.1109/ACC.1995.529783.

[12] G. Julió and J.-S. Plante. "An experimentally-validated model of rubber-belt {CVT} mechanics." In: *Mechanism and Machine Theory* 46.8 (2011), pp. 1037–1053. ISSN: 0094-114X. DOI: http://dx.doi.org/10.1016/j.mechmachtheory.2011.04.001. URL: http://www.sciencedirect.com/science/article/pii/S0094114X11000619.

[13] Dean C. Karnop, Donald L. Margolis, and Ronald C. Rosenberg. *System Dynamics: Modeling, Simulation, and Control of Mechatronic Systems*. 5th ed. Wiley, 2012.

[14] Karel J. Keesman. *System Identification: An Introduction*. Springer, 2011.

[15] J. Kennedy and R. Eberhart. "Particle swarm optimization." In: *Neural Networks, 1995. Proceedings., IEEE International Conference on*. Vol. 4. Nov. 1995, 1942–1948 vol.4. DOI: 10.1109/ICNN.1995.488968.

[16] Maaike van der Laan and Joachim Luh. "Model-based variator control applied to a belt type CVT." In: *International Congress on Continuously Variable Power Transmission CVT99*. 1999, pp. 16–17.

[17] J. H. Lee and W. S. Yoo. "Non-singular slip (NSS) method for longitudinal tire force calculations in a sudden braking simulation." In: *International Journal of Automotive Technology* 13.2 (2012), pp. 215–222.

[18] Polaris Sales Australia Pty Ltd. *Polaris Ranger XP 900 Sunset Red Limited Edition.* URL: http://cdn.polarisindustries.com/polaris/orv/2013/documents/en-au/2013%20Polaris%20Ranger%20XP%20900%20LE%20Brochure.pdf (visited on 05/02/2015).

[19] K. Macek et al. "Dynamics modeling and parameter identification for autonomous vehicle navigation." In: *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Oct. 2007, pp. 3321–3326. DOI: 10.1109/IROS.2007.4399640.

[20] Behrooz Mashadi and David Crolla. *Vehicle Powertrain Dynamics*. Wiley, 2012.

[21] *Polaris SAE sponsorship program: FAQ.* URL: http://www.polarissuppliers.com/sae_team/faq.htm.

[22] Robert Rudd. "Estimating the Mu Slip Curve via Extended Kalman Filtering." In: *The Mathematica Journal* 11.1 (2008), pp. 91–106. URL: http://www.mathematica-journal.com/issue/v11i1/contents/MuSlipCurve/MuSlipCurve.pdf.

[23] Abhijeet Sanchawat. *Simulation of a Drivetrain of a Vehicle comprising Continuously Variable Transmission*. Ed. by Baskar P, Online–Ressource. URL: https://doaj.org/article/2e29257e9436425c84cb104f6d90d5e8.

[24] Diter Schramm and Bardini Roberto Hiller Manfred. *Vehcile Dynamics: Modeling and Simulation*. Springer, 2014.

[25] E. A. Wan and R. Van Der Merwe. "The unscented Kalman filter for nonlinear estimation." In: *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*. 2000, pp. 153–158. DOI: 10.1109/ASSPCC.2000.882463.

:wq