NTNU

Norwegian University of
Science and Technology

Autonomous Drifting of a 1:5 Scale Model Car

Jakob Lieng Jakobsen

Master of Science in Engineering Cybernetics
Submission date: June 2011
Supervisor: Sverre Hendseth, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Autonomous Drifting of a 1:5 Scale Model Car

Jakob Lieng Jakobsen

June 2011

Master's Thesis for the Degree of
MSc in Engineering Cybernetics

NTNU
Department of Engineering Cybernetics

KONGSBERG

Problem Description

LocalBug is a platform for development and testing of control systems and guidance strategies. It is a radio-controlled car with on-board GPS and IMU instrumentation. Available control actuators are the front wheel steering angle and throttle/break, which is configurable to front or rear wheels, or both. The LocalBug simulator is a vehicle simulator implemented in SIMULINK. The purpose of the simulator is to aid in the design of control systems for LocalBug.

The main goal of this thesis is to use feedback linearization to design a control system that uses the sideslip angle to stabilize LocalBug at a desired yaw rate. The student shall study feedback linearization in preparation for the control design. Testing the of the control system is to be performed in the LocalBug Simulator.

The LocalBug simulator is to be improved by adding a motor model and allowing for the selection of a front, rear or four wheel drive mode. In addition, it is desired that realistic sensor noise can be simulated in the feedback loop. Unknown vehicle parameters should be measured, and the fidelity of the simulator validated against logged test data from LocalBug.

Because this thesis is a part of an ongoing student project, other students should be able to continue the work in this report.

Abstract

Current automotive safety systems restrict the vehicle to the linear region of operation where the sideslip angle is small. Recent research in the field has discovered that drifting possesses unstable equilibria in which the vehicle is controllable even after the handling limits in the linear region have been exceeded.

This thesis presents the design and simulation of a feedback linearization controller that, by using yaw rate as input to controlling the sideslip angle, is able to find the equilibrium point corresponding to the initial velocity and the desired yaw rate. Simulation results show that the controller is able to achieve a yaw rate within 5 degrees of the desired yaw rate. It is demonstrated that utilization of drifting techniques increases the maneuverability of the vehicle compared to normal cornering. Based on the successful handling of coupling in actuator authority at high angles of sideslip, feedback linearization as a control design tool is recommended for further development of controllers in the LocalHawk project.

The LocalBug simulator has been improved by the addition of a dc motor model that includes selection of front, rear and four wheel drive. Measurement of the moment of inertia of LocalBug and recording of true noise data, which is added to the simulator output, has increased the fidelity of the simulator. Validation of the simulator shows that the simulation results largely is in agreement with logged test data, except for the case of hard breaking where the simulation model is inclined to experience a spin.

Preface

This master's thesis concludes my 5 years of higher educational studies, of which the last two has been here at the Norwegian University of Science and Technology. It has been great fun, a lot of surprises, and moments of despair and excitement exploring the world of cybernetics.

I would like to thank my supervisors, Associate Professor Sverre Hendseth (Department of Engineering Cybernetics) and Jon Bernhard Høstmark (Kongsberg Defence Systems), for guidance and constructive criticism while working on this thesis, and for keeping up my motivation in times of slow progress. I would also like to thank Professor Kristin Ytterstad Pettersen for assistance on the controller design, and Professor Lars Imsland for good advice on car dynamics literature.

Finally, thanks to my parents for always supporting me and to my dear Ida for giving me inspiration and holding out with me even in my darkest moments.

Contents

List of Figures

Chapter 1

Introduction

Rally tracks usually consist of curvy, narrow roads with occasional hairpin turns. In addition, the road surface is often a medium to low friction surface such as gravel or snow. Driving as fast as possible through these challenging courses requires a skilled driver and is only possible by maximizing the cornering forces. Because regular front wheel steering, as the only actuator capable of altering the direction of the vehicle, is not adequate to negotiate steep corners at high speeds, rally drivers turns to special driving techniques to push their vehicles to the limit.

Drifting occurs when a vehicles direction of travel at the center of gravity differs from its orientation. This angle, illustrated in figure 1.1, is called *the vehicle sideslip angle*. It is mathematically defined as the arctangent of the lateral velocity to the longitudinal velocity, given in the body frame:

$$\beta = \arctan\left(\frac{V_y}{V_x}\right) \tag{1.1}$$

Under normal driving conditions, the sideslip angle is typically within $\pm 2^\circ$ (Grip et al. 2009). In the sense of vehicle controllability, however, it is not optimal to keep this angle low. Expert rally drivers exploit the possibilities that arise at high slip angles to maximize cornering speed and thereby reducing lap times (Velenis et al. 2007). By doing so, they are utilizing the nonlinear coupling between throttle and steering at high sideslip angles to produce the forces required to keep the vehicle on the road.

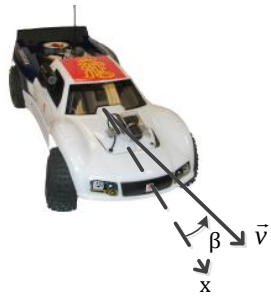

Figure 1.1: Definition of the sideslip angle.

1.1 Motivation

Recent research in automotive control systems looks into the possibility of utilizing expert rally race driving skills in active safety systems. Current automotive control systems aim at restricting the vehicle to the linear region of operation (Velenis et al. 2010). This ensures that the vehicle is open-loop stable and that the driver easily can predict the vehicle response to steering, throttle and breaking inputs. Once the vehicle enters the nonlinear region characterized by high sideslip angles it is in general open-loop unstable, but not necessarily uncontrollable (Hindiyeh and Gerdes 2010). While drifting, the rear wheels are saturated, but direction control is still available through front wheel steering. In fact, steady state equilibrium points have been found for different combinations of turning radius, speed, sideslip angle and control effort (Velenis et al. 2010). By exploiting these equilibria, collision avoidance can be attempted even when the handling limits of the vehicle have been exceeded.

The main goal of this thesis is to design a nonlinear controller that is able to obtain and sustain a desired yaw rate by controlling the sideslip angle. The principle idea is that the sideslip will contribute to rotate the force vector so it points directly into the center of the turn. For a given turn on a road surface with a certain friction, there exists a range of initial velocities that allows LocalBug to complete the turn without running of the road. Inspired by rally driving, it is desirable to extend this envelope of feasible initial conditions by establishing a sideslip during cornering. Balancing the amount of sideslip with throttle and steering will increase the magnitude of cornering force, and thus the yaw rate, that it is possible to produce as compared to normal cornering. As a result,

LocalBug will be able to negotiate steeper turns at a speed where a regular heading controller would not suffice.

It is an ulterior motive that the principles and experienced gained from the challenges met in this thesis can be applicable to later development of autopilots in the LocalHawk project. An aircraft is a second order non-holonomic system (Olfati-Saber 2002), where constraints are placed on accelerations in the lateral and vertical directions. In comparison, a car under a no-slip assumption is a first order non-holonomic system, because limitations on the sideways velocity are present. Under high angles of sideslip, the no-slip constraint is temporarily relaxed. In this regime, the vehicle dynamics resembles aerobatic maneuvers where both attitude control and agile maneuvers are important (Abdulrahim 2006), albeit in a reduced configuration space. Even though the degrees of freedom for the vehicle are less, a challenge faced in both situations is the coupling of actuator authority resulting from the nonlinear effects throughout the service envelope.

In this thesis, only the case of rear wheel drive is considered. In addition, LocalBug is assumed to be configured with a locked differential. It is further presumed that the road surface is flat, that there are no pitch and roll motions and that the road surface friction is uniform in the operating area. As a consequence, differences in traction between the left and right track are considered negligible. The thesis will restrict the geometry of the turns to circles and arcs.

1.2 The LocalHawk Project

The LocalHawk project is an interdisciplinary student project aiming at constructing an Autonomous Unmanned Aerial Vehicle (AUAV) from ground up. It originated as a master's thesis at the Norwegian University of Science and Technology in 2007 and is now supported by Kongsberg Defence Systems for promotion and recruitment purposes. Students from several educational institutions contribute through projects, master's theses and summer internships. The challenges undertaken in the project spans from design of on-board electronics, autopilot software for autonomous take-off, flight and landing, and a ground station capable of receiving telemetry data from the aircraft and sending commands. In addition to the flight critical equipment, a radar range finder and image recognition software for smart guidance is also developed as payload. This resulted in a need for a simpler test platform to evaluate separate system modules without risking a fatal crash.

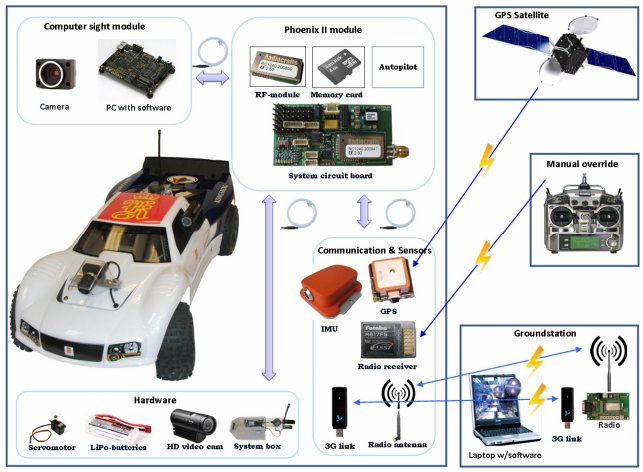

Figure 1.2: The LocalBug system.

1.2.1 The LocalBug System

LocalBug is a radio controlled, wheel-based test platform intended for testing of equipment and exploring different control strategies. The platform is equipped with the PhoenixII hardware developed for LocalHawk for data processing and an Xsens Inertial Measurement Unit (IMU) to provide sensor data. In addition, a ground station communicating with LocalBug through a radio link provides control of the on-board software and real-time telemetry monitoring. A radio controller is used to manually steer LocalBug and switch between autonomous and manual mode. Figure 1.2 illustrates the different components that constitutes the system.

PhoenixII is the on-board hardware designed to integrate all the electronics required for autonomous control. It interfaces sensor modules and actuators, and provides the autopilot software with processing power, sensor data, radio communication and data storage. Two AVR XMEGA microcontrollers, denoted "Node A" and "Node B", handles critical control tasks and non-critical tasks such as logging, respectively. The two nodes communicate through a Serial Peripheral Interface (SPI)-bus. Logged data is stored on a microSD-card which can be read on any computer, and a Radiocrafts RC1240 module is included to provide radio communication with the ground station.

The Xsens MTi-G IMU consists of a three dimensional accelerometer and gyroscope, magnetometer, barometer and a GPS receiver. A built-in Kalman filter provides estimates of velocities, position and attitude.

A laptop with an external Radiocrafts radio transceiver constitutes the ground station. Paths defined by GPS waypoints can be created on the ground station and uploaded wirelessly to LocalBug.

1.2.2 Previous Work

This thesis is a continuation of the work already done in the LocalHawk project. Major developments relevant to this thesis are presented in this section.

In the spring of 2010, the PhoenixII hardware and software drivers were completed by Veierland (2010). At the same time, the LocalBug platform was developed as a master's thesis project in Wenstad (2010). It consisted of the HPI Savage Flux HP R/C car and was equipped with the same instrumentation as the LocalHawk aircraft. A software framework to interface the autopilot software with the drivers and provide logging of autopilot data was made. The autopilot software is created using SIMULINK, and the source code is generated through Real-Time Workshop in Matlab. The autopilot source code is then linked together with the LocalBug software framework and uploaded to the two processor nodes on LocalBug. A kinematic simulator based on a no sliding assumption was also created to design a guidance controller to test the platform autonomously.

Students working at the summer internship program at Kongsberg the following summer continued contributing to the project. Amongst other things, improvements were made to the framework source code and the ground station, and logged data was used to produce a black box model of LocalBug. However, this simulator is only valid for driving conditions equal to those during the data collection. A user manual describing the LocalBug system was also produced. These developments are described in Wigestrand et al. (2010a;b).

In the fall of 2010, Jakobsen (2010) implemented a simulator based on a mathematical model of forces acting on the vehicle. The simulator demonstrated realistic vehicle behavior during aggressive driving and produced results similar to the previous kinematic simulator during normal driving. However, the simulator lacked a proper motor model and was not validated against logged data.

1.3 Contribution of Thesis

This thesis improves and validates the LocalBug simulator. A dc motor model has been implemented for better simulation of the longitudinal dynamics. Options to select front, rear or four wheel drive and differential lock have been added. A data series containing noise has been recorded using the exact sensor fitted on LocalBug. This increases the fidelity of the simulator output, which then resembles the sensor data available to the on-board controller. The moment of inertia of LocalBug in yaw, as well as the moment of inertia of the wheels around the rolling axis has been determined by measurements, and a user guide describing how to initialize and run the LocalBug simulator is written. The LocalBug simulator has been validated by comparing the logged response to simulation results using identical actuator commands.

A feedback linearization controller that obtains and sustains the desired yaw rate by the use of drifting techniques has been designed. The desired yaw rate is achieved by finding the sideslip angle that corresponds to the equilibrium point determined by the initial conditions and the desired yaw rate. The controller has been successfully tested on the LocalBug simulator.

During the course of the semester, guidance has been provided to a group of students in the subject *Eksperter i Team*. The group designed and constructed a water- and dustproof container to protect the on-board electronics from environmental impact.

1.4 Consulted Literature

Consulted literature on nonlinear systems includes Khalil (2002) and Marino and Tomei (1995), who deals with non-linear control theory and feedback linearization for single-input-single-output systems, and Isidori (1995) and Slotine and Li (1991) who in addition discusses feedback linearization for multi-input multi-output systems. Fossen (2011) goes in depth on the theory of motion control systems. Topics in modeling and simulation of dc motors are covered in Egeland and Gravdahl (2002). Kiencke and Nielsen (2005), Rajamani (2006) and Wong (2001) provides a complete description on vehicle dynamics. Methods for measurement and estimation of the vehicle sideslip angle β are described in Grip (2010) and Croft-White (2006). Ackerman (1997) presents ideas for decoupling of the yawing motion of the vehicle from the path following task of the driver. In Hindiyeh and Gerdes (2010), a dynamic surface controller for

autonomous drifting is designed, while open loop commands are used to achieve drifting in Henry and Perrault (2010). Properties of high sideslip maneuvers are investigated in Abdulrahim (2006) and Velenis et al. (2010), and stability properties of high sideslip angle cornering equilibria are discussed in Voser et al. (2010). In Velenis et al. (2007), a mathematical analysis of rally race driving techniques is performed.

1.5 Outline of Thesis

In section 2, some mathematical tools and concepts used in the thesis are reviewed. Chapter 3 presents principles and tools for the design of autonomous control systems. A mathematical vehicle model capable of describing the vehicle dynamics at high angles of sideslip is presented in chapter 4. Improvements to the simulation model and verification of the fidelity of the LocalBug simulator is described in chapter 5, while presentation of the controller design is given in chapter 6. The simulation results are discussed in chapter 7.

Chapter 2

Mathematical Preliminaries

Some of the mathematical tools and expressions used in the subsequent chapters are reviewed here.

2.1 Lie Derivatives

According to Slotine and Li (1991), the Lie derivative is the directional derivative of a function along a vector field.

Consider the single input-single output n'th order system

$$\begin{aligned} \dot{\boldsymbol{x}} &= f(\boldsymbol{x}) + g(\boldsymbol{x})u \qquad (2.1) \\ y &= h(\boldsymbol{x}) \qquad (2.2) \end{aligned}$$

where $f : \mathbb{R}^n \to \mathbb{R}^n$, $g : \mathbb{R}^n \to \mathbb{R}^n$ and $h : \mathbb{R}^n \to \mathbb{R}$ are sufficiently smooth. Both f and g are vector fields, and h is a function of $\boldsymbol{x}$. The Lie derivatives of the output function $h(\boldsymbol{x})$ with respect to the trajectories of the system are found by differentiating y with respect to time:

$$\dot{y} = \frac{\partial h}{\partial \boldsymbol{x}} \cdot \frac{\partial \boldsymbol{x}}{\partial t} = \frac{\partial h}{\partial \boldsymbol{x}} \left(f(\boldsymbol{x}) + g(\boldsymbol{x})u \right) \stackrel{def}{=} L_f h(\boldsymbol{x}) + L_g h(\boldsymbol{x})u \qquad (2.3)$$

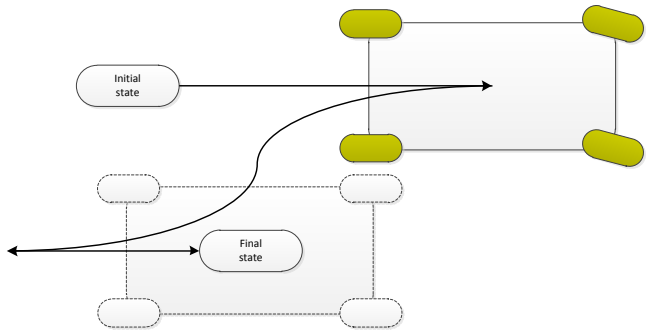

Figure 2.1: Parallel parking is an example of a system with kinematic constraints.

where

$$L_f h(\boldsymbol{x}) = \frac{\partial h}{\partial \boldsymbol{x}} f(\boldsymbol{x}) \tag{2.4}$$

$$L_g h(\boldsymbol{x}) = \frac{\partial h}{\partial \boldsymbol{x}} g(\boldsymbol{x}) \tag{2.5}$$

are the Lie derivatives along $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$, respectively. These are new scalar functions, and the process of finding the Lie derivatives can be repeated. For instance, when taking the Lie derivative of $L_f h(\boldsymbol{x})$ along f and g, the following notation is common

$$L_f^2 h(\boldsymbol{x}) = \frac{\partial L_f h(\boldsymbol{x})}{\partial \boldsymbol{x}} \cdot f(\boldsymbol{x}) \tag{2.6}$$

$$L_g L_f h(\boldsymbol{x}) = \frac{\partial L_f h(\boldsymbol{x})}{\partial \boldsymbol{x}} \cdot g(\boldsymbol{x}) \tag{2.7}$$

2.2 Non-Holonomic Systems

A *non-holonomic* system is a system where constraints on the derivatives of the configuration of the system are present, that is kinematic constraints limiting the directions in which the system can move. A *non-holonomic constraint* is expressed as an equality condition or differential equation on the form (Newman 2003)

$$f(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \ldots) = 0 \tag{2.8}$$

where $\boldsymbol{q}$ is a vector of generalized positions. On the other hand, a holonomic constraint is a constraint where the derivatives of the generalized position can be integrated to be expressed as a function of the position. Such constraints are not truly non-holonomic and is said to be *integrable* (Mason 2011). Otherwise it is said to be *non-integrable.*

A popular example of a non-holonomic system is the rolling disc without slipping. Imagine a coin rolling on a table. The configuration of the coin can be described by the position on the table, the heading of the coin, the rotation angle, and the inclination of the coin from the vertical. There is no slip, which means that the relative velocity at the contact point between the coin and the table is zero at all times. This constraint leads to the fact that the state of the system is not only dependent on the initial condition, but also on the path it took to reach it. If the coin rolled in a circular path back to the starting point, the rotation angle would, in general, not be the same as the starting angle for any arbitrary path on the table.

Non-holonomicity arises in locomotion and robotic problems when the system is underactuated. This happens when the degrees of freedom are higher than the number of independent motions the vehicle can produce (Mason 2011). A car is in general unable to travel sideways, and is therefore non-holonomic. In addition, a car cannot turn unless it is moving. A train, on the other hand, is a holonomic system even though it cannot travel sideways, because the velocity constraint can be expressed as a function of the position (the track). The zero velocity constraint at the contact point in the coin example is independent of the position of the coin, and thus one cannot integrate the velocity constraint into a constraint on the position in that example.

Non-holonomic systems are generally much harder to control because the non-holonomic constraints have to be satisfied at all times (Newman 2003). This is easily experienced when parallel parking a car, since care has to be taken to maneuver the vehicle in position, as illustrated in figure 2.1. Depending on the free space around the vehicle, an arbitrary number of movements have to be executed to attain the desired position. However, as long as the system is non-holonomic, no constraints are present on position and all configurations can be attained by carefully selecting the control inputs (Mason 2011).

2.3 The Moore-Penrose Pseudoinverse

Consider the linear system

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{2.9}$$

where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{b} \in \mathbb{R}^m$.

Assume that $m > n$. Since $\boldsymbol{A}$ is rectangular, the inverse of $\boldsymbol{A}$ does not exist. However, it is possible to find a number of generalized inverse matrices that exhibits some of the properties of the inverse matrix.

If $\boldsymbol{A}$ is of rank n, left multiplying (2.9) with $\boldsymbol{A}^T$ gives an invertible $n \times n$ matrix on the left side. It is now possible to obtain an expression for $\boldsymbol{x}$:

$$\begin{aligned} \boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} &= \boldsymbol{A}^T\boldsymbol{b} && (2.10) \\ \boldsymbol{x} &= (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{b} && (2.11) \end{aligned}$$

where $(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T = \boldsymbol{A}^\dagger$ is the Moore-Penrose pseudoinverse. For the case where $m < n$ and $rank(\boldsymbol{A}) = m$, the pseudoinverse is $\boldsymbol{A}^\dagger = \boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{A}^T)^{-1}$.

The pseudoinverse is defined and unique for all matrices with real and complex entries. What is particular about the Moore-Penrose pseudoinverse is that it is the solution of the minimization problem

$$\begin{aligned} \min_x \quad & \tfrac{1}{2}\boldsymbol{x}^2 \\ s.t \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \end{aligned} \tag{2.12}$$

If no solution to $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ exist, the pseudoinverse minimizes $\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2$ and is therefore suitable to find a best fit solution to linear systems that lacks a unique solution.

The Moore-Penrose pseudoinverse holds the following properties (Burdick n.d.):

$$\begin{aligned} \boldsymbol{A}\boldsymbol{A}^\dagger\boldsymbol{A} &= \boldsymbol{A} && (2.13) \\ \boldsymbol{A}^\dagger\boldsymbol{A}\boldsymbol{A}^\dagger &= \boldsymbol{A}^\dagger && (2.14) \\ \left(\boldsymbol{A}\boldsymbol{A}^\dagger\right)^T &= \boldsymbol{A}\boldsymbol{A}^\dagger && (2.15) \\ \left(\boldsymbol{A}^\dagger\boldsymbol{A}\right)^T &= \boldsymbol{A}^\dagger\boldsymbol{A} && (2.16) \end{aligned}$$

- If $m > n$, there are more constraints than free variables and there exists at most one solution to the system. The pseudoinverse gives the solution that minimizes the least squares residuals of $\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}$. In other words, it gives the solution closest to $\boldsymbol{b}$ in a least squares sense.
- If $m < n$, there are an infinite number of solutions to the system. In this case, $\boldsymbol{A}^\dagger$ gives the minimum norm solution of $\boldsymbol{x}$.
- In the case where $m = n$ and $\boldsymbol{A}$ is of full rank, the Moore-Penrose pseudoinverse reduces to the inverse of $\boldsymbol{A}$.

Chapter 3

Principles of Guidance, Navigation & Control

The science of designing control systems for automatic or remote motion control of vehicles and crafts is called guidance, navigation and control (GNC)(Fossen 2011). The task is traditionally divided into three separate subsystems working together, as illustrated in figure 3.1. This separation simplifies the design of each task, as well as making the system easier to maintain. The three modules are a path generator, a sensor system and a feedback control system.

Guidance system Whether the problem at hand is adaptive cruise control, automatic parking of automobiles or autopilots for ships and aircrafts, the control system requires a reference to follow. The guidance system copes with the challenge of deducing the desired paths or trajectories to be tracked by the vehicle. In its simplest form, the guidance system is a set-point defined by the operator, for instance a heading or speed command. More advanced guidance systems calculates the position, velocity and acceleration required for the vehicle to steer safely and comfortably to the target. These algorithms utilize weather data, topological and obstacle information to calculate collision-free trajectories, and optimization techniques to minimize for instance fuel consumption, travel time or actuator wear.

A number of guidance schemes exists, some of which are described in Fossen (2011), Isidori et al. (2003) and Vold (2010).

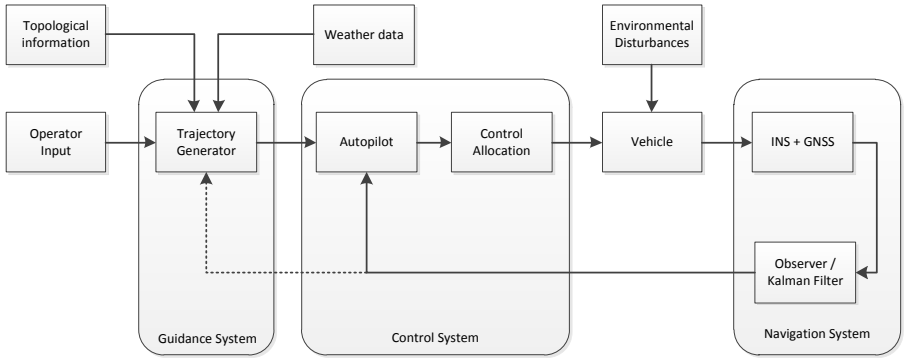

Figure 3.1: Principle sketch of a Guidance, Navigation and Control system.

Navigation system The objective of the navigation subsystem is to determine the current position and attitude of the vehicle. Today, this is usually accomplished by the use of Global Navigation Satellite Systems (GNSS) in combination with an Inertial Navigation Systems (INS), which consists of accelerometers and gyroscopes (Vik n.d.). In addition, the navigation system often uses a Kalman filter or an observer to remove noise and estimate states such as velocities and accelerations of the vehicle.

Control system The task of the control system is to produce the actual actuator commands that cause the vehicle to track the output from the guidance system. Often the control system needs to be designed in conjunction with the guidance system to ensure that the control objectives are met. Depending on the guidance system, different requirements are put on the controller. For instance, a set-point regulation problem will only require stabilization of the vehicle, while a tracking problem demands that the controller is able to follow a time-varying reference signal.

For practical purposes it is common control the forces and moments acting on the vehicle. This leads to the problem of control allocation, which is how to assign the available actuators to produce the desired forces that solves the control problem. This often results in an optimization problem, as several actuator configurations can give the same forces and moments.

3.1 Control of Underactuated Vehicles

A motion control problem is said to be *underactuated* if the vehicle does not have independent control forces and moments in all its degrees of freedom. The *degrees of freedom* (DoF) are the number of generalized coordinates ($\boldsymbol{\eta}$) required to completely describe the motion of the system. For a general rigid body operating in a Euclidean space, the number of DoF is six: three translational displacements in surge, sway and heave, and three rotations, roll, pitch and yaw.

$$\boldsymbol{\eta} = \begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \end{bmatrix} \tag{3.1}$$

The space spanned by all possible positions the vehicle can attain is called the *configuration space* (Fossen 2011). In other words, the dimension of the configuration space equals the number of degrees of freedom

$$\dim(\boldsymbol{\eta}) = n \tag{3.2}$$

Solving an underactuated control problem is non-trivial, as there are no general methods to accomplish this. It is possible, however, to reduce the configuration space so the *control problem* is fully actuated. This is done by considering the control objective in a *working space* while leaving the other states uncontrolled. The working space is a reduced space of dimension $m < n$ in which the control objective is defined. The uncontrolled equations of motion will then appear as dynamic constraints in the state space (Fossen 2011), and the system to be controlled is then, by definition, non-holonomic.

3.2 Feedback Linearization

The general idea of feedback linearization is to find a feedback control law that exactly cancels the nonlinearities of the system. The system will then act as a linear system, and the wide range of tools available for linear systems can be utilized to design a linear controller that is superimposed on the nonlinear

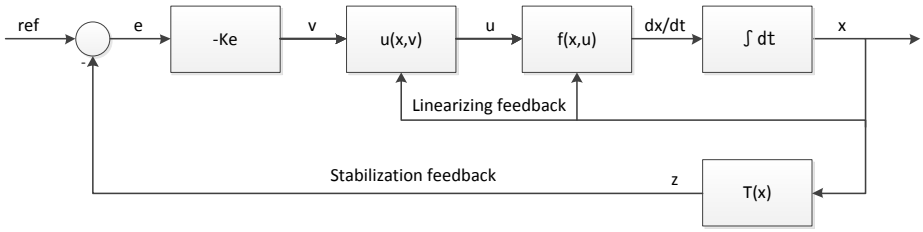

Figure 3.2: Schematic of a feedback linearization control loop.

controller to solve the control problem at hand. Figure 3.2 gives an illustration of a feedback linearization scheme.

There are a couple of problems associated with this approach. First, one can in practice not expect to be able to model a system and implement a controller where all parameters are exactly known and constant for all future time. If this is the case then exact cancellation of the system is not possible because of the uncertainties present. However, in many applications the feedback linearization approach proves to be sufficient to describe the nonlinearities of the system. If adequate robustness is not achieved, other techniques such as sliding mode control (SMC) can force the system onto a manifold for which stability is achieved, even in the presence of disturbances. The reader is referred to Khalil (2002) or Slotine and Li (1991) for further discussion on the topic.

Secondly, the structure of the system may not allow a feedback control law to completely linearize the system equations. It is, however, well known that a state space representation is not unique. Therefore, a coordinate transformation to another set of state variables may give a structure where feedback linearization is possible. It is shown in Khalil (2002) that for a system to be feedback linearizable, there must exist a transform $\boldsymbol{z} = T(\boldsymbol{x})$ that transforms the system equations into

$$\dot{\boldsymbol{z}} = \boldsymbol{A}\boldsymbol{z} + \boldsymbol{B}\gamma(\boldsymbol{z})\left[\boldsymbol{u} - \alpha(\boldsymbol{z})\right] \tag{3.3}$$

where the pair $(\boldsymbol{A}, \boldsymbol{B})$ is controllable and $\gamma(\boldsymbol{x})$ is non-singular. To be able transform the state equations back to the original state variables $\boldsymbol{x}$, T should be invertible such that

$$\boldsymbol{x} = T^{-1}(\boldsymbol{z}) \tag{3.4}$$

In addition, T and T^{-1} is required to be continuously differentiable. A function T that fulfills these requirements is called a *diffomorphism*. T is a *global diffomorphism* if and only if $\partial T/\partial \boldsymbol{x}$ is non-singular for all $\boldsymbol{x} \in \mathbb{R}^n$ and T is proper (Khalil 2002). Even if such a transformation cannot be found, the system may be partially linearizable. For instance, the system output is often the most interesting variable, and if the output equation is linearizable one may not care if the internal states are nonlinear in their response as long as they are stable.

This discussion gives rise to two techniques called *input-output linearization,* in which the input-output map is linearized, and *full-state linearization* which is achieved when the complete state vector is linearized (Khalil 2002). State linearization does, however, not guarantee that the input-output map is linear.

In the following sections input-output linearization for single-input single-output (SISO) systems is investigated, and then the methods are extended to multi-input multi-output systems (MIMO).

3.2.1 I/O Linearization For SISO Systems

Consider the n'th order system

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}) + g(\boldsymbol{x})u \tag{3.5}$$
$$y = h(\boldsymbol{x}) \tag{3.6}$$

where f, g and h are sufficiently smooth in a domain in $\mathbb{R}^n$. Since the objective is to find a *linear* relation between the input and the output, the starting point is to find *a* relationship. It is clear from (3.6) that the output depends on the system states, and from (3.5) that the input appears in the state equations. Differentiating the output gives

$$\dot{y} = \frac{\partial h}{\partial \boldsymbol{x}} \cdot \frac{\partial \boldsymbol{x}}{\partial t} = L_f h(\boldsymbol{x}) + L_g h(\boldsymbol{x})u \tag{3.7}$$

If $L_g h(\boldsymbol{x}) = 0$, then the output is not yet directly influenced by the input. However, one can continue the process of differentiating the output. As long as the system is controllable, an equation where both the input and the output are present will appear after at most n iterations (Slotine and Li 1991). This has now produced a chain of integrators that connects the input u to the output y. The output equation takes the following form

$$y^{(r)} = L_f^r h(\boldsymbol{x}) + L_g L_f^{r-1} h(\boldsymbol{x})u \tag{3.8}$$

where r is known as the relative degree of the system. This integer corresponds to the number of times one has to differentiate the output for the input to appear. For linear systems, the relative degree equals the difference between the numerator and denominator polynomial degree. It can now be seen that choosing the input as

$$u = \frac{1}{L_g L_f^{r-1} h(\boldsymbol{x})} \left[v - L_f^r h(\boldsymbol{x}) \right] \tag{3.9}$$

reduces the system to

$$y^{(r)} = v \tag{3.10}$$

where v can be designed as a linear controller, e.g. $v = -k\boldsymbol{x}$. It is important to notice, however, that the control law is not defined at $L_g L_f^{r-1} h(\boldsymbol{x}) = 0$.

In this procedure, one or more states can be rendered unobservable from the output. It is therefore necessary to determine if these states are stable, referred to as internal stability. This is accomplished by setting up the system on *normal form* and checking that the zero-dynamics are stable. According to Khalil (2002), the normal form is obtained by the transformation

$$\boldsymbol{z} = T(\boldsymbol{x}) = \begin{bmatrix} \phi_1(\boldsymbol{x}) \\ \vdots \\ \phi_{n-r}(\boldsymbol{x}) \\ h(\boldsymbol{x}) \\ \vdots \\ L_f^{r-1} h(\boldsymbol{x}) \end{bmatrix} \tag{3.11}$$

where the last elements in the transformation are a chain of integrators and ϕ_1 to ϕ_{n-r} are chosen such that $T(\boldsymbol{x})$ is a diffomorphism. These elements will define the internal states of the system which are unobservable from the output. Therefore it follows that they have to satisfy

$$\frac{\partial \phi_i}{\partial \boldsymbol{x}} g(\boldsymbol{x}) = 0, \quad \text{for } 1 \leq i \leq n - r \tag{3.12}$$

If the relative degree is equal to the order of the system then there are no internal states, as can be seen from (3.11) since all the ϕ's disappear. Otherwise there exist $n - r$ internal states. The complete system is now on the normal form (Khalil 2002)

$$\dot{\boldsymbol{\eta}} = f(\boldsymbol{\eta}, \boldsymbol{\zeta}) \tag{3.13}$$
$$\dot{\boldsymbol{\zeta}} = \boldsymbol{A}\boldsymbol{\zeta} + \boldsymbol{B}\gamma(\boldsymbol{x})\left[u - \alpha(\boldsymbol{x})\right] \tag{3.14}$$
$$y = \boldsymbol{C}\boldsymbol{\zeta} \tag{3.15}$$

The zero-dynamics are found by letting $y \equiv 0$. It follows that

$$\boldsymbol{\zeta} = 0 \Rightarrow \dot{\boldsymbol{\zeta}} = 0 \Rightarrow \dot{\boldsymbol{\eta}} = f(\boldsymbol{\eta}, 0) \tag{3.16}$$

If $f(\boldsymbol{\eta}, 0)$ is Hurwitz, then the internal dynamics are stable.

3.2.2 I/O Linearization For MIMO Systems

Feedback linearization for MIMO systems is for some classes of systems, where the number of inputs is the same as the number of outputs, quite similar to the SISO case. Consider a system of order n with m inputs and m outputs and a well defined vector relative degree

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}) + \sum_{i=1}^{m} g_i(\boldsymbol{x})u_i \tag{3.17}$$
$$y_1 = h_1(\boldsymbol{x}) \tag{3.18}$$
$$\vdots$$
$$y_m = h_m(\boldsymbol{x}) \tag{3.19}$$

Since the system consists of more than one output, the concept of a relative degree has to be extended. The *vector relative degree*, $\{r_1, \ldots, r_m\}$, is the number of times the m'th output has to be differentiated before one of the inputs appear (Isidori 1995). Calculating the Lie derivatives for each output

$$\begin{bmatrix} y_1^{(r_1)} \\ y_2^{(r_2)} \\ \vdots \\ y_m^{(r_m)} \end{bmatrix} = \begin{bmatrix} L_f^{r_1} h_1(\boldsymbol{x}) \\ L_f^{r_2} h_2(\boldsymbol{x}) \\ \vdots \\ L_f^{r_m} h_m(\boldsymbol{x}) \end{bmatrix} + \boldsymbol{A}(\boldsymbol{x})\boldsymbol{u} \tag{3.20}$$

where the matrix $\boldsymbol{A}(\boldsymbol{x})$ is

$$\boldsymbol{A}(\boldsymbol{x}) = \begin{bmatrix} L_{g_1} L_f^{r_1-1} h_1(\boldsymbol{x}) & \cdots & L_{g_m} L_f^{r_1-1} h_1(\boldsymbol{x}) \\ L_{g_1} L_f^{r_2-1} h_2(\boldsymbol{x}) & \cdots & L_{g_m} L_f^{r_2-1} h_2(\boldsymbol{x}) \\ \vdots & \ddots & \vdots \\ L_{g_1} L_f^{r_m-1} h_m(\boldsymbol{x}) & \cdots & L_{g_m} L_f^{r_m-1} h_m(\boldsymbol{x}) \end{bmatrix} \tag{3.21}$$

and non-singular. The feedback linearizing control law

$$\boldsymbol{u} = \boldsymbol{A}(\boldsymbol{x})^{-1} \begin{bmatrix} v_1 - L_f^{r_1} h_1(\boldsymbol{x}) \\ v_2 - L_f^{r_2} h_2(\boldsymbol{x}) \\ \vdots \\ v_m - L_f^{r_m} h_m(\boldsymbol{x}) \end{bmatrix} \tag{3.22}$$

with v_1 to v_m designed as linear controllers will reduce the system to the linear input-output relationship

$$y_i^{(r_i)} = v_i, \quad \text{for } 1 \leq i \leq m \tag{3.23}$$

If the total relative degree $r = r_1 + \ldots + r_m = n$, then there are no internal dynamics (Slotine and Li 1991). Otherwise, it is proved in Isidori (1995) that there is always possible to find $n - r$ more functions such that

$$\phi(\boldsymbol{x}) = \begin{bmatrix} h_1(\boldsymbol{x}) & \ldots & L_f^{r_i-1} h_1(\boldsymbol{x}) & \ldots & h_m(\boldsymbol{x}) & \ldots \end{bmatrix} \tag{3.24}$$

$$L_f^{r_m-1} h_m(\boldsymbol{x}) \quad \phi_{r+1}(\boldsymbol{x}) \quad \ldots \quad \phi_n(\boldsymbol{x}) \Big]^T \tag{3.25}$$

has a Jacobian matrix which is non-singular at $\boldsymbol{x}_0$, and can be used as a local coordinate transformation to achieve the multivariable equivalent of the normal form.

If the MIMO-system in question does not have a well defined vector relative degree, the control law (3.22) is not defined. A possible solution is then to use the *dynamic extension algorithm*. This technique introduces a new system in form of a set of state variables. The input to the original system is set to be the output of the newly introduced system. This is called *dynamic state feedback*, as opposed to *static state feedback*, since the input to the system in question gains its own dynamics. The advantage of this scheme is that one can extend the relative degree and hope that all inputs eventually appear in the output equation. The dynamic extension algorithm will not be pursued further in this thesis, but the reader is referred to Isidori (1995) for more discussion on the topic.

Chapter 4

Vehicle Models

Several vehicle models exist, and the most famous is perhaps the bicycle model. The reason for its popularity is mainly because of its linear nature and the many mathematical tools available for linear systems. During normal driving, the speed is often constant for longer periods of time and the slip angles are small. Under these assumptions, the linear bicycle model gives accurate results. However, the dynamic motion of vehicles is highly non-linear. For driving situations where the vehicle and driver is pushed to the limits, for instance in rally driving where high sideslip angles are common, this model is not adequate. Therefore a nonlinear vehicle model is needed to describe the vehicle dynamics in these extreme situations.

In the following section, the vehicle coordinate system is defined. A non-linear vehicle model is presented in section 4.2 and the linearized version of it is then given in section 4.3.

4.1 Vehicle Coordinate Systems

The vehicle body frame is defined with the origin located at the center of mass of the vehicle, and with the positive x-axis pointing in the longitudinal direction of the vehicle, the positive z-axis up from the road surface and the y-axis orthogonal to the previous axes according to the right hand rule. This coordinate frame, denoted b, is rotated an angle ψ relative to the inertial frame, designated in. The inertial reference frame is taken to be the local tangent plane on the surface

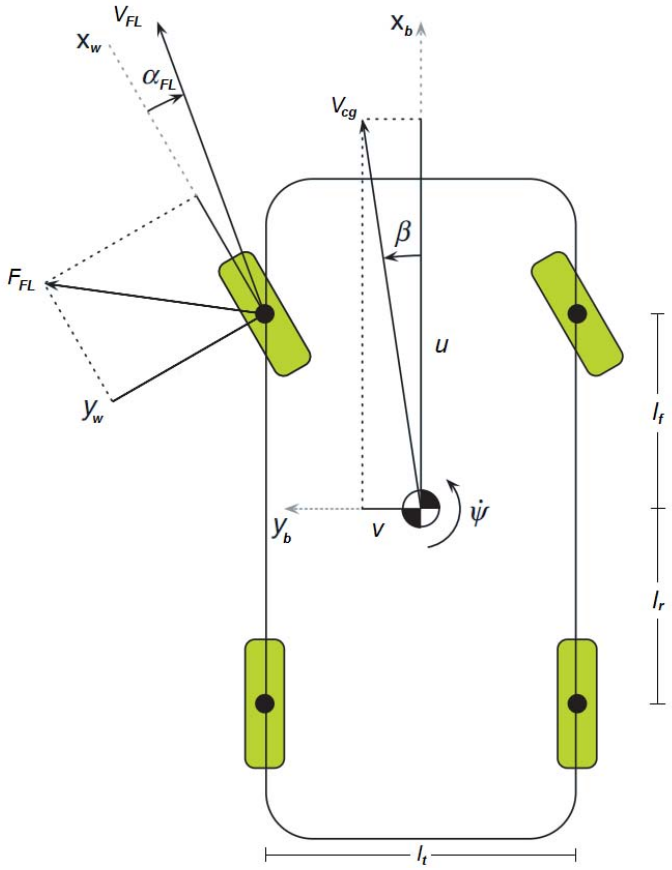

Figure 4.1: Illustration of the vehicle and its variables.

of the Earth.

A wheel frame is defined with positive x-axis along the longitudinal direction of the tire and the other axes consistent with the vehicle body frame convention. The rear wheel coordinate frame coincides with the body frame, while the front wheel coordinate frames are rotated a steering angle δ relative to the body frame. Variables belonging to the four wheels are denoted with subscript FL, FR, RL and RR for front left, front right, rear right and rear left wheel, respectively.

Figure 4.1 illustrates the definition of the variables describing the vehicle. The length l_f is the distance from the center of gravity to the front of the vehicle, while l_r is the distance from the center of gravity to the rear. The width of the vehicle is denoted l_t. The vehicle sideslip angle β is the angle between the vehicle x-axis and the velocity vector at the center of gravity. Each wheel has a slip angle α from the wheel x-axis to the velocity vector, which is positive in the clockwise direction.

The vehicle is considered as a rigid body moving on a flat surface. A pair of longitudinal and sideways wheel forces, F_L and F_S, acts in the contact point between the tire and ground in each corner of the rigid body. These forces are defined in the wheel frame. The aerodynamic drag is defined along the longitudinal axis of the vehicle. The origin and mathematical derivation of the forces acting on the vehicle are further discussed in Jakobsen (2010).

4.2 Nonlinear Two-Track Vehicle Model

In contrast to e.g. the simulation model discussed and implemented in Jakobsen (2010), this thesis is concerned with control design. While a simulation model is designed to describe every aspect of the vehicle behavior, it is too complex to be used in control design. It is therefore necessary to find a model that is only concerned with the states necessary for feedback control. A common selection of states in vehicle control is the vehicle speed, v_{cg}, the sideslip angle, β, and the yaw rate, $\dot{\psi}$. The reduced nonlinear vehicle model described in Kiencke and Nielsen (2005) is presented here.

Because Newtonian mechanics are defined in the inertial reference frame, the basis for developing the model starts by considering a moving vehicle in the inertial coordinate system. By accounting for the slip angle β, the velocities in each direction is written as

$$\begin{bmatrix} \dot{x}_{in} \\ \dot{y}_{in} \end{bmatrix} = v_{cg} \begin{bmatrix} \cos(\beta + \psi) \\ \sin(\beta + \psi) \end{bmatrix} \tag{4.1}$$

where v_{cg} is the vehicle speed at the center of gravity and ψ is the vehicle heading. Equation (4.1) is differentiated to obtain the inertial accelerations

$$\begin{bmatrix} \ddot{x}_{in} \\ \ddot{y}_{in} \end{bmatrix} = v_{cg} \left(\dot{\beta} + \dot{\psi}\right) \begin{bmatrix} -\sin(\beta + \psi) \\ \cos(\beta + \psi) \end{bmatrix} + \dot{v}_{cg} \begin{bmatrix} \cos(\beta + \psi) \\ \sin(\beta + \psi) \end{bmatrix} \tag{4.2}$$

and the resulting expression is transformed to the body frame

$$\begin{bmatrix} \ddot{x}_{cg} \\ \ddot{y}_{cg} \end{bmatrix} = R_{in}^{b}(\psi) \cdot \begin{bmatrix} \ddot{x}_{in} \\ \ddot{y}_{in} \end{bmatrix} \tag{4.3}$$

$$= v_{cg} \left(\dot{\beta} + \dot{\psi}\right) \begin{bmatrix} -\sin\beta \\ \cos\beta \end{bmatrix} + \dot{v}_{cg} \begin{bmatrix} \cos\beta \\ \sin\beta \end{bmatrix} \tag{4.4}$$

where $R_{in}^{b}(\psi)$, the rotation matrix from body to inertial frame, is written as

$$R_{in}^{b}(\psi) = R_{b}^{in}(\psi)^{-1} = \begin{bmatrix} \cos\psi & \sin\psi \\ -\sin\psi & \cos\psi \end{bmatrix} \tag{4.5}$$

According to Newton's second law, the equations for horizontal motion are written as

$$v_{cg} \left(\dot{\beta} + \dot{\psi}\right) \begin{bmatrix} -\sin\beta \\ \cos\beta \end{bmatrix} + \dot{v}_{cg} \begin{bmatrix} \cos\beta \\ \sin\beta \end{bmatrix} = \frac{1}{m_{cg}} \cdot \begin{bmatrix} F_X \\ F_Y \end{bmatrix} \tag{4.6}$$

where F_X and F_Y is the sum of forces in the body frame, acting in the longitudinal and lateral direction, respectively. The two equations in (4.6) is rearranged to solve for the vehicle acceleration and side slip velocity

$$\dot{v}_{cg} = \frac{1}{m_{cg}\cos\beta} \cdot F_X + v_{cg} \left(\dot{\beta} + \dot{\psi}\right) \tan\beta \tag{4.7}$$

$$\dot{\beta} = \frac{1}{m_{cg} v_{cg} \cos\beta} \left(F_Y - m_{cg}\dot{v}_{cg}\sin\beta\right) - \dot{\psi} \tag{4.8}$$

and the coupling between them is eliminated by substitution of the other

$$\dot{v}_{cg} = \frac{\cos\beta}{m_{cg}} F_X + \frac{\sin\beta}{m_{cg}} F_Y \tag{4.9}$$

$$\dot{\beta} = -\frac{\sin\beta}{m_{cg} v_{cg}} F_X + \frac{\cos\beta}{m_{cg} v_{cg}} F_Y - \dot{\psi} \tag{4.10}$$

As discussed in 4.1, the forces acting in the horizontal plane consists of

$$\begin{bmatrix} F_X \\ F_Y \end{bmatrix} = \begin{bmatrix} F_{x,FL} + F_{x,FR} + F_{x,RL} + F_{x,RR} + F_{x,wind} \\ F_{y,FL} + F_{y,FR} + F_{y,RL} + F_{y,RR} \end{bmatrix} \tag{4.11}$$

where $F_{x,ij}$ and $F_{y,ij}$ are the wheel forces and $F_{x,wind}$ is the aerodynamic drag

$$F_{x,wind} = -\frac{1}{2} c_{air,x} A_{front} \rho_{air} v_{cg}^2 \tag{4.12}$$

The wheel forces acts in the wheel frame. For the front wheels this frame is rotated by the steering angle δ from the body frame. The expressions for the transformed wheel forces acting in the longitudinal direction of the vehicle is written as

$$F_{x,FL} = F_{L,FL}\cos\delta - F_{S,FL}\sin\delta \tag{4.13}$$

$$F_{x,FR} = F_{L,FR}\cos\delta - F_{S,FR}\sin\delta \tag{4.14}$$

$$F_{x,RL} = F_{L,RL} \tag{4.15}$$

$$F_{x,RR} = F_{L,RR} \tag{4.16}$$

and the wheel forces in the lateral direction

$$F_{y,FL} = F_{S,FL}\cos\delta + F_{L,FL}\sin\delta \tag{4.17}$$

$$F_{y,FR} = F_{S,FR}\cos\delta + F_{L,FR}\sin\delta \tag{4.18}$$

$$F_{y,RL} = F_{S,RL} \tag{4.19}$$

$$F_{y,RR} = F_{S,RR} \tag{4.20}$$

The longitudinal wheel forces can be controlled by the driving torque from the engine and the breaks and therefore considered as the control input. Only the

lateral wheel forces are then undecided. These forces can be approximated by assuming a linear relationship between the wheel slip angle α and the resulting force. The wheel slip angle is found from the geometrical relationship between the steering angle, vehicle sideslip and rotational velocity.

$$F_{S,FL} = c_{FL} \cdot \alpha_{FL} = c_{FL} \cdot \left(\delta - \beta - \frac{l_f \dot{\psi}}{v_{cg}} \right) \tag{4.21}$$

$$F_{S,FR} = c_{FR} \cdot \alpha_{FR} = c_{FR} \cdot \left(\delta - \beta - \frac{l_f \dot{\psi}}{v_{cg}} \right) \tag{4.22}$$

$$F_{S,RL} = c_{RL} \cdot \alpha_{RL} = c_{RL} \cdot \left(-\beta + \frac{l_r \dot{\psi}}{v_{cg}} \right) \tag{4.23}$$

$$F_{S,RR} = c_{RR} \cdot \alpha_{RR} = c_{RR} \cdot \left(-\beta + \frac{l_r \dot{\psi}}{v_{cg}} \right) \tag{4.24}$$

As described in Jakobsen (2010), the force coefficient c_{ij} can be found as the gradient of the friction curve evaluated at zero slip. This, however, requires knowledge of the conditions at the road surface. To avoid this, Kececi and Tao (2006) suggests that the force coefficient can instead be described by a tire constant and the friction parameter which then can be adapted by an online parameter estimation scheme.

$$c_{ij} = c\mu_{ij} \tag{4.25}$$

The yawing moment is described in Jakobsen (2010) as

$$\begin{aligned} I_z \ddot{\psi} = {} & (F_{y,FR} + F_{y,FL}) \cdot l_f - (F_{y,RR} + F_{y,RL}) \cdot l_r \\ & + (F_{x,RR} - F_{x,RL}) \cdot \frac{l_t}{2} + (F_{x,FR} - F_{x,FL}) \cdot \frac{l_t}{2} \end{aligned} \tag{4.26}$$

where l_f is the distance from center of gravity to the front axle, l_r is the distance from center of gravity to the rear axle and l_t is the wheel tread. Equations (4.9), (4.10) and (4.26) forms the basis for the vehicle model. Inserting the expressions for the wheel forces completes the nonlinear two-track vehicle model:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{v}_{cg} \\ \dot{\beta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} f_1(\boldsymbol{x}, \boldsymbol{u}) \\ f_2(\boldsymbol{x}, \boldsymbol{u}) \\ f_3(\boldsymbol{x}, \boldsymbol{u}) \end{bmatrix} \tag{4.27}$$

with the state and input vector

$$\boldsymbol{x} = \begin{bmatrix} v_{cg} \\ \beta \\ \dot{\psi} \end{bmatrix} \quad , \qquad \boldsymbol{u} = \begin{bmatrix} F_{L,FL} \\ F_{L,FR} \\ F_{L,RL} \\ F_{L,RR} \\ \delta \end{bmatrix} \tag{4.28}$$

where the full expressions for the nonlinear functions $f_1(\boldsymbol{x}, \boldsymbol{u})$, $f_2(\boldsymbol{x}, \boldsymbol{u})$ and $f_3(\boldsymbol{x}, \boldsymbol{u})$ are

$$\begin{aligned} f_1(\boldsymbol{x}, \boldsymbol{u}) \;=\; & \frac{1}{m_{cg}} \{ (F_{L,FL} + F_{L,FR}) \cdot \cos(\delta - \beta) \\ & - (c_{FL} + c_{FR}) \cdot \left(\delta - \beta - \frac{l_f \dot{\psi}}{v_{cg}} \right) \cdot \sin(\delta - \beta) \\ & + \left(F_{L,RL} + F_{L,RR} - \frac{1}{2} c_{air,x} A_{front} \rho_{air} v_{cg}^2 \right) \cdot \cos\beta \\ & + (c_{RL} + c_{RR}) \cdot \left(-\beta + \frac{l_r \dot{\psi}}{v_{cg}} \right) \cdot \sin\beta \Big\} \end{aligned} \tag{4.29}$$

$$\begin{aligned} f_2(\boldsymbol{x}, \boldsymbol{u}) \;=\; & \frac{1}{m_{cg} v_{cg}} \left\{ (c_{FL} + c_{FR}) \cdot \left(\delta - \beta - \frac{l_f \cdot \dot{\psi}}{v_{cg}} \right) \cdot \cos(\delta - \beta) \right. \\ & + (F_{L,FL} + F_{L,FR}) \cdot \sin(\delta - \beta) \\ & - \left(F_{L,RL} + F_{L,RR} - \frac{1}{2} c_{air,x} A_{front} \rho_{air} v_{cg}^2 \right) \cdot \sin\beta \\ & \left. + (c_{RL} + c_{RR}) \cdot \left(-\beta + \frac{l_r \cdot \dot{\psi}}{v_{cg}} \right) \cos\beta \right\} - \dot{\psi} \end{aligned} \tag{4.30}$$

$$
\begin{aligned}
f_3(\boldsymbol{x},\boldsymbol{u}) \quad = \quad & \frac{1}{I_z}\{l_f \cdot (F_{L,FL} + F_{L,FR}) \cdot \sin\delta \\
& + l_f \cdot (c_{FL} + c_{FR}) \cdot \left(\delta - \beta - \frac{l_f\dot{\psi}}{v_{cg}}\right) \cdot \cos\delta \\
& + \frac{l_t}{2} \cdot (F_{L,FR} - F_{L,FL}) \cdot \cos\delta \\
& - \frac{l_t}{2} \cdot (c_{FR} - c_{FL}) \cdot \left(\delta - \beta - \frac{l_f\dot{\psi}}{v_{cg}}\right) \cdot \sin\delta \\
& - l_r \cdot (c_{RL} + c_{RR}) \cdot \left(-\beta + \frac{l_r\dot{\psi}}{v_{cg}}\right) \\
& + \frac{l_t}{2} \cdot (F_{L,RR} - F_{L,RL})\Big\}
\end{aligned} \tag{4.31}
$$

4.3 Linearized Vehicle Model

Because of the many mathematical tools available for linear systems, valuable insight into the behavior of a non-linear system around an operating point can often be investigated by the linear approximation in the point. The nonlinear model (4.27) is linearized using first order Taylor series expansion

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}_0,\boldsymbol{u}_0) + \frac{\partial f(\boldsymbol{x},\boldsymbol{u})}{\partial \boldsymbol{x}}\Delta\boldsymbol{x} + \frac{\partial f(\boldsymbol{x},\boldsymbol{u})}{\partial \boldsymbol{u}}\Delta\boldsymbol{u} \tag{4.32}$$

Defining the perturbation

$$\boldsymbol{x} = \boldsymbol{x}_0 + \Delta\boldsymbol{x} \tag{4.33}$$

the resulting linear system takes the form

$$\Delta\dot{\boldsymbol{x}} = \boldsymbol{A}\Delta\boldsymbol{x} + \boldsymbol{B}\Delta\boldsymbol{u} \tag{4.34}$$

where the matrices A and B are given as

$$\boldsymbol{A} = \begin{bmatrix} \frac{\partial \dot{v}_{cg}}{\partial v_{cg}} & \frac{\partial \dot{v}_{cg}}{\partial \beta} & \frac{\partial \dot{v}_{cg}}{\partial \dot{\psi}} \\ \frac{\partial \dot{\beta}}{\partial v_{cg}} & \frac{\partial \dot{\beta}}{\partial \beta} & \frac{\partial \dot{\beta}}{\partial \dot{\psi}} \\ \frac{\partial \ddot{\psi}}{\partial v_{cg}} & \frac{\partial \ddot{\psi}}{\partial \beta} & \frac{\partial \ddot{\psi}}{\partial \dot{\psi}} \end{bmatrix} \tag{4.35}$$

$$\boldsymbol{B} = \begin{bmatrix} \frac{\partial \dot{v}_{cg}}{\partial F_{L,FL}} & \frac{\partial \dot{v}_{cg}}{\partial F_{L,FR}} & \frac{\partial \dot{v}_{cg}}{\partial F_{L,RL}} & \frac{\partial \dot{v}_{cg}}{\partial F_{L,RR}} & \frac{\partial \dot{v}_{cg}}{\partial \delta} \\ \frac{\partial \dot{\beta}}{\partial F_{L,FL}} & \frac{\partial \dot{\beta}}{\partial F_{L,FR}} & \frac{\partial \dot{\beta}}{\partial F_{L,RL}} & \frac{\partial \dot{\beta}}{\partial F_{L,RR}} & \frac{\partial \dot{\beta}}{\partial \delta} \\ \frac{\partial \ddot{\psi}}{\partial F_{L,FL}} & \frac{\partial \ddot{\psi}}{\partial F_{L,FR}} & \frac{\partial \ddot{\psi}}{\partial F_{L,RL}} & \frac{\partial \ddot{\psi}}{\partial F_{L,RR}} & \frac{\partial \ddot{\psi}}{\partial \delta} \end{bmatrix} \tag{4.36}$$

The equations for each element in these matrices are found in appendix A.

Chapter 5

The LocalBug Simulator

The LocalBug simulator implemented in Jakobsen (2010) is used to simulate the controller developed in chapter 6. The simulator individually calculates the forces at in the ground-tire interface for each wheel, and drag forces acting on the vehicle chassis. Computation of the wheel forces is based on the friction model presented in Kiencke and Nielsen (2005). The advantage of the friction model is that it does not require tire specific data which may be hard to obtain. While the simulator showed good results, some elements were not sufficiently explored. The following chapter gives improvements to the simulation model and provides verification of the accuracy of the simulator against logged data.

5.1 Status of the LocalBug Simulator

Some possible shortcomings in the simulator were mentioned in Jakobsen (2010). Among these were

- The motor model was only approximated by a transfer function. Since the motor torque is not constant when the vehicle is subject to various loads, the simulator should include a dc motor model.
- LocalBug can be configured as front, rear or four wheel drive, in addition to enabling or disabling differential lock on each axle. The simulator should be able to reflect the various settings of LocalBug.

- The simulation outputs are the real vehicle states. Since noise will be present in any practical implementation, the simulator should include noise on the output variables.
- The simulator was not verified against real-life data and no measure of the fidelity of the simulator has been performed.

These drawbacks are discussed and implemented in this chapter. In addition, some of the vehicle parameters that previously were estimated, have been measured. This includes the vehicle moment of inertia in yaw and the wheel moment of inertia around the rolling axis.

5.2 Motor and Driveline Model

The simplified motor and driveline model has been replaced by a dc motor model. The assumptions of a frictionless and rigid driveline from Jakobsen (2010) are also taken here.

The differential equations for a dc motor are (Egeland and Gravdahl 2002)

$$L_a \frac{di_a}{dt} = -R_a i_a - k_E \omega_m + u_a \tag{5.1}$$

$$I_m \dot{\omega}_m = k_t i_a - T_{\mathrm{L}} \tag{5.2}$$

An explanation of the notation is given in table 5.1.

Table 5.1: Notation used in the dc motor equations.

Symbol	Description
i_a	Armature current [A]
u_a	Armature voltage [V]
L_a	Armature inductance [H]
R_a	Armature resistance [Ω]
k_E	Back EMF constant [kgm^2/s]
ω_m	Rotational velocity of the motor shaft [rad/s]
I_m	Inertia of the motor armature [kgm^2]
k_t	Torque constant [Nm/A]
T_L	Load torque [Nm]

Table 5.2: Estimated values for the DC motor parameters.

Coefficient	Value
L_a	0.001
R_a	0.8
k_t	0.17
k_e	0.17

Equation (5.1) is modeled in the *DC Motor model*-block, with voltage u_a as the input and torque $k_t i_a$ as the output. The implementation of (5.2) is further discussed in section 5.3.

Assuming the Flux Torque 2200kv motor installed in LocalBug has a voltage constant $k_v = 2200\, rpm/v$ as implied in the name, the torque and back EMF constants, k_t and k_E, can be calculated. First, the value is converted to SI-units

$$k_v = \frac{2200 \frac{rpm}{v}}{60s \cdot 2\pi} = 5.84 \frac{\frac{rad}{s}}{v} \tag{5.3}$$

The relationship between the torque and back EMF constant, and the voltage constant is given as

$$k_t = k_E = \frac{1}{k_v} = 0.17 \tag{5.4}$$

The remaining coefficients were estimated. Some tuning was performed to adjust the simulation to achieve a reasonable acceleration and a top speed of approximately 100 km/t, according to the specification from the manufacturer (HPI-Racing n.d.). Because the real motor constants are unknown, only an approximation that fits the test data could be found. A thorough testing of the motor itself could be conducted to more accurately determine these parameters. Figure 5.1 shows the response of the dc motor model to a step in throttle. The estimated values for the motor coefficients are listed in table 5.2.

5.3 Wheel Drive Configuration

The LocalBug simulator has been modified to make it possible to select front wheel drive, rear wheel drive or four wheel drive from the simulation settings file. In addition, differential locking can be enabled or disabled for all drive

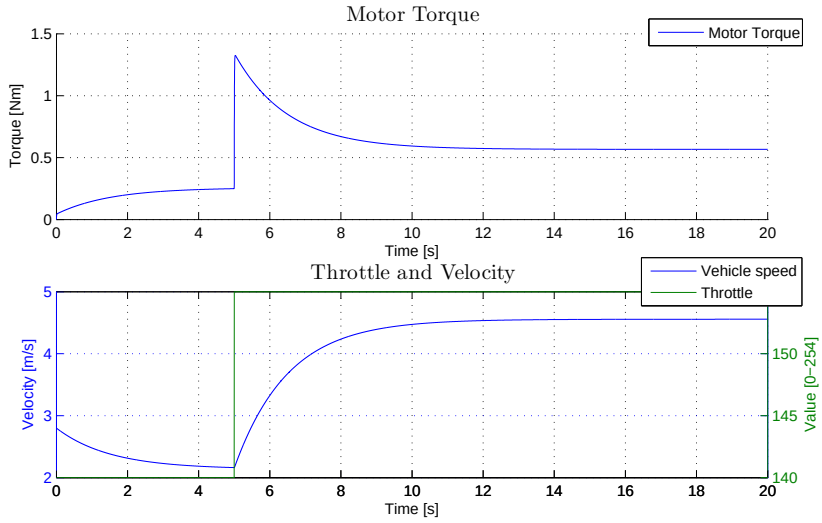

Figure 5.1: Simulation results showing the produced motor torque resulting from a step in throttle.

configurations. The differential allows two wheels on the same axle to rotate with different speeds to avoid unnecessary wear and tear of the tires. This occurs most commonly in curves when the wheels on the outer side has a longer distance to travel, and instead of being dragged sideways are allowed to rotate slightly faster than the inner wheel.

The following equations are based on the assumptions of a frictionless and rigid driveline. Remembering the torque balance for each wheel is expressed as (Kiencke and Nielsen 2005)

$$I_w\dot{\omega}_w = T_{drive} - T_{break} - r_{eff}F_{fric,x} \tag{5.5}$$

where T_{drive} is the torque from the motor and $r_{eff}F_{fric,x}$ is the torque resulting from the friction forces acting between the road surface and the tires. T_{break} is the breaking torque which is always zero, as there are no dedicated breaks available on LocalBug.

Without losses in the driveline, the load on the motor is equal to the driving torque in (5.5). Solving for the driving torque and substituting (5.5) into (5.2) gives the torque balance for the combined system consisting of motor and wheel

$$I_m\dot{\omega}_m + I_w\dot{\omega}_w = T_{drive} - r_{eff}F_{fric,x} \tag{5.6}$$

Since the driveline is rigid it follows that $\omega_m = \omega_w$, which gives

$$(I_m + I_w)\,\dot{\omega}_w = T_{drive} - r_{eff}F_{fric,x} \tag{5.7}$$

In the case of an open differential configuration where the wheel speeds are allowed to differ, the motor speed is calculated as the average of the angular velocity of the driving wheels. This is justified since the construction of the differential symmetrically distributes the rotational velocities to each wheel on the same axle.

Open differential With an open differential, the driving torque is evenly distributed to all driving wheels. Accounting for the moment of inertia of the motor, as discussed in section 5.2, the torque balance for each driving wheel becomes

$$\left(I_w + \frac{I_m}{n}\right)\dot{\omega}_w = \frac{T_{drive}}{n} - r_{eff}F_{fric,x} \tag{5.8}$$

where n is the number of driving wheels. For the free rolling tires, the only torque present arises from the friction forces acting at the wheel-ground contact point

$$I_w \dot{\omega}_w = -r_{eff} F_{fric,x} \tag{5.9}$$

A problem that often arises when driving on a slippery surface is that one wheel is driven onto an ice patch and looses traction, while the other wheel is still able to provide some driving force. Due to the mechanical construction of the differential, the maximum driving torque is limited to the torque the wheel with the least traction is able to transfer to the ground. This model does not take into account the effects experienced when one wheel is spinning while the other retains traction. Because the simulator is not capable of simulating different road conditions for each wheel, this simplified model is still considered sufficient.

Locked differential When the differential lock is enabled, the rotational speeds for the wheels connected to the same axle are forced equal. In this case, the torque balance for each axle is expressed as

$$\left(2I_w + \frac{I_m}{m}\right)\dot{\omega}_w = \frac{T_{drive}}{m} - r_{eff}\left(F_{fric,x,left} + F_{fric,x,right}\right) \tag{5.10}$$

where m is the number of axles connected to the motor. The free rolling tires are modeled as in (5.9) where the driving torque is set to zero.

Equations (5.8), (5.9) and (5.10) are implemented in the *Wheel angular velocity*-function block which calculates the wheel rotational velocities. A simulation where LocalBug is accelerated and simultaneously is turning to the left has been performed. The simulation is conducted with rear wheel drive, and one can clearly see that the wheels on each side of the vehicle are rolling with approximately the same speed. The driving wheels have a slightly larger velocity than the free rolling front tires, which is to be expected as these tires provide the driving force required to overcome air drag and rolling resistance. The simulation result is seen in figures 5.2 and 5.3.

5.4 Measurement Noise Model

The difference between the exact motion of LocalBug and the measured variables is due to errors in the measurements and noise. Accurately modeling the noise in

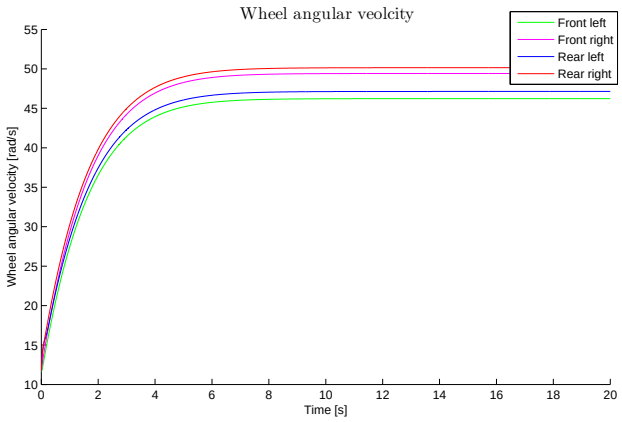

Figure 5.2: Simulation of driving in a curve with open differential. The wheels on each side of the vehicle are rolling with approximately the same speed. The rear wheels, which are connected to the motor, have a slightly larger angular velocity than the free rolling front tires. This occurs because the rear wheels are producing lateral force to oppose air drag and rolling resistance.

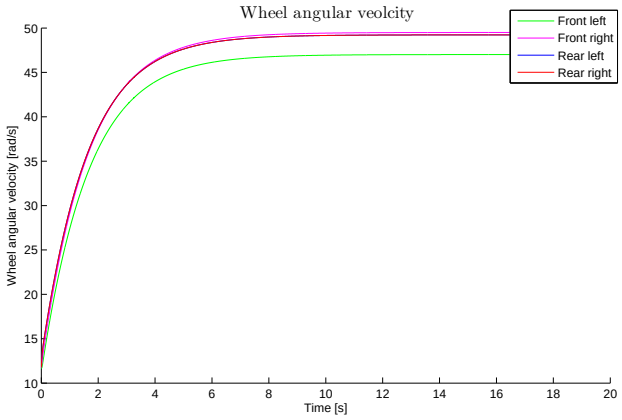

Figure 5.3: Simulation of driving in a curve with locked differential. Notice that both the rear wheels are forced to drive with the same speed while the free rolling front wheels have different rotational velocities.

Table 5.3: Xsens MTi-G Performance Specification. (Xsens 2010)

	Roll	Pitch	Yaw	Position
Static accuracy	< 0.5 deg	< 0.5 deg	< 1 deg	2.5 m CEP
Dynamic accuracy	1 deg RMS	1 deg RMS	1 deg RMS	

Table 5.4: Kalman filter operating scenarios.

Setting	Active sensors
General Purpose	IMU, GPS, Barometer
Aerospace	IMU, GPS, Magnetometer, Barometer
Automotive	IMU, GPS, Barometer (incl. non-holonomic constraint)
Marine	IMU, GPS, Magnetometer

SIMULINK was considered difficult, since real world noise is not perfect white noise. It was therefore decided to record a real data sample from the sensor when at rest. The noise series will then be added to the perfect output from the SIMULINK model to simulate realistic sensor output. A real data series will also capture non-parametric uncertainties in the measurements.

The Xsens MTi-G sensor is an integrated GPS and Micro Electro-Mechanical System (MEMS) IMU and consists of three accelerometers and gyroscopes for three dimensional sensing, a magnetometer and a static pressure sensor. An integrated Kalman filter provides position, orientation and velocity estimates. The performance of the Xsens MTi-G is stated in table 5.3.

The on-board Kalman filter can be configured to one of the environments listed in table 5.4. The scenarios decide which sensors are used as inputs to the Kalman filter. The MTi-G is accompanied by MT Manager, a software tool for calibration, configuration and logging of data.

5.4.1 Data Recording

The Xsens MTi-G unit was mounted inside an Audi A4 with the GPS antenna fixed on top of the roof, as shown in figure 5.4. To record as accurate data as possible, the magnetometer was calibrated using the MT Manager software. As the automotive scenario setting is based on a non-holonomic assumption, the integrated Kalman filter was configured to the aerospace setting instead. According to the manual, the Kalman filter requires minimum one minute to

Figure 5.4: Test setup for logging of IMU data.

initialize and the estimates continues to improve up to 15 minutes after power-on (Xsens 2010). To proper initialize the filter, a longer drive was conducted before data were recorded.

After the Kalman filter was initialized, a data series was logged while standing still using the MT Manager software provided with the sensor. The data were then exported to Matlab. Bias on the measurements were removed in Matlab by subtracting the average offset from zero to only capture the high frequency noise, and the position were converted from longitude-latitude-altitude to the local tangent plane coordinate system that is compatible with the simulator output.

A Matlab file containing noise data for each output variable was created and is added to the SIMULINK signals during simulations.

5.5 Measurement of Wheel Moment of Inertia

The moment of inertia (MoI) of the wheel is considered important because it affects the acceleration of the wheel, subject to a given motor torque. The

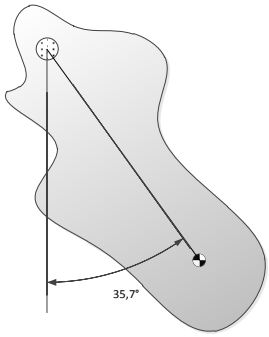

Figure 5.5: Principle sketch of the compound pendulum.

acceleration is integrated to rotational velocity and then used to calculate wheel slip, which, in turn, is the basis for calculation of the friction forces.

A description of the method used to measure the moment of inertia of the wheels is presented below, followed by the experimental setup and test results.

The Compound Pendulum The compound pendulum is similar to a simple pendulum, except that there are no restrictions on the mass distribution. In a simple pendulum, the mass lies entirely in the bob and the string is massless, while the compound pendulum can consist of any arbitrary object. Given a small offset, it will oscillate freely with a constant period as a simple harmonic oscillator. By measuring the period of the system, the moment of inertia about the pivot point can be found.

The well-known torque balance for the pendulum is

$$I_M\ddot{\theta} + mgl\sin\theta = 0 \tag{5.11}$$

where I_M is the measured moment of inertia, m the object mass, g the gravitational acceleration, l the distance between the pivot point and the center of percussion, which is assumed to be close to the center of mass, and θ the angle between the line connecting the pivot point and center of percussion and the vertical. The parallel axis theorem can be used to calculate the moment of inertia about the center of gravity, I_y:

$$I_M = I_{CG} + ml^2 \tag{5.12}$$

Table 5.5: Measured and estimated tire data.

Tire	Mass	Radius	Length	Expected MoI
Tarmac Buster	0.400 kg	0.085 m	0.06 m	$0.0014 - 0.0022$ kgm^2
Badlands Pro-line	0.495 kg	0.0875 m	0.06 m	$0.0019 - 0.0028$ kgm^2
Terrapin	0.250 kg	0.0875 m	0.04 m	$0.0011 - 0.0014$ kgm^2

Using small angle approximation $\sin\theta = \theta$, the system takes the form

$$\ddot{\theta} + \omega_n^2 \theta = 0 \tag{5.13}$$

The period of oscillation is

$$t = \frac{2\pi}{\omega_n} = 2\pi\sqrt{\frac{I_M}{mgl}} \tag{5.14}$$

Solving for the moment of inertia and using the parallel axis theorem gives

$$I_{CG} = \left(\frac{t}{2\pi}\right)^2 mgl - ml^2 \tag{5.15}$$

5.5.1 Experimental Setup

A screwdriver was fastened in between two shelf sections and the test object was hung upon the screwdriver. The distance from the pivot point to the center of the wheel was measured. The measured length of the pendulum, and the wheel mass and radius, taken from Jakobsen (2010), is given in table 5.5.

Three sets of wheels are available for LocalBug, the Tarmac Buster, Badlands Pro-line and Terrapin tires. The period of oscillation were measured, and the moment of inertia calculated, for each set of tires.

5.5.2 Test Result

After giving the wheel an initial oscillation, thirty oscillations were timed and the average period calculated. The experiment was repeated three times for each tire. Equation (5.15) was used to calculate the moments of inertia:

Figure 5.6: Construction of the Tarmac Buster wheel.

Figure 5.7: Experimental setup for measurement of the wheel moment of inertia.

Table 5.6: Experimental results for the wheel's moment of inertia.

Tire	Trial	Oscillations	Time	Average time
Tarmac Buster	1	30	21.6 s	0.720 s
	2	30	21.4 s	0.713 s
	3	30	21.5 s	0.717 s
	Average			**0.717** s
Badlands Pro-line	1	30	21.6 s	0.720 s
	2	30	21.7 s	0.723 s
	3	30	21.7 s	0.723 s
	Average			**0.722 s**
Terrapin	1	30	22.9 s	0.763 s
	2	30	22.9 s	0.763 s
	3	30	22.7 s	0.757 s
	Average			**0.761 s**

$$I_{tarmac} = 0.0016 \text{ kg}m^2 \tag{5.16}$$
$$I_{badlands} = 0.0021 \text{ kg}m^2 \tag{5.17}$$
$$I_{terrapin} = 0.0012 \text{ kg}m^2 \tag{5.18}$$

Considering the wheel as a cylinder, the moment of inertia can be calculated as (Haugan 1992)

$$I_{cylinder} = \frac{mr^2}{2} \tag{5.19}$$

where r is the radius of the cylinder. Observing the construction of the wheels (see figure 5.6), it is clear that most of the mass is located in the rim and the tread. Thus, a hollow cylinder might be a more appropriate approximation:

$$I_{hollow\,cylinder} = \frac{m(r_{outer}^2 + r_{inner}^2)}{2} \tag{5.20}$$

where r_{inner} and r_{outer} is the inner and outer radius, respectively. However, the weight of the wheel hub is uncertain, and can very well be significant. It is therefore expected that the wheel moment of inertia lies in between the minimum and maximum value for the two approximations, given in table 5.5. Hence, the

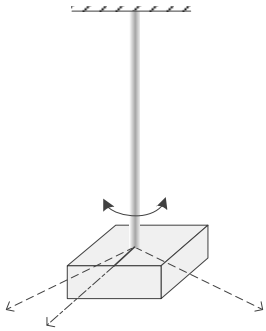

Figure 5.8: Principle sketch of the torsion pendulum.

measured values seem very reasonable since the moment of inertia for all tires are within the expected range.

5.6 Measurement of Vehicle Moment of Inertia

The moment of inertia around the vehicle z-axis is the basic physical quantity determining the yawing motion from a set of friction forces. As seen in (4.10), the yaw rate has a significant influence on the sideslip dynamics and is therefore vital for producing accurate simulations. The formula currently used for calculating the moment of inertia in yaw is that of a rectangular box which assumes uniform mass distribution. Since the steering servo and engine is located at the very front and rear of LocalBug, the moment of inertia is assumed to be slightly larger than the current estimate[1] of

$$I_z = \frac{m \times (l^2 + b^2)}{12} = \frac{7 \times (0.56^2 + 0.38^2)}{12} = 0.2672 \text{ kg}m^2 \tag{5.21}$$

Because pitch and roll dynamics is ignored in the simulator, the moment of inertia around these axes were not measured. In the following sections the test method is explained, followed by the experimental setup and test results.

[1] The values for mass, length and width of LocalBug is taken from Jakobsen (2010)

The Torsion Pendulum This method for calculating the moment of inertia is described by Kooijman (2006). The torsion pendulum is a vertical steel rod suspended in the ceiling and connected in the lower end to the object to be measured using a stiff coupling, see illustration in figure 5.8. By inducing an oscillatory motion about the longitudinal axis of the rod, the system will oscillate with a specific frequency depending on the moment of inertia of the connected object and the torsional rigidity of the rod. Timing the period of the system makes it possible to calculate the unknown moment of inertia.

Assuming that the torque resulting from the rod being twisted is linear with respect to the displaced angle, one can write

$$T_{rod} = \frac{GI_P}{L}\theta \tag{5.22}$$

where G is the shear modulus of elasticity , I_P is polar moment of inertia and L the length of the rod. The torque balance for the complete system is

$$I_M\ddot{\theta} + T_{rod} = 0 \tag{5.23}$$

where I_M is the moment of inertia to be measured. Substituting (5.22) into (5.23) gives

$$\ddot{\theta} + \frac{GI_P}{I_M L}\theta = 0 \tag{5.24}$$

$$\ddot{\theta} + \omega_n^2\theta = 0 \tag{5.25}$$

which is a simple harmonic oscillator with natural frequency

$$\omega_n = \sqrt{\frac{GI_P}{I_M L}} \tag{5.26}$$

The period for one oscillation is then

$$t = \frac{2\pi}{\omega_n} = 2\pi\sqrt{\frac{I_M L}{GI_P}} \tag{5.27}$$

Rearranging the last equation gives

$$I_M = \left(\frac{t}{2\pi}\right)^2 \frac{GI_P}{L} \tag{5.28}$$

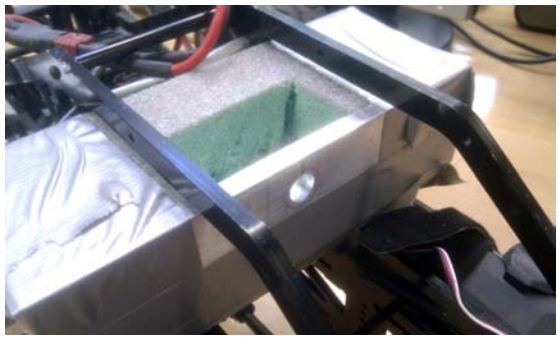

Figure 5.9: Preparing LocalBug for measurement of the moment of inertia. The steel bar is fitted in the LocalBug frame.

For steel, $G \approx 77.5 \times 10^9 Pa$ (Haugan 1992), and the value for I_P can be calculated as

$$I_P = \frac{\pi d^4}{32} \tag{5.29}$$

where d is the diameter of the rod.

5.6.1 Experimental Setup

The measurement apparatus consisted of a 0.696 m long and 3 mm thick steel rod clamped to a sheet in the upper end, and a bracket to fasten the object to be measured in the lower end. The top end was fastened to the roof beams in Forsøkshallen.

A steel bar was placed abeam the vehicle frame in the center of gravity in the XY-plane to avoid bending the rod during the test, and a small piece of Divinycell was cut to act as a standoff to stabilize the bar. The frame, standoff and steel bar was taped together to prevent it from slip out of position (figure 5.9). LocalBug was then bolted to the torsion rod as shown in figure 5.10. During the test, LocalBug was equipped with wheels and batteries, but without the instrumentation as this came in conflict with the mounting of the rod to the steel bar.

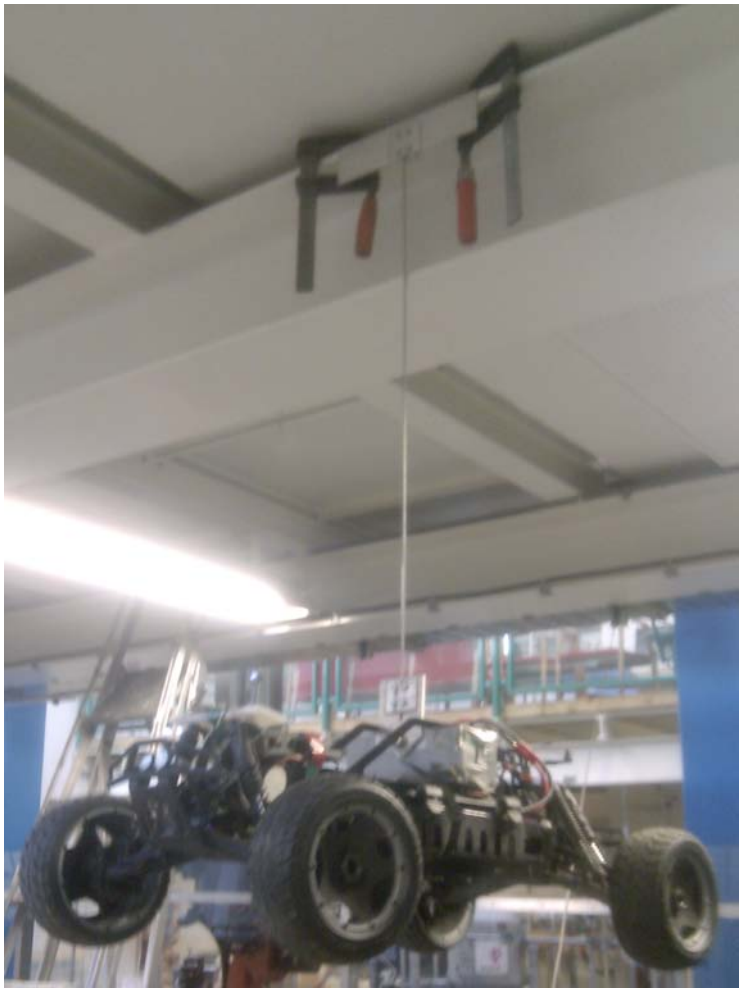

Figure 5.10: Experimental setup for measurement of the moment of inertia around the vehicle z-axis.

5.6.2 Calibration of testing apparatus

Divinycell is a lightweight material, and close to the center of gravity it is assumed to have negligible influence on the moment of inertia. The bracket, however, is heavier, and the rod may have a different shear modulus of elasticity than the nominal value for steel. A calibration test was performed on the testing apparatus by Ånnestad (2010) to eliminate these factors. Equation (5.28) can be rearranged into

$$I_M \left(\frac{2\pi}{t}\right)^2 = \frac{GI_P}{L} = K \tag{5.30}$$

which enables the calculation of the rod spring constant K. Using the nominal values for steel, the spring constant was calculated to be $K_{nominal} = 0.8855$.

A thin metal rod of mass $m = 2.108$ kg and length $l = 0.9515$ m was used for calibration. The period of oscillation was found to be $t = 2.695$ s (Ånnestad 2010). Using the formula for moment of inertia for a rod, $I_M = 0.1590$ kgm^2. The calibration resulted in

$$K_{calibrated} = 0.8645 \tag{5.31}$$

which was used for the test. The nominal and calibrated values only differ with 2.5 percent, which indicates that the assumption of a linear relationship between torque and displaced angle holds quite well.

5.6.3 Test Result

LocalBug was given an impulse to start oscillating in the vertical plane. The time it took for a number of oscillations was recorded and the test repeated five times to eliminate timing errors. The test results are found in table (5.7).

Using the data obtained, the mass moment of inertia in yaw was calculated as

$$I_z = \left(\frac{4.55}{2\pi}\right)^2 \times 0.8648 = 0.4535 \text{ kg}m^2 \tag{5.32}$$

The result shows that the moment of inertia is almost twice the estimate in (5.21).

Table 5.7: Experimental results for LocalBug's moment of inertia around the z-axis.

Trial	Number of Oscillations	Time [s]	Average time [s]
1	10	45.5	4.55
2	10	45.4	4.54
3	5	22.7	4.54
4	10	45.6	4.56
5	5	22.9	4.58
Average			4.55

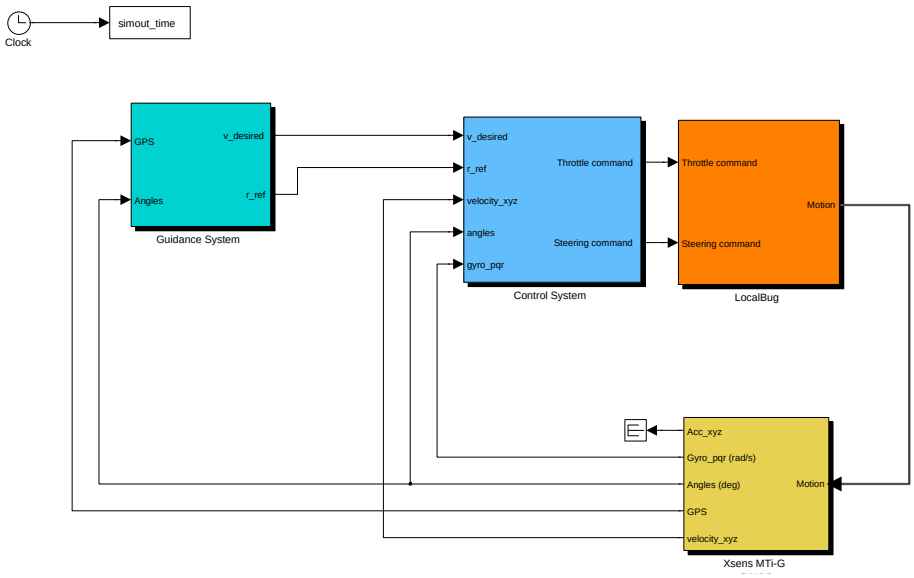

Figure 5.11: Overview of the LocalBug simulator with noise model, guidance system and control system.

5.7 Using the LocalBug Simulator

The LocalBug simulator is located in the file *"carsim.mdl"*. Figure 5.11 illustrates the contents of the simulator model file. The inputs are steering and throttle commands in the range between 0-254, which corresponds to the autopilot interface described in Wigestrand et al. (2010b). The outputs are position, heading, yaw rate, inertial velocities and accelerations given according to the right handed north-west-up convention.

All simulator settings are located in the file *"init_sim.m"*. This includes simulation settings such as road condition and noise, initial conditions and vehicle and motor parameters. When noise is enabled, the noise data must be stored in the same directory as the simulator. These files are named *"noise_acc.mat"*, *"noise_ang.mat"*, *"noise_gyr.mat"*, *"noise_pos.mat"* and *"noise_vel.mat"*.

The simulator stores many variables to the workspace during simulation. All the variable names start with *"simout_"* followed by the name of the variable.

Two files are created to plot the simulation results. The first file, *"show_movie.m"*, shows a movie-like motion of the vehicle during the simulation. The second file, *"plot_figures.m"*, creates an overview of the simulation, which is similar to the movie. In addition, plots of velocities, accelerations, wheel slip and actuator inputs are created. It is possible to save the movie as an .mpg file and the figures as .fig and .eps files by executing the *"save_movie.m"* and *"save_figures.m"* files.

5.8 Validation of the Simulator

To validate the simulator, a series of three simulations is compared to log data. The simulator is initialized with the position, velocity and orientation from the first measurement in each test case. Since the actuator commands were logged, the exact same inputs are given to the simulator. The simulator parameters were tuned to increase the correspondence between simulation and raw data. Because the objective is to test the fidelity of the simulator, noise is not added to the simulator output to simplify the validation process.

In the first test the longitudinal dynamics of the simulator will be examined by using the throttle as input. Secondly, the yaw dynamics is investigated using the steering angle as input and holding the throttle constant. Finally, the coupled longitudinal-lateral dynamics are verified by simultaneous throttle and steering input.

Table 5.8: Tuned values for the air drag and rolling resistance coefficients.

Coefficient	Value
C_{air}	1.07
$C_{rolling}$	0.02

To achieve the best possible match between the simulator and real data, it was necessary to adjust the air drag and rolling resistance coefficients. Initially, the simulated velocity was too high and the coefficient values were therefore increased. The aerodynamic drag friction coefficient was approximated by assuming LocalBug is a three dimensional cube and using the drag coefficient provided in White (2008). The rolling resistance coefficient was then tuned accordingly. The values for the tuned coefficients are given in table 5.8.

5.8.1 Raw Data

The simulator is validated against data recorded in Kongsberg in the summer of 2010 by the summer interns. At the time of the testing, LocalBug was configured with real wheel drive with open differential. The tests were conducted on asphalt, which is a high friction road surface.

Since the actuator inputs are pulse width modulated servo signals, the logged data is an integer representing the length of the high level of the pulse. At 32Mhz clock frequency and a prescale value of 64, this results in a minimum value between 500 for 1ms, and a maximum value of 1000 for 2ms pulse width. This number is converted into an integer between 0 and 254 to be used as input to the simulator.

The GPS output data is given in longitude, latitude and altitude coordinates. Because the simulator is built around a north-west-up coordinate system, the raw data were converted using the algorithm suggested by Wenstad (2010).

There are some uncertainties present regarding the alignment of the IMU compared to the vehicle body frame during the test, and the state of the source code used for logging. This uncertainty becomes visible through some inconsistencies in the data. The IMU was supposedly facing the rear of LocalBug to accommodate the cables interfacing the IMU. This should not pose any problems as the data can be rotated to the vehicle frame. However, the sign of the inertial velocities are not consistent with the change of position measured by the GPS. The sign of the velocity in the y-direction was changed to make sense

of the data. In addition, the velocities are by default given in centimeters per second. The x and y velocity seems to be scaled to meters per second, while the z velocity is not. Thus, this correction is made before utilizing the data.

The data were imported to Matlab using the script embedded in appendix B.

5.8.2 Case 1: Straight-line Acceleration and Breaking

The first case is a straight-line acceleration and breaking test to verify the longitudinal dynamics of the simulator. From zero velocity, full throttle is applied. After reaching maximum velocity, the throttle is swapped to reverse to initiate a rapid deceleration.

The logged data shows that the velocity is drifting, especially in the lateral direction. However, figure 5.12 suggests that the simulated velocity tracks the logged longitudinal velocity well. The plot shows a large initial lateral velocity which in the simulator is quickly damped. The logged longitudinal velocity seems to lag behind the actuator command by up to 1.5 seconds. This indicates that the timestamp of the control signals is not synchronized with the sensor data. At time $t = 4$ when the breaking begins, this is especially evident. It is unlikely that the speed will remain constant when the throttle is reversed, as seen in figure 5.12. During breaking it is observed that the small steering input at $t = 4.5$ is enough to cause the simulation to skid 180 degrees. This spin causes the discrepancies in sideslip and heading at the end of the test.

Figure 5.13 shows the calculated sideslip angle. The straight driving test should not produce any significant sideslip, but the drifting velocities causes this value to be strongly erroneous. The sideslip is calculated from the longitudinal and lateral velocity according to (1.1), and is therefore sensitive to errors in the velocities.

As seen in figure 5.14, the yaw rate seems to be correlated in the two data sets, accounting for the indicated offset in time between logged and simulated data. When the simulation spins around, the yaw rate and heading no longer coincides with the logged data.

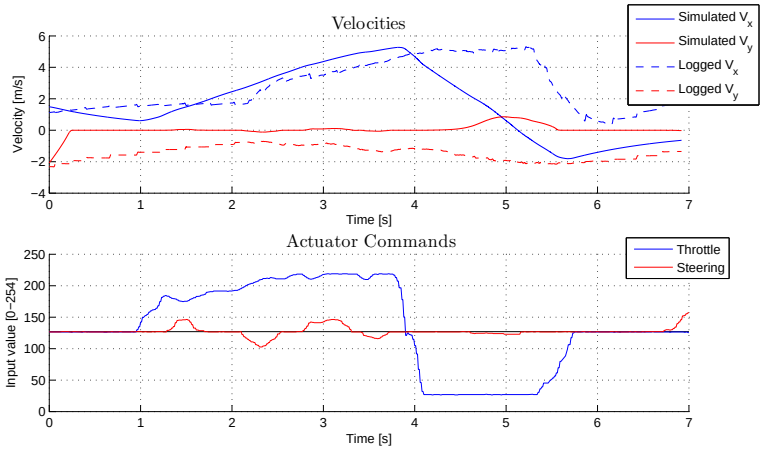

Figure 5.12: Velocity and actuator commands during straight-line acceleration and breaking. It is seen that the simulated velocity in the x-direction tracks the logged longitudinal velocity well.

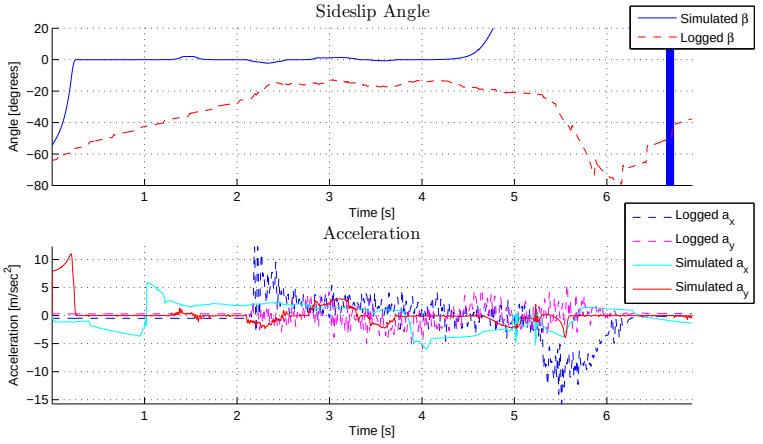

Figure 5.13: Sideslip angle and acceleration during straight line acceleration and breaking. The drifting velocities are causing the logged sideslip angle to be erroneous.

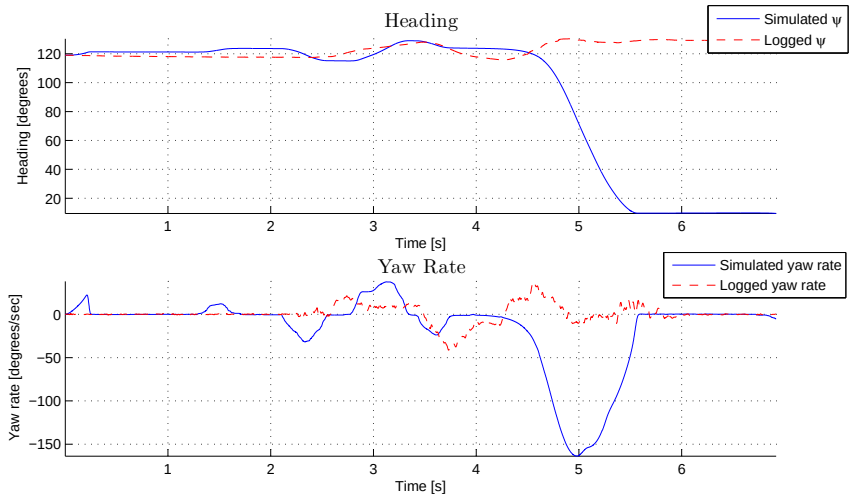

Figure 5.14: Heading and yaw rate during straight line acceleration and breaking. The yaw rates appears coherent up $t = 4.5$, but with a time lag between logged and simulated data that might originate from logging issues.

5.8.3 Case 2: Slalom Steering with Constant Velocity

In the second case, the lateral dynamics of the system are investigated. This is clearly a challenging test because of the large and rapid control inputs. The test consists of a slalom-like maneuvering with constant velocity. From the heading plot in 5.17 it is seen that the test can be divided in two sub-tests. The first test starts at approximately $t = 6$ seconds when the steering inputs begin. At time $t = 17$ seconds, the driver stops, turns LocalBug around and drives back to the starting point.

In the actuator commands plot in figure 5.15, it seems like the driver was not able to keep constant throttle while turning. This can attributed to difficulties in accurately controlling the joystick on the controller. Even though the logged velocities are oscillating, the trend shows that the simulated velocities are close, especially in the first half of the test.

Figure 5.16 shows the calculated sideslip angles. The logged and simulated case shows a better match than during the first test. Notice the sideslip peaks in the logged data series. This occurs when LocalBug spins around, which seems to

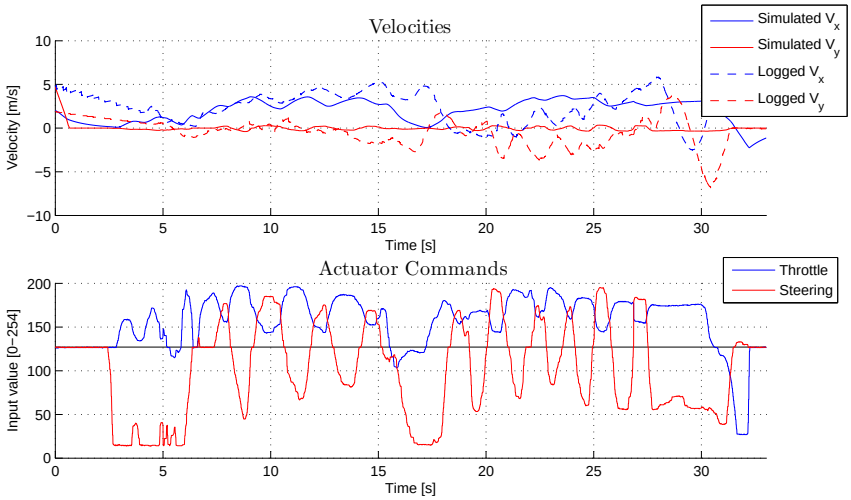

Figure 5.15: Velocity and actuator commands during slalom steering. It seems difficult for the driver to keep the throttle fixed.

be the case at the end of the test. One can see the driver turning and throttling simultaneously, followed by a drop in longitudinal velocity and an increase in lateral velocity. When the velocity in longitudinal direction is zero and the lateral velocity peaks, LocalBug is sliding sideways, halfway through the spin. The simulator does not capture this skid, although a spin is observed at the end when breaking. This also happened in the first test case and might indicate that the simulator is not accurately reproducing reality during hard breaking. At time $t = 18$ to $t = 20$ the large sideslip angles in the logged data series is contributed to uncertainties in the velocities. At this time, the logged velocities are low and noise can easily corrupt the sideslip calculation.

The simulated yaw rate is smaller in magnitude, which may indicate that the steering servo provides a higher steering angle than the simulator for the same input value. In between the two sets, the simulated velocity is very low at time $t = 17$, hence the small change in heading compared to what is observed in the log data. Apart from that, the simulated heading and yaw rate seems to be in agreement with logged data, as figure 5.17 indicates.

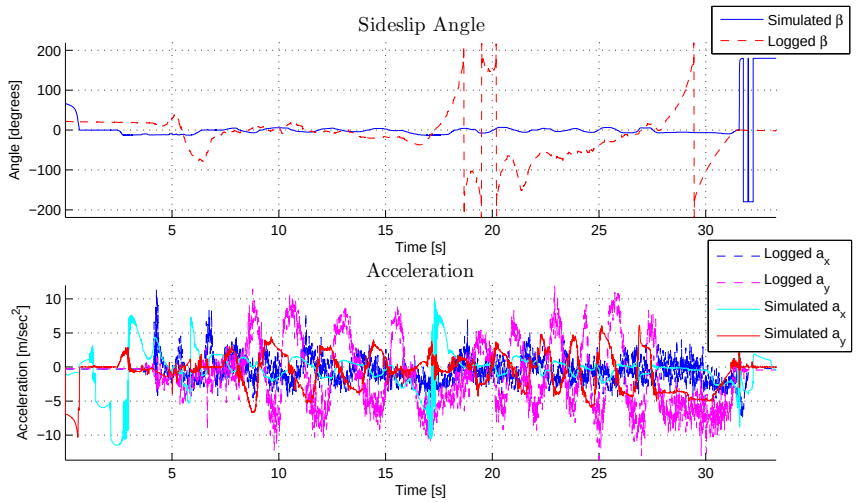

Figure 5.16: Sideslip angle and acceleration during slalom steering.

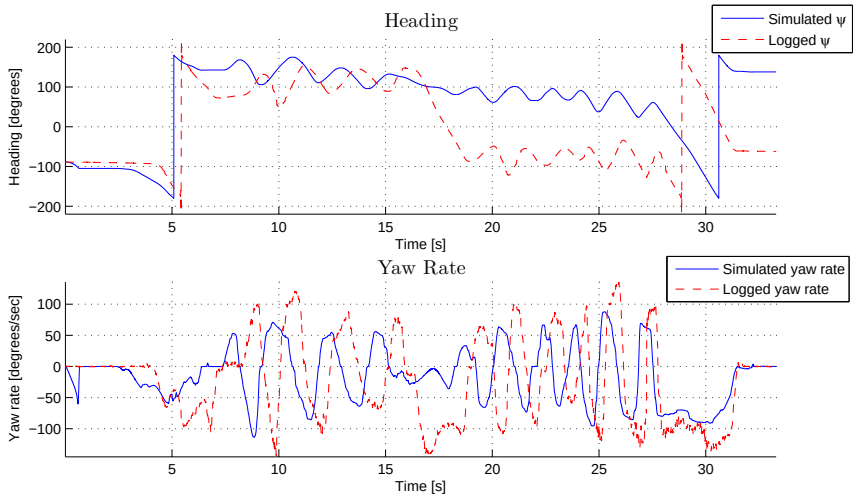

Figure 5.17: Heading and yaw rate during slalom steering.

5.8.4 Case 3: Constant Steering with Increasing Velocity

The final test is a combined lateral and longitudinal stability test where LocalBug drives in a constant circle with increasing velocity.

Figure 5.18 shows good correspondence between the logged and simulated velocities up to time $t = 25$ when a combination of breaking and steering causes the simulation model to skid. An explanation may be that the wheel slip for the rear wheels are not modeled accurately during hard breaking. If the longitudinal wheel slip is calculated too large and the friction force saturates, a loss of stabilizing lateral force occurs.

The steering input seems to be saturated at a value of 200, which strengthens the observations in the second test case where it is indicated that the steering servo might be providing a steeper turn angle at a given steering command.

The simulation shows a stable sideslip angle during this test, shown in figure 5.19, while the sideslip angle calculated from the log data is varying as a result of the oscillating velocities. It is seen that the mean value of the logged accelerations are very close to the simulated values. The yaw rate and heading in figure 5.20 is accurately reproduced in the simulator.

5.9 Discussion of Results

Looking at the data sets, the logged velocities are not satisfactory. Because the velocities are oscillating, and in some tests constantly changes sign, it would be difficult to use these data in a feedback controller. Because the measurements are so inaccurate, one might suspect that raw data is logged and not the filtered data. Based on the performance specification, the MTi-G should be able to provide more accurate data. In order to acquire better measurements in the future, one should ensure that

- The sensor is located close to the center of gravity of LocalBug and that the sensor axes are aligned with the vehicle axes.
- The magnetometer is calibrated. The MT Manager software is capable of performing a magnetometer tuning. This should be done after the IMU has been placed inside LocalBug for highest accuracy.
- The filter has had time to initialize, this can take up to 15 minutes.

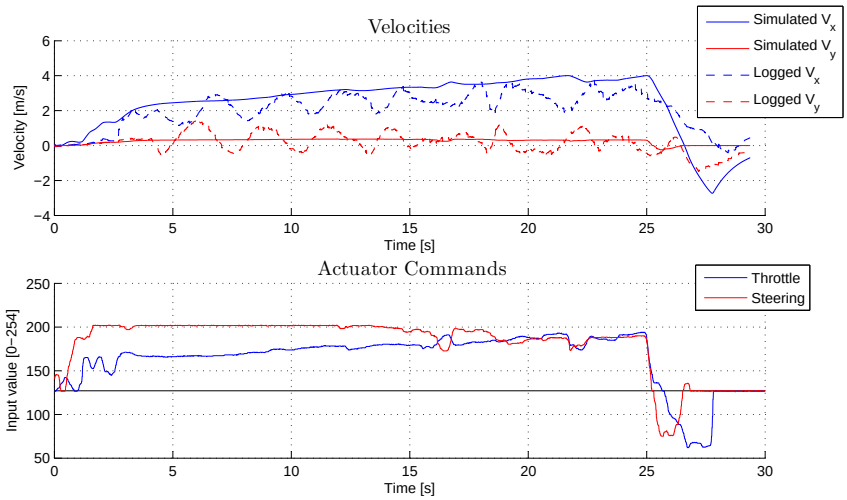

Figure 5.18: Velocity and actuator commands during acceleration and steering. The simulated longitudinal velocity is in agreement with logged data.

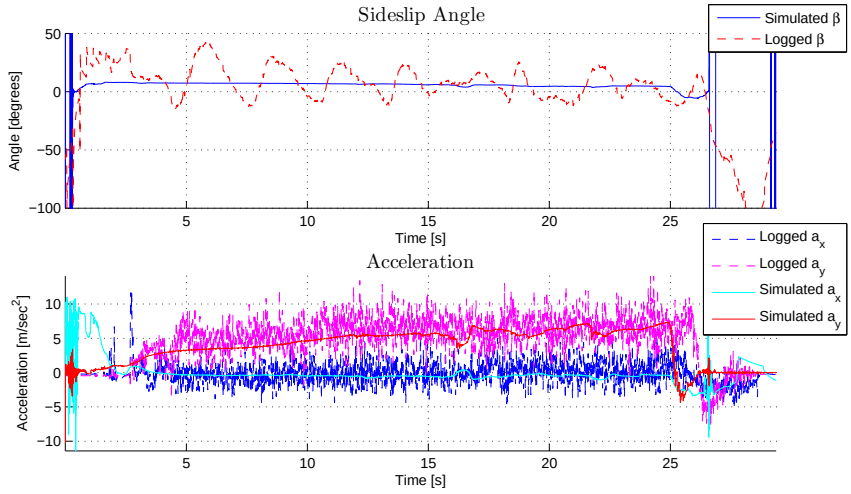

Figure 5.19: Sideslip angle and acceleration during acceleration and steering. The oscillating sideslip angle is not well suited for control design.

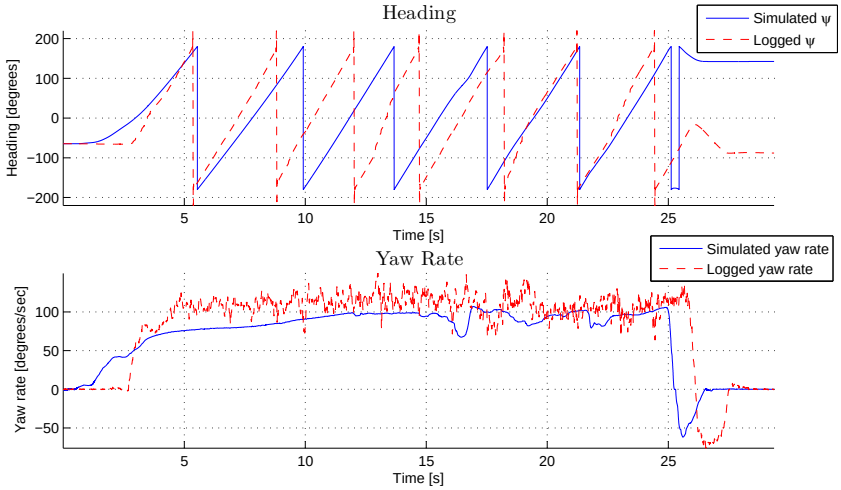

Figure 5.20: Heading and yaw rate during acceleration and steering. There is a good correspondence between logged and simulated data.

- The area has good GPS reception and that the GPS antenna lever arm is configured.
- The filtered data are extracted and that the filter is set up to use the appropriate scenario.

In addition, all data available from the sensors should be logged, such as magnetometer and barometer data, as this can provide a good reference data for later use.

Throughout the tests there are evidence that the logging routine is not synchronized, which should be fixed in later revisions of the software.

Considering the LocalBug simulator, the results are promising as the simulator is in good agreement with the trend in the logged velocities. The comparison of simulated and logged yaw rate shows that the yawing motion is accurately described in the simulator, which is important for testing of a drifting controller. However, log data from a drifting maneuver would have been informative regarding the fidelity of the simulator in such a scenario.

It was discovered that the simulation results are not accurate during hard breaking, as this may destabilize the vehicle. Since no verification of the motor model itself has been performed, determination of the correct motor coefficients could improve the simulation model. In addition, it should be checked that the transmitter and servo is configured such that the full range of actuator values are used to describe steering angle deflection.

Originally, it was unknown whether the friction model and parameters developed for automobiles would scale to a model car, which does not necessarily have the same size, weight and wheel proportions. However, the performed comparison of logged and simulated test cases suggests that the simulation model largely is in agreement with the test data.

Chapter 6

Nonlinear Controller Design

In rally and dirt track racing, drifting has proven its ability to negotiate corners at high speeds on low friction surfaces (Abdulrahim 2006). Optimization of a sequence of control inputs have shown that the solutions to minimum time cornering on low friction surfaces (Velenis et al. 2007) are indeed drift-like maneuvers. These are situations in which conventional driving would normally cause loss of traction and understeer. Enabling a control system to reproduce the actions of a professional driver is an intriguing challenge which can help to improve automotive safety systems beyond the capabilities of such systems today. However, a skilled driver makes a series of rapid maneuvers that is not straight forwardly implemented in a control loop.

Recent research have discovered that when the handling limits of the vehicle in the linear region has been exceeded, the vehicle is still controllable (Hindiyeh and Gerdes 2010). In Voser et al. (2010), unstable equilibria corresponding to drifting maneuvers are identified as saddle points. Since no trajectories converge to these points, continuous control effort is required to keep the vehicle in such a drifting condition.

Because of the nonlinear vehicle dynamics, linear theory is not able to describe the effects resulting from high sideslip angles. One way to approach the problem is by gain scheduling. This approach includes the design of many linear controllers linearized at different operating points throughout the state space. As the vehicle sideslip angle increases, the vehicle response to actuator commands changes, and a scheduling of the controllers ensures that the system is stabilized. This method was first used on fighter aircrafts in the 1950s (Iaon-

nou and Sun 1996). However, it introduces other problems, such as weighting and interpolation of the different controllers to avoid sudden steps in actuator inputs.

Another alternative is to use feedback linearization to cancel the nonlinearities of the system, which is the approach taken in this thesis. Because of the severe nonlinearities inherent in the vehicle dynamics at high sideslip angles, a model which describes the effect of control inputs according to the current state of the system is required. The nonlinear vehicle model presented in chapter 4.2 will be used as the basis for the control design.

6.1 The Control Objective

In current automotive control systems, which aim at stabilizing the vehicle in its linear region, it is important to control the vehicle heading so it corresponds to the drivers intention. This leads to predictable vehicle behavior, but does not take full advantage of the available traction. The motivation behind the feedback linearization controller is to increase the cornering ability of the vehicle by the use of drifting techniques.

The main idea behind the controller design is to use the throttle and steering input to adjust the sideslip angle such that the desired yaw rate is achieved. Normally, the throttle controls velocity and steering regulates vehicle heading. As the sideslip angle increases, the nonlinear behavior of the system becomes apparent. An increase in throttle, and the subsequent increase in longitudinal force, consumes the available traction at the rear wheels. When the wheel force saturates, very little lateral force is produced. The result is skidding of the rear end of the vehicle, which will cause a large yaw rate and sideslip angle. Hence, throttle is very well suited to control the sideslip motion in this state. High sideslip angles therefore introduce a different dynamic response of the actuator inputs which have to be reflected in the control design.

Figure 6.1 illustrates three different equilibrium points for the vehicle. The vehicle approaches the corner in a straight line, as illustrated in situation *a*. The forces acting on the vehicle in this configuration is the aerodynamic drag force and the rolling resistance. Since the vehicle is traveling at constant speed, a small longitudinal force, shown with red arrows at the wheels, is required to counteract the resistive forces.

In a steady state turn, which corresponds to situation *b*, the vehicle follows a circular trajectory. This motion requires a force acting in the center of mass

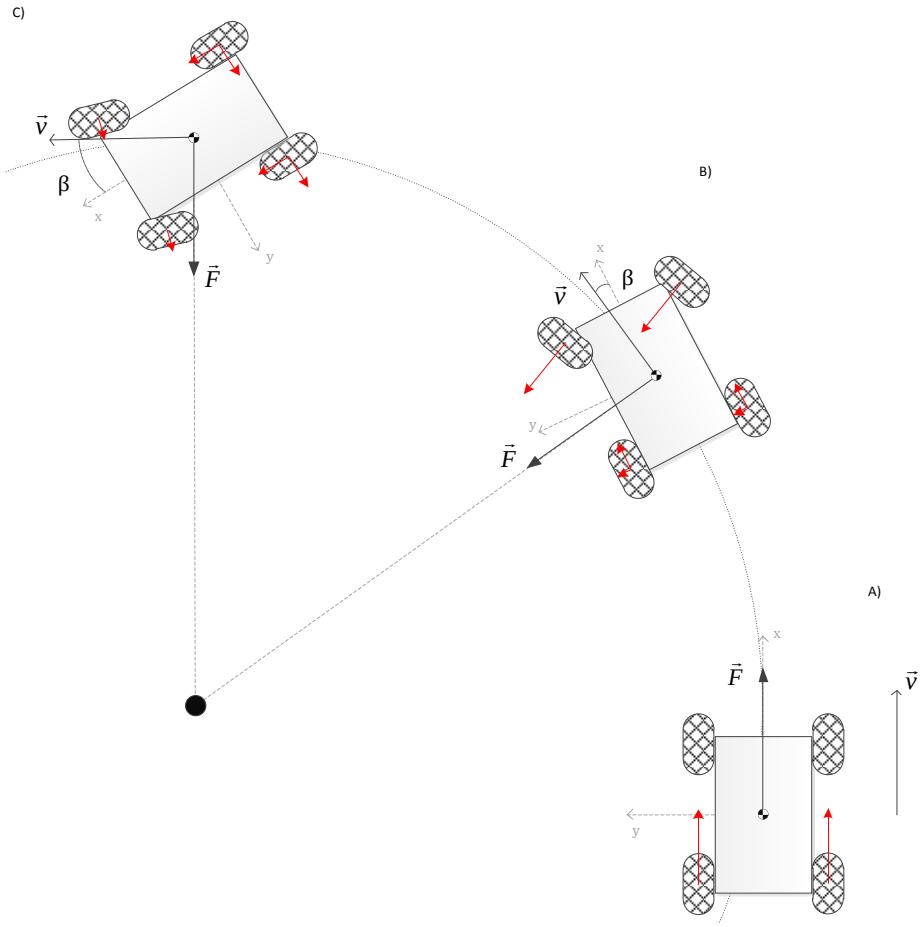

Figure 6.1: Illustration of the vehicle at three equilibrium points.

of the vehicle to be directed towards the center of the turn. The lateral force that accomplishes this is produced by the front wheels, which causes both a lateral force and a yawing moment around the z-axis of the vehicle. The vehicle velocity at the center of gravity is tangential to the circular path. Due to the fact that the vehicle is not a mass point, but rather an object of some size, and also rotating with respect to an inertial frame, the heading of the vehicle is not coincidental with the direction of speed at the center of gravity. This is the sideslip angle β, which is usually insignificant. When the vehicle turns, the rear wheels experiences small slip angles and contribute to stabilize the yawing moment.

It is in this stage it is possible to saturate the rear wheel longitudinal force to destabilize the vehicle. Situation c refers to a drifting equilibrium where coupling of the longitudinal and lateral forces keeps the vehicle in a turn with a large sideslip angle. This cornering technique is also called trail-breaking. A professional driver would break to shift the weight of the vehicle to the front axle. As the driver steers the vehicle into the turn, the rear wheels looses grip. Counter steer is applied as necessary to balance the front and rear wheel forces to control the skid.

Inspired by rally driving, the control objective is to reach and stabilize the vehicle at this equilibrium point. Contrary to the dynamic surface controller in Hindiyeh and Gerdes (2010), which was designed to stabilize the vehicle at a predetermined equilibrium point and track a certain sideslip angle, this controller will search for the equilibrium point corresponding to the given initial conditions and the desired yaw rate. With this in mind, it is a requirement that there exists an equilibrium point the system is able to reach, meaning that the initial conditions are feasible.

Ackerman (1997) claims that the path following task of the driver is primarily concerned with producing the appropriate accelerations a_x and a_y to keep the vehicle on the road. By using the control design model described in section 6.2, it is seen that the forces required to produce these accelerations can obtained by controlling the sideslip angle.

The vehicle dynamics in the horizontal plane have three degrees of freedom. Only two actuators are available for control of LocalBug; the rear wheel throttle and the front wheel steering angle. Hence, the control problem is underactuated. To overcome this challenge, the control objective is defined only by the motion in the longitudinal and lateral direction, meaning that the vehicle will not run off the road. The control problem is now fully actuated in the working space, while the heading is left uncontrolled and appears as a dynamic constraint.

Because of the limited processing power available on LocalBug, it has been a design goal to keep the computational cost down.

6.2 The Control Design Model

It is important that the control design model captures the essential dynamics of the system. The nonlinear two-track vehicle model presented in chapter 4.2 describes well the high level vehicle dynamics with coupling of longitudinal and lateral forces (Kiencke and Nielsen 2005). Because of this complex relationship it is difficult to find a transform that linearizes the input-output map. The model is therefore simplified by ignoring the differences between the left and right track, which is justified from the assumption of a uniform friction on the road surface and differential lock of the rear wheels. The rear wheel drive configuration of LocalBug makes it clear that the free rolling front wheels cannot produce longitudinal forces. The following equations are substituted into the vehicle model:

$$F_{L,FL} = F_{L,FR} = 0 \tag{6.1}$$

$$F_{L,rear} = \frac{1}{2}\left(F_{L,RL} + F_{L,RR}\right) \tag{6.2}$$

$$c_F = \frac{1}{2}(c_{FL} + c_{FR}) \tag{6.3}$$

$$c_R = \frac{1}{2}(c_{RL} + c_{RR}) \tag{6.4}$$

Instead of substituting the expressions for the wheel forces as in Kiencke and Nielsen (2005), a new vector of generalized control inputs consisting of the combined forces and moments resulting from the wheel-ground interaction is defined:

$$\boldsymbol{\tau} = \begin{bmatrix} F_x \\ F_y \\ M_z \end{bmatrix} \tag{6.5}$$

Using (4.9) and (4.10), the control design model is expressed as

$$\begin{bmatrix} \dot{v}_{cg} \\ \dot{\beta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} 0 \\ -\dot{\psi} \\ 0 \end{bmatrix} + \begin{bmatrix} \frac{\cos\beta}{m} & \frac{\sin\beta}{m} & 0 \\ \frac{-\sin\beta}{mv_{cg}} & \frac{\cos\beta}{mv_{cg}} & 0 \\ 0 & 0 & \frac{1}{I_z} \end{bmatrix} \begin{bmatrix} F_x \\ F_y \\ M_z \end{bmatrix} \tag{6.6}$$

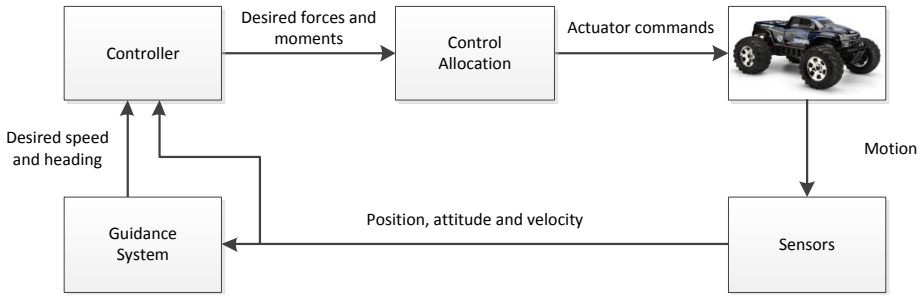

Figure 6.2: Illustration of the control loop signal flow.

which conveniently is on the form

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}) + g(\boldsymbol{x}) \cdot \boldsymbol{u} \tag{6.7}$$

Due to the small frontal cross sectional area of LocalBug, the resistive forces are dominated by the rolling resistance almost in the entire operating envelope of LocalBug. A rolling resistance term could be used to eliminate steady state errors for speed control, but high angles of sideslip makes it difficult to control the speed independently. To avoid interference with the sideslip dynamics, the term has been ignored.

6.3 Feedback Linearizing Controller

The control design model has the following structure

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}) + g_1(\boldsymbol{x})u_1 + g_2(\boldsymbol{x})u_2 + g_3(\boldsymbol{x})u_3 \tag{6.8}$$

where the number of inputs is equal to the number of outputs $m = n = 3$. First, the output is selected to be the first state, $y_1 = v_{cg}$. Differentiating the output gives

$$\dot{y} = \frac{\partial v_{cg}}{\partial \boldsymbol{x}} \frac{\partial \boldsymbol{x}}{\partial t} = L_f h_1(\boldsymbol{x}) + L_{g_1} h_1(\boldsymbol{x}) + L_{g_2} h_1(\boldsymbol{x}) + L_{g_3} h_1(\boldsymbol{x}) \tag{6.9}$$

and the belonging Lie derivatives are computed as

$$L_f h_1(\boldsymbol{x}) = 0 \tag{6.10}$$
$$L_{g_1} h_1(\boldsymbol{x}) = \frac{1}{m}\cos\beta \tag{6.11}$$
$$L_{g_2} h_1(\boldsymbol{x}) = \frac{1}{m}\sin\beta \tag{6.12}$$
$$L_{g_3} h_1(\boldsymbol{x}) = 0 \tag{6.13}$$

Seeing that the Lie derivatives corresponding to the two first inputs are not zero, it is clear that they are connected to the output in some sense. The third input, however, does not influence the output. Because the relative degree is defined when any of the inputs appear in the output equation, the relative degree associated with this output is one. Continuing with the second output, $y_2 = \beta$, the Lie derivatives are

$$L_f h_2(\boldsymbol{x}) = -\dot{\psi} \tag{6.14}$$
$$L_{g_1} h_2(\boldsymbol{x}) = -\frac{1}{m v_{cg}}\sin\beta \tag{6.15}$$
$$L_{g_2} h_2(\boldsymbol{x}) = \frac{1}{m v_{cg}}\cos\beta \tag{6.16}$$
$$L_{g_3} h_2(\boldsymbol{x}) = 0 \tag{6.17}$$

The relative degree of the third input is again not defined, but as long as at least one of the inputs are connected to the output, the relative degree associated with this output channel is also one. Finally, letting the output be $y = \dot{\psi}$, we have

$$L_f h_3(\boldsymbol{x}) = 0 \tag{6.18}$$
$$L_{g_1} h_3(\boldsymbol{x}) = 0 \tag{6.19}$$
$$L_{g_2} h_3(\boldsymbol{x}) = 0 \tag{6.20}$$
$$L_{g_3} h_3(\boldsymbol{x}) = \frac{1}{I_z} \tag{6.21}$$

which also has relative degree one. The total relative degree of the system is $r = r_1 + r_2 + r_3 = 1 + 1 + 1 = 3$. The matrix (3.21) is then defined and becomes

$$\boldsymbol{A}(\boldsymbol{x}) = \begin{bmatrix} \frac{1}{m}\cos\beta & \frac{1}{m}\sin\beta & 0 \\ -\frac{1}{m v_{cg}}\sin\beta & \frac{1}{m v_{cg}}\cos\beta & 0 \\ 0 & 0 & \frac{1}{I_z} \end{bmatrix} \tag{6.22}$$

From chapter 3 it is shown that the feedback linearizing controller

$$\boldsymbol{u} = \boldsymbol{A}(\boldsymbol{x})^{-1} \left(\boldsymbol{v} + \begin{bmatrix} 0 \\ \dot{\psi} \\ 0 \end{bmatrix} \right) \tag{6.23}$$

reduces the system to the linear equivalent

$$\dot{\boldsymbol{x}} = \boldsymbol{v} \tag{6.24}$$

Since the total relative degree is three, there are no internal dynamics. It is important to notice that the control law is not defined for $v_{cg} = 0$, and that a different controller or a manual input is required to initially set off the vehicle. It is now desirable to design a linear controller that makes the yaw rate converge to the desired yaw rate.

The linear control law is chosen as a simple P-controller:

$$\boldsymbol{v} = \begin{bmatrix} -k_v \left(v_{cg} - v_{cg,ref} \right) \\ -k_r \left(r - r_{ref} \right) \\ 0 \end{bmatrix} \tag{6.25}$$

where the weighting between k_v and k_r decides the importance of speed or yaw rate stabilization, respectively.

6.4 Control Allocation

The desired forces calculated in the controller have to be converted into actuator inputs. The output value from the autopilot is an integer ranging from $0 - 254$. On LocalBug, this value is used to create a pulse width modulated (PWM) signal that is fed to the servo motors. This period of the signal lies between 1 ms and 2 ms, which corresponds to the maximum and minimum steering angle or throttle. In the simulator, the integer value is used as input to the vehicle model to imitate the autopilot interface.

A mapping of real control inputs into actuator forces can be taken as Fossen (2011)

$$\boldsymbol{f} = \boldsymbol{K}\boldsymbol{u} \tag{6.26}$$

where $\boldsymbol{f}$ is the force produced by the actuators, $\boldsymbol{K}$ is the force coefficient matrix and $\boldsymbol{u}$ the real control inputs. As discussed in chapter 4.2 and illustrated in

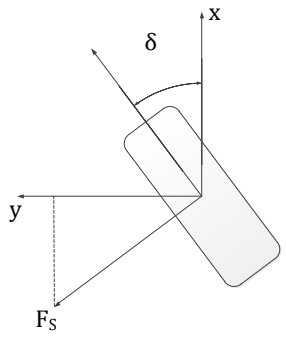

Figure 6.3: Illustration of front wheel lateral force decomposed in the body frame.

figure 6.3, the forces produced by the wheel is directed in the direction of the tire and at right angles to it. At a certain steering angle, the front wheel forces will be rotated an angle δ compared to the desired forces which is calculated in the vehicle body frame. The virtual control inputs is then related to the actuator commands by the following equation:

$$\boldsymbol{\tau} = \boldsymbol{T}(\delta)\boldsymbol{f} = \boldsymbol{T}(\delta)\boldsymbol{K}\boldsymbol{u} \tag{6.27}$$

where $\boldsymbol{T}(\delta)$ is

$$\boldsymbol{T}(\delta) = \begin{bmatrix} 1 & -\sin\delta \\ 0 & \cos\delta \\ 0 & l_{front} \cdot \cos\delta \end{bmatrix} \tag{6.28}$$

An expression for the actuator inputs in terms of dc motor voltage and steering angle is found by solving (6.27) equation for $\boldsymbol{u}$

$$\boldsymbol{u} = \boldsymbol{K}^{-1}\boldsymbol{T}(\delta)^{\dagger}\boldsymbol{\tau} \tag{6.29}$$

where $\boldsymbol{T}(\delta)^{\dagger}$ is the Moore-Penrose pseudo inverse.

$$\boldsymbol{T}(\delta)^{\dagger} = (\boldsymbol{T}^T \cdot \boldsymbol{T})^{-1} \cdot \boldsymbol{T}^T \tag{6.30}$$

Steering Servo The steering servo converts the PWM signal into a voltage corresponding to desired position. The output position of the servo is measured

by a potentiometer, and the two voltages are compared. If the positions do not coincide, the servo will provide the necessary corrections.

In the nonlinear vehicle model described in chapter 4.2, a linear relationship between the wheel slip angle and the cornering force is assumed. However, this is only valid for small angles of wheel slip. In the control scenario discussed in this thesis, the front wheels will counter steer into the turn and saturation of the friction forces will not occur. Thus, the front wheel slip is assumed to lie in the linear region and this validates the use of a linear relationship between slip angle and lateral force. The relationship between lateral force and slip angle is given in section 4.2 as

$$F_S = C_\alpha \alpha = C_\alpha \left(\delta - \beta - \frac{l_{front}\dot{\psi}}{v_{cg}} \right) \tag{6.31}$$

After the desired wheel slip angle is found, the steering angle is resolved from the mapping

$$\delta = \alpha + \beta + \frac{l_{front}\dot{\psi}}{v_{cg}} \tag{6.32}$$

The maximum steering angle on LocalBug is 30° (Wigestrand et al. 2010b), and the output is converted to an integer between 0-254 with the value 127 corresponding to neutral steering.

The cornering coefficient C_α can be found as the gradient of the friction slip curve evaluated at zero slip (Wong 2001). Figure 6.4 illustrates the friction-slip curves for different road surfaces implemented in the LocalBug Simulator. It can be seen that for asphalt and concrete, $C_\alpha \approx 25$, for snow and cobblestones, $C_\alpha \approx 12$ and for ice, $C_\alpha \approx 5$.

Throttle Servo The brushless dc motor fitted in LocalBug is controlled by an Electronic Speed Controller (ESC). The ESC takes the servo signal and converts it into a three phase signal which is used to toggle electromagnets that drives the motor. The current induced by the magnets in the motor is measured to decide the timing of the switching of the magnets.

The force from the tires acting on the road surface, provided the tire forces are not saturated, is given by the motor torque and the radius of the wheel

$$F_L = r_{wheel} \cdot T_{motor} \tag{6.33}$$

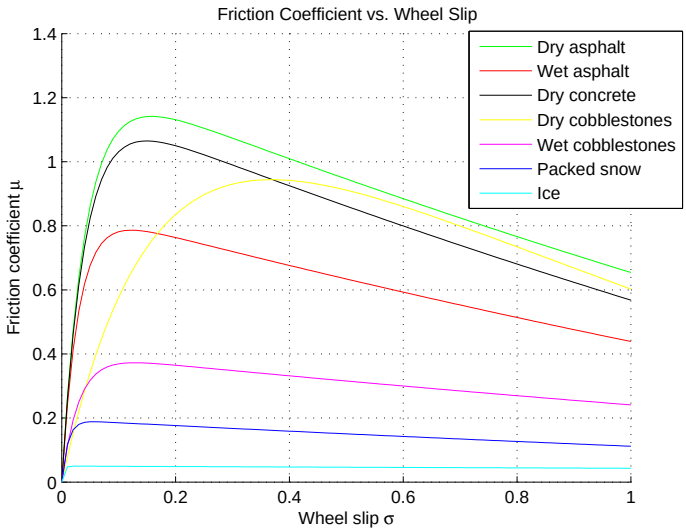

Figure 6.4: Friction coefficients vs. wheel slip for different road surfaces.

Equations (5.2) and (5.1) describes the dc motor dynamics. The motor torque is proportional to the motor current, which is related to the input voltage. However, when the motor starts turning, the back electromechanical force (EMF) reduces the voltage across the terminals and thus the current flowing through the armature is reduced as the engine accelerates. In addition, a part of the engine torque is consumed accelerating the wheel itself. Because of this, the force coefficient is assumed to be varying with the motor speed and a static force coefficient would only be accurate at the start when the back EMF is zero.

Another way to approach the problem of determining the force coefficient is to evaluate the amount of force that can be transferred to the ground and assuming this occurs when maximum throttle is applied. The maximum force is proportional to the normal load on the tire and the friction coefficient (Kiencke and Nielsen 2005). The latter can either be adapted on-line or set to a fixed value. While the normal load can be calculated from the vehicle accelerations, it is taken as a constant to avoid a time-varying friction coefficient matrix. A linear relationship between the produced force and the throttle input can thus

be found as

$$F_L = \frac{\mu F_z}{V_{max}} V \tag{6.34}$$

where μ is the friction coefficient, F_z the normal force on the driving wheels, and V the motor voltage. The motor voltage that can be applied to LocalBug depends on the number of battery cells connected to the motor. LocalBug can fit two 3-cell Li-Po batteries, each cell with a nominal voltage of 4.2 volts (HPI-Racing n.d.). The maximum voltage input is therefore approximately 25 volts.

The resulting force coefficient matrix is given as

$$\boldsymbol{K} = \begin{bmatrix} \frac{\mu F_z}{V_{max}} & 0 \\ 0 & C_\alpha \end{bmatrix} \tag{6.35}$$

where the front wheels produces lateral forces and the rear wheels are assumed to only provide longitudinal forces.

6.5 Controllability During High Angle of Sideslip

To check whether the vehicle is controllable at high angles of sideslip, the vehicle is linearized around a drifting condition. The linearized vehicle model presented in section 4.3 is used to check controllability. Because LocalBug does not have individual torque control on each wheel, and thus only has two inputs, the input matrix B is modified to reflect this. The same considerations as discussed in section 6.2 is taken. The matrices A and B to the linear system are then given as

$$\boldsymbol{A} = \begin{bmatrix} \frac{\partial \dot{v}_{cg}}{\partial v_{cg}} & \frac{\partial \dot{v}_{cg}}{\partial \beta} & \frac{\partial \dot{v}_{cg}}{\partial \dot{\psi}} \\ \frac{\partial \dot{\beta}}{\partial v_{cg}} & \frac{\partial \dot{\beta}}{\partial \beta} & \frac{\partial \dot{\beta}}{\partial \dot{\psi}} \\ \frac{\partial \ddot{\psi}}{\partial v_{cg}} & \frac{\partial \ddot{\psi}}{\partial \beta} & \frac{\partial \ddot{\psi}}{\partial \dot{\psi}} \end{bmatrix}, \qquad \boldsymbol{B} = \begin{bmatrix} \frac{\partial \dot{v}_{cg}}{\partial F_L} & \frac{\partial \dot{v}_{cg}}{\partial \delta} \\ \frac{\partial \dot{\beta}}{\partial F_L} & \frac{\partial \dot{\beta}}{\partial \delta} \\ \frac{\partial \ddot{\psi}}{\partial F_L} & \frac{\partial \ddot{\psi}}{\partial \delta} \end{bmatrix} \tag{6.36}$$

The system is linearized around an equilibrium point determined by a simulation where drifting was achieved. The road surface condition was set to wet cobblestones, and LocalBug were configured to rear wheel drive with locked differential. After an initial settling period, the steady state values for the vehicle states were

$$\begin{bmatrix} v_{cg} \\ \beta \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1.93\,{}^{m}/_{s} \\ -0.74\,rad \\ 1.175\,{}^{rad}/_{s} \end{bmatrix} \tag{6.37}$$

with the control inputs

$$\begin{bmatrix} F_L \\ \delta \end{bmatrix} = \begin{bmatrix} 11.71\,N \\ -0.514\,rad \end{bmatrix} \tag{6.38}$$

Using a cornering stiffness value of $C_\alpha = 12$, as determined for wet cobblestones in section 6.4, the controllability matrix $\boldsymbol{C}(\boldsymbol{A},\boldsymbol{B}) = \begin{bmatrix} \boldsymbol{B} & \boldsymbol{AB} & \boldsymbol{A}^2\boldsymbol{B} \end{bmatrix}$ is computed to

$$\boldsymbol{C}(\boldsymbol{A},\boldsymbol{B}) = \begin{bmatrix} 0.1055 & -0.4554 & 0.2005 & 3.3096 & -0.5645 & -4.8829 \\ 0.0499 & 0.8572 & -0.1438 & -1.3156 & -0.0217 & 1.1556 \\ 0 & 0.1992 & 0.1270 & -0.9804 & -0.0053 & 6.0361 \end{bmatrix}$$

It is seen that the controllability matrix is of full rank. Because the linearized system equations only contain information about the point which the system is linearized about, nothing can be said about the controllability in general. However, the system is controllable for the selected operating point, and this indicates that the vehicle is controllable at high angles of sideslip.

6.6 SIMULINK Implementation

The feedback linearizing controller and control allocation algorithms developed in section 6.3 and 6.4 is implemented in SIMULINK as shown in figure 6.5.

Data required from the IMU are gyro data and estimated velocities and orientation. The velocities from the IMU are given in a north-east-up coordinate frame, and thus have to be converted to the body frame for calculation of the sideslip angle. The last two inputs are the desired speed and yaw rate.

The controller and control allocation is implemented using embedded functions in Matlab. The controller code is given in appendix C.

A saturation block limits the output of the control allocation block to the specified limits of LocalBug. For throttle, the allowed input is 0-25 volts, while the steering angle saturates at ± 30 degrees. The last step is a conversion from voltage to an input command from $0 - 254$, with 127 being neutral position for both actuators.

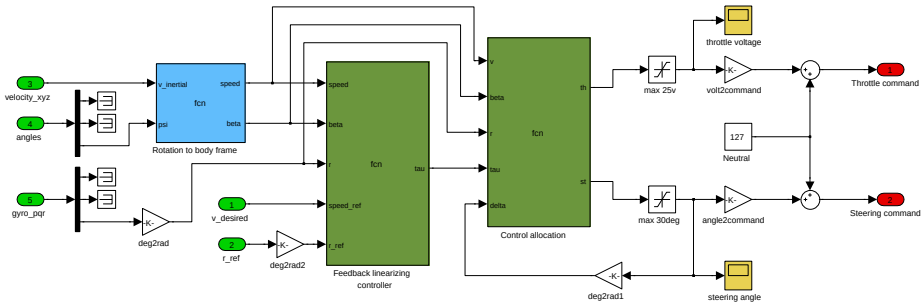

Figure 6.5: Screenshot of the SIMULINK controller block.

Chapter 7

Simulation Results

Several test cases that requires the use of drifting techniques to achieve the desired yaw rate has been designed. The following test cases will test the performance of the controller:

- The first test case investigates the performance of the controller for two yaw rate setpoints on wet cobblestones, which is a medium friction surface.
- The second test case investigates the performance of the controller for two yaw rate setpoints on snow, which is a low friction surface.
- The third test case examines the importance of uncertainties in the mass parameter used in the controller.
- The final test case compares the cornering abilities of the feedback linearization controller designed in chapter 6 to a conventional PID controller.

To obtain a sufficiently realistic simulation, the noise data described in section 5.4 were added to the feedback loop to ensure the controller is robust enough to be implemented on LocalBug. However, the vehicle states are plotted without noise to simplify discussion of the results.

From the derivation of the force coefficient matrix $\boldsymbol{K}$ in section 6.4 it is seen that the values for $\boldsymbol{K}$ can be calculated as

$$\boldsymbol{K} = \begin{bmatrix} \frac{0.4 \cdot m \cdot g}{V_{max}} & 0 \\ 0 & C_\alpha \end{bmatrix} = \begin{bmatrix} 1.1 & 0 \\ 0 & 12 \end{bmatrix} \tag{7.1}$$

for wet cobblestones, where the $\mu = 0.4$ is the peak friction coefficient seen in figure 6.4 and $m = 7$ is the weight of LocalBug (Jakobsen 2010). Similarly for snow, the coefficients are

$$\boldsymbol{K} = \begin{bmatrix} \frac{0.2 \cdot m \cdot g}{V_{max}} & 0 \\ 0 & C_\alpha \end{bmatrix} = \begin{bmatrix} 0.55 & 0 \\ 0 & 5 \end{bmatrix} \tag{7.2}$$

During testing of the controller, it was discovered that tuning of the force coefficient matrix was necessary to achieve a satisfactory result. The following matrix is used throughout the simulations

$$\boldsymbol{K} = \begin{bmatrix} 3 & 0 \\ 0 & 6 \end{bmatrix} \tag{7.3}$$

The values for the feedback linearization controller gains were set to

$$k_v = 4 \tag{7.4}$$
$$k_r = 100 \tag{7.5}$$

and is used for all simulations.

7.1 Case 1: Cornering on Wet Cobblestones

For this test, the initial and commanded velocity is set to $4\,{}^{m}/_{s}$. The controller is given two different yaw rate setpoints of $-40\,{}^{deg}/_{s}$ and $-60\,{}^{deg}/_{s}$. Halfway in the simulation, the sign of the yaw rate is changed so the transient behavior between two opposite turns can be observed.

With a reference yaw rate of $40\,{}^{deg}/_{s}$ the controller is able to complete the turn without engaging in a drifting maneuver. Figure 7.3 indicates that a sideslip of 2 degrees results from the maneuver. A steering angle of only 10 degrees is sufficient to provide the lateral force required to obtain the desired yaw rate. The drop in velocity with approximately 20 percent in velocity is significant, but results from the controller being tuned to prioritize the yaw rate. During

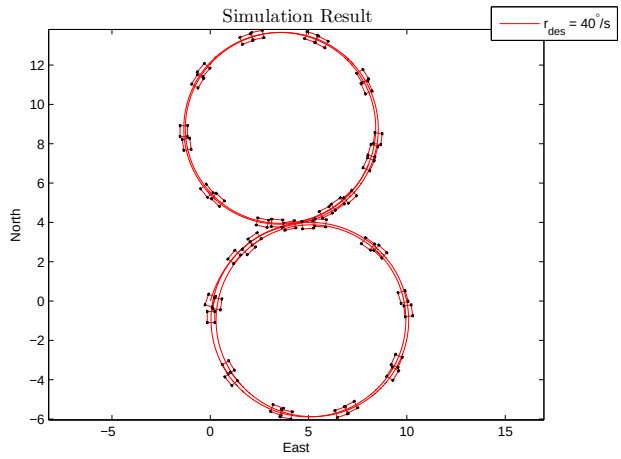

Figure 7.1: Simulation result on wet cobblestones with $\dot{\psi}_{des} = 40\ ^{deg}/_{s}$. It is seen that no significant sideslip occurs in this case. The front wheels are shown in red.

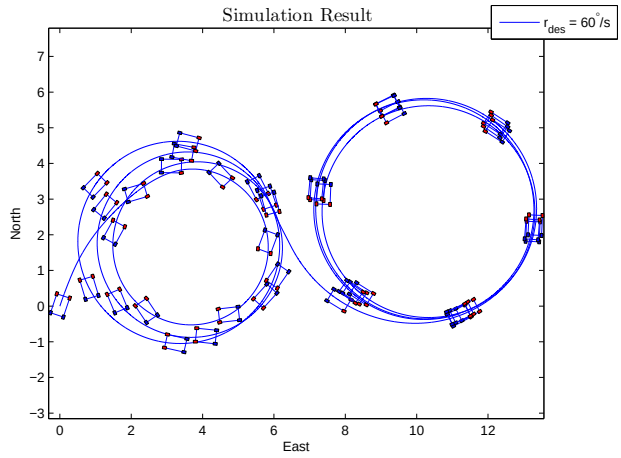

Figure 7.2: Simulation result on wet cobblestones with $\dot{\psi}_{des} = 60\ ^{deg}/_{s}$. The controller initiates a drift to reach the yaw rate reference. The front wheels are shown in red.

the transition from the left hand turn to the right hand turn, the controller is aggressive on both steering and throttle input to quickly stabilize LocalBug at the new setpoint.

At 60 $^{deg}/_{s}$, figure 7.4 shows that the controller initially uses a lot of control effort to stabilize the vehicle. Once the sideslip angle starts to build up, the controller responds with counter steering and by releasing the throttle to avoid a spin. Up to time $t = 20$, the yaw rate is stabilized at 60 degrees, but the vehicle is not resting at the equilibrium point. During this time, the throttle setting remains fairly constant. The steering angle is gradually relaxed as the sideslip angle decreases. The equilibrium point is quickly found in the second turn, since the initial velocity was lower.

Both yaw rates are achieved very fast, in about 0.5 seconds.

7.2 Case 2: Cornering on Snow

The second test case takes place on snow, which is a low friction surface. The initial and commanded velocity is $2.5^{m}/_{s}$. Two test runs are performed, the first with an initial desired yaw rate of -40 $^{deg}/_{s}$ and the second test run with -60 $^{deg}/_{s}$. At time $t = 20$ the setpoints are changed to 40 $^{deg}/_{s}$ and 60 $^{deg}/_{s}$, respectively.

From figure 7.7 it is seen that with the lowest yaw rate reference, the controller is able to maintain a higher speed throughout the turn. The desired yaw rate can in this case only be achieved by establishing a sideslip angle. The sideslip angle is increasing up to 35 degrees at which the dynamics are stabilized.

In the second test run with a desired yaw rate of 60 $^{rad}/_{s}$, there is not enough traction to negotiate the turn at 2.5 $^{m}/_{s}$. Figure 7.8 shows that the controller tries to control the skid by counter steering and setting the throttle to idle, but the steering angle saturates and a spin is inevitable. The spin is caused by a too large initial velocity, as the same response is not seen after the velocity has dropped. In the second turn, an equilibrium point is quickly found with a sideslip angle of 46 degrees. Due to the high slip angle and relatively low velocity, the circular path is very tight, as seen in figure 7.6.

Both test runs stabilizes at a yaw rate which is approximately 5 degrees higher than the commanded yaw rate. It is seen from figure 7.8 that the control inputs are not saturated and tuning of the controller could provide a better result.

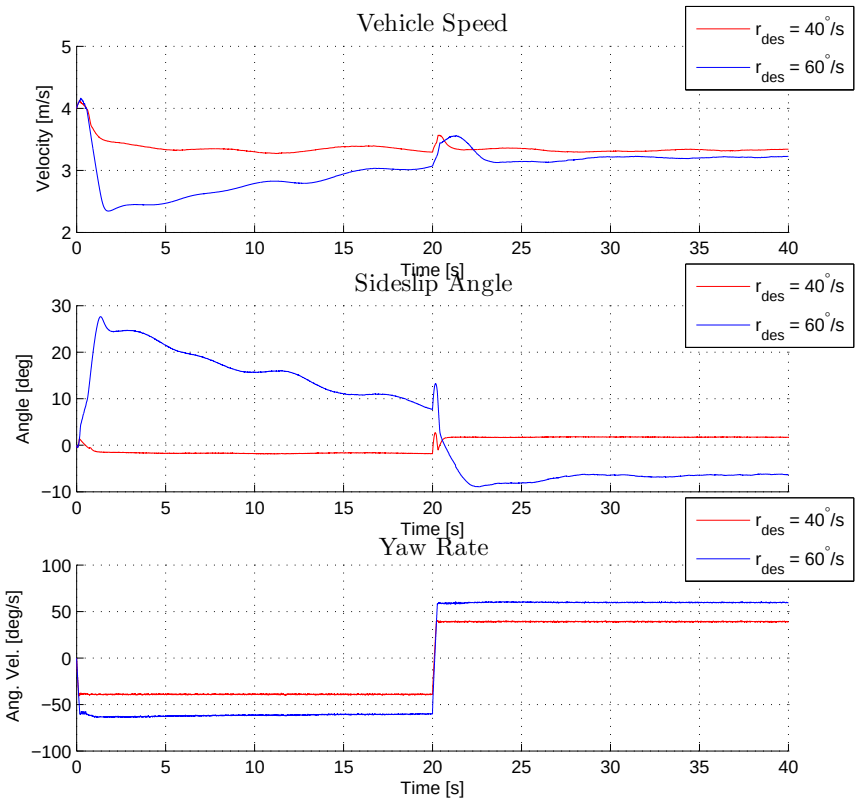

Figure 7.3: Plot of the vehicle states while driving in circles on wet cobblestones. The desired yaw rate is achieved in both cases.

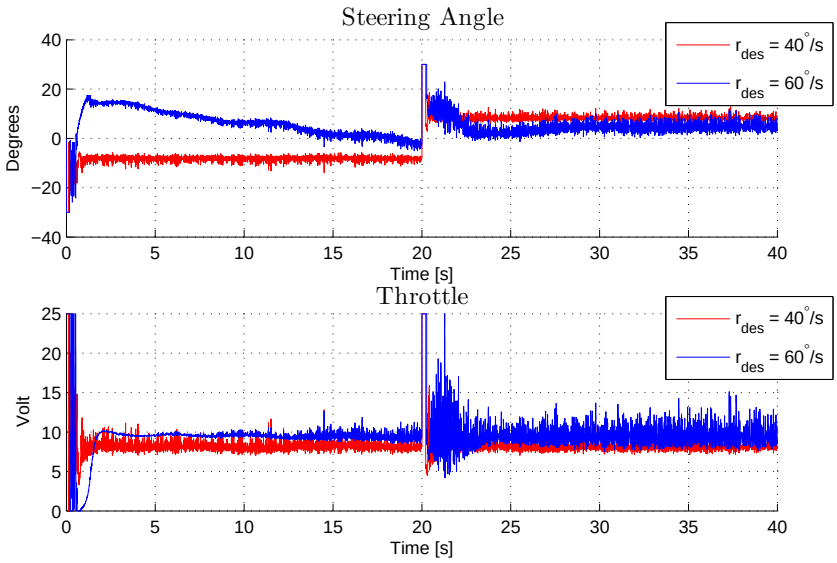

Figure 7.4: Plot of the control inputs while driving in circles on wet cobblestones.

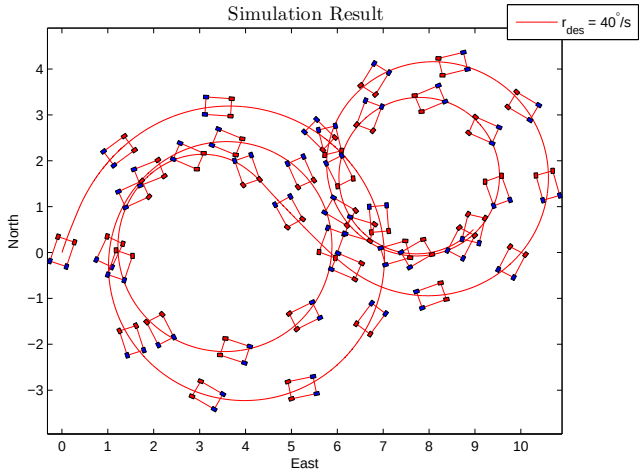

Figure 7.5: Simulation result on snow with $\dot{\psi}_{des} = 40^{deg}/_{s}$. The spiraling motion results from the velocity decreasing. The front wheels are shown in red.

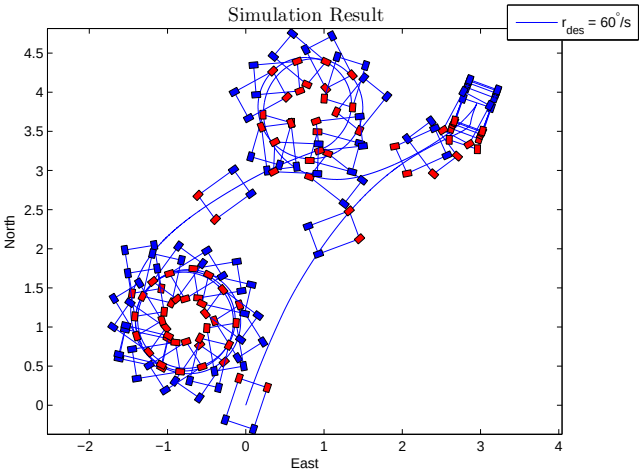

Figure 7.6: Simulation result on snow with $\dot{\psi}_{des} = 60^{deg}/_{s}$. The initial spin due to too low traction is clearly seen in this figure. The front wheels are shown in red.

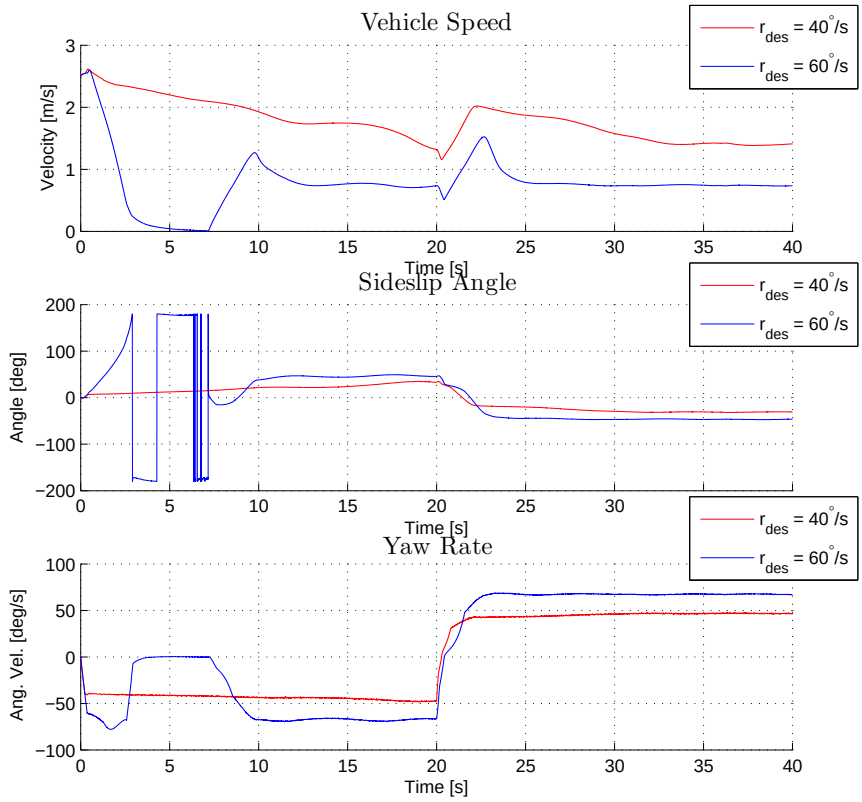

Figure 7.7: Plot of the vehicle states while driving in circles on snow.

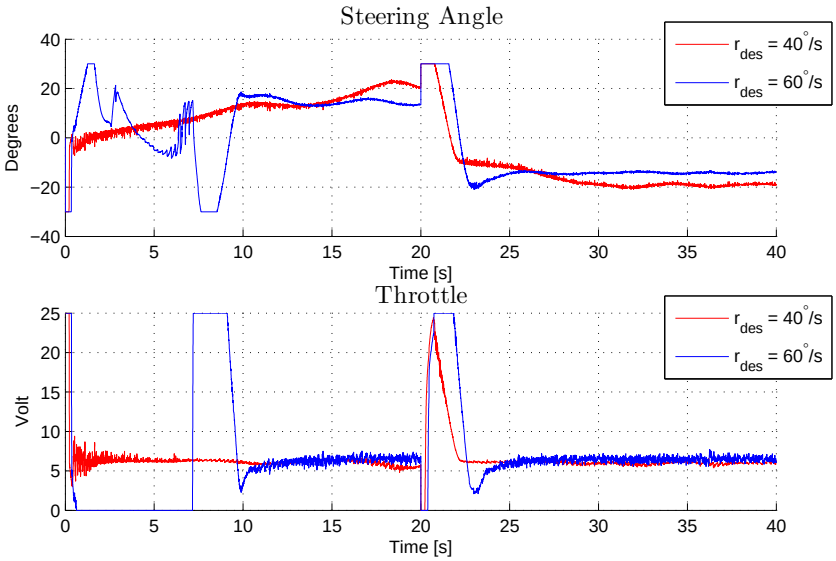

Figure 7.8: Plot of the control inputs while driving in circles on snow. Saturation of the control inputs occur for the desired yaw rate of 60 degrees.

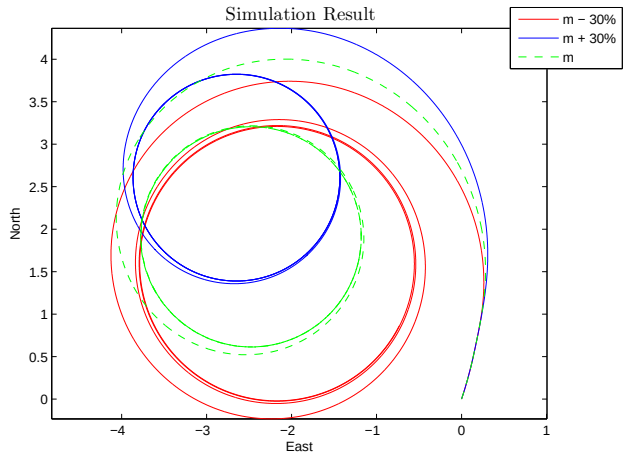

Figure 7.9: Simulation result of different values in the mass parameter. The vehicle outlines have been removed for clarity.

7.3 Case 3: Robustness to Parametric Uncertainties

In this case the influence of the mass parameter in the feedback linearization controller is investigated. The simulation is performed on wet cobblestones with an initial speed of 2.5 $^{m}/_{s}$, a desired speed of 4 $^{m}/_{s}$, and a desired yaw rate of 70 $^{deg}/_{s}$. The simulation is carried out without noise to easier spot the differences. The simulation was performed three times; with a decrease in the mass parameter of 30 percent, with an increase in the mass parameter by 30 percent, and a reference simulation with the exact mass of the vehicle.

From figure 7.9 it is clearly seen that the change in mass parameter influences the path driven by LocalBug. A further investigation reveals that the tracked yaw rate is not very different in the three cases. The main difference in vehicle path is attributed different velocities that results from different sideslip angles. This coincides with intuition, since the mass parameter in the feedback linearization controller acts like a gain on the control forces. An increase of 30 percent in this parameter causes LocalBug to use more control effort, which is also seen in figure 7.11.

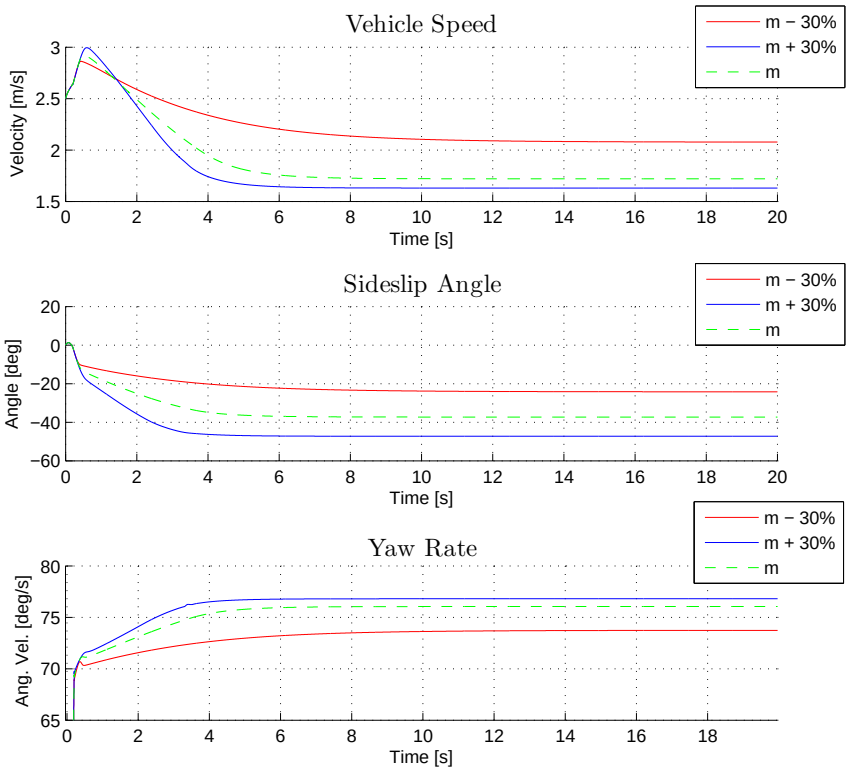

Figure 7.10: Vehicle states for different values of the mass parameter. The yaw rates do not differ as much as the speed and sideslip angles.

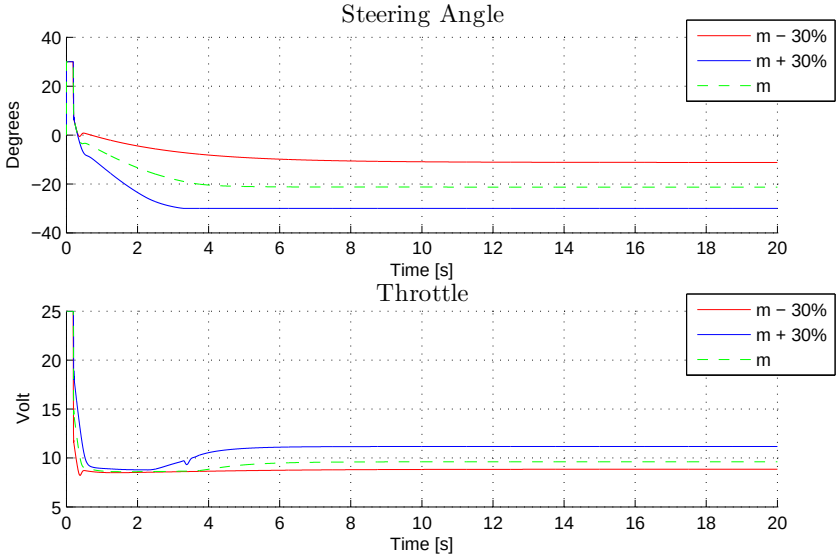

Figure 7.11: Control inputs for different values of the mass parameter.

Table 7.1: Values for the reference PID controllers.

	P	I	D
Throttle controller	10	0	0
Steering controller	35	20	0

7.4 Case 4: Comparison With PID Controller

The final test case highlights the advantage of drifting through a corner versus using a conventional cornering technique. The initial and command speed during the test is 4.5 $^{m}/_{s}$, and the road surface consists of wet cobblestones. At the first waypoint, the yaw rate setpoint for the controllers changes to 60 $^{deg}/_{s}$. After completing a 210 degree hairpin turn, which happens when LocalBug is within two meters of the second waypoint, a yaw rate command is given to aim for the next waypoint located out of the picture, a distance along the green line shown in figure 7.12. The initial speed is too large to complete the turn without running of the road. For comparison, at a yaw rate of 60 $^{deg}/_{s}$, the speed cannot be greater than 3.6 $^{m}/_{s}$ to successfully complete the turn.

The PID throttle controller was tuned to provide the same straight-line velocity as the feedback linearization controller. Integral action was added to the PID steering controller to prevent a reduction in steering angle when the yaw rate error diminishes. In addition, this mimics the average driver, which is prone to increase steering when the vehicle does not respond as much to steering commands as expected. The values for the PID controllers are given in table 7.1.

Looking at the simulation result in figure 7.12, it is evident that by using trail-breaking, LocalBug manages to stay on the road. Since the speed drops to 3 $^{m}/_{s}$, the path is not completely circular and the vehicle completes the turn within the path described by the waypoints. The conventional driving technique overshoots the waypoint by two meters and is not able to keep the desired yaw rate or the vehicle on the road. In other words, the maneuverability of the vehicle has increased by not restricting the vehicle to operate within the linear region characterized by low sideslip angles.

Inspection of the yaw rate in figure 7.14, which is very close the setpoint, reveals that the feedback linearization controller has completed the turn at time $t = 12$ seconds, while the vehicle controlled by the PID controller is through at $t = 14$ seconds. Even though the exit velocity is lower for the feedback linearizing controller, the simulation showed that the vehicle commanded by this controller was the first to reach the final waypoint.

Figure 7.13 shows that the steering angle saturates for the PID controller. Non-linearities in the lateral friction force are also visible in this case, seeing that the maximum yaw rate is achieved at 11 degrees steering angle.

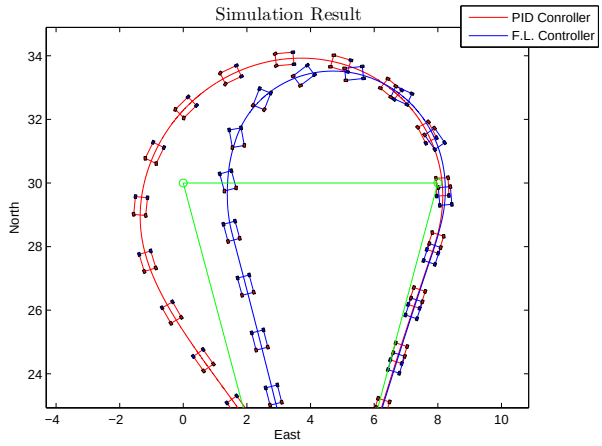

Figure 7.12: Simulation result of the trail-breaking turn. LocalBug is able to complete the turn only by employing drifting techniques. The front wheels are shown in red.

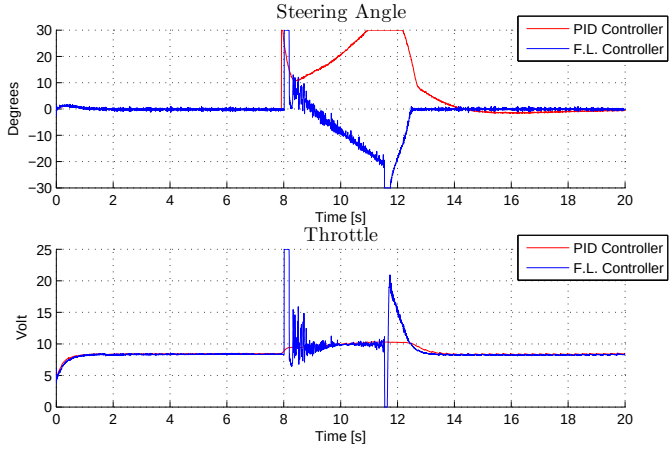

Figure 7.13: Comparison of steering input during trail breaking and conventional cornering. Notice the more aggressive driving behavior during trail breaking.

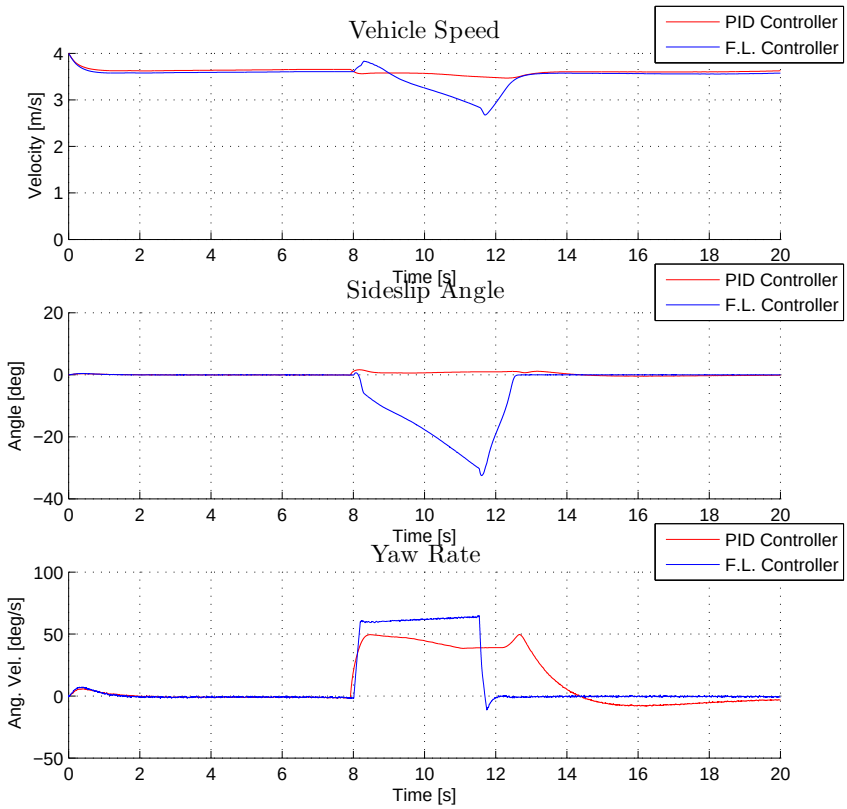

Figure 7.14: Comparison of vehicle states during trail breaking and conventional cornering. It is seen that the trail-breaking turn is completed faster, although the exit velocity is lower.

7.5 Discussion of Results

The simulation results show that the controller is able to obtain and sustain the desired yaw rate on the medium friction road surface. On the low friction surface, the obtained yaw rate is within 5 degrees of the desired yaw rate. It is seen that the amount of sideslip is a result of the initial velocity, the desired yaw rate and the road surface friction. If possible, the desired yaw rate is achieved without sideslip. This corresponds to the controller searching for the equilibrium satisfying the initial conditions and the desired yaw rate.

In the second test case it is observed that LocalBug may become uncontrollable if the available traction is not sufficient to obtain the yaw rate setpoint. Throttle is applied to increase the sideslip angle and thereby the yaw rate, and counter steer is used to control the yawing moment. When the steering angle saturates, there are not enough steering forces to prevent an uncontrolled spin.

Because the speed controller only includes a proportional term, the desired speed is not attained in any of the simulations. This was expected considering that the design goal of the controller did not prioritize speed. When the sideslip angle becomes large, the speed controller is completely dominated by the sideslip controller because of the differences in controller gains. This is desirable, since it is more pertinent to sustain correct yaw rate rather than speed in this setting. The high gain on the yaw rate controller is also required to obtain a fast response and excite the vehicle into a drifting configuration.

Considering the large variations in calculated sideslip angle from logged data set in chapter 5, it seems appropriate not to use this value directly in the control objective. The yaw rate, which is easier to measure, is better suited for the purpose.

The decision to only include P-controllers for speed and yaw rate was taken to avoid phase lag that could destabilize the sideslip dynamics. Although integral action could improve the steady state error seen in the second test case, the extra phase lag was not desirable considering the sensitivity of the dynamics with respect to the sideslip angle. Phase lag or wind-up could quickly cause the sideslip angle to grow beyond salvation. Derivative action would help to reduce the phase lag, but may be susceptible to noise and was therefore discarded. Integral and derivative action can perhaps be included in future revisions of the control system.

It is interesting to notice that the same controller parameters were successfully tested on different road surface conditions. This corresponds well to the findings in Voser et al. (2010) which suggests that drifting possesses some predictable

handling characteristics over varying friction. In the article, ±10 percent variation in friction only causes a 1.2 degrees variation of β at the equilibrium point.

The experience that tuning of the force coefficient matrix was necessary to achieve a sufficient result indicates that the control allocation block should include lateral forces produced by the rear wheels, as well as nonlinear effects in the tire-friction response. Especially the linear approximation between the lateral slip and lateral force is inaccurate even at modest wheel slip angles. The use of accelerometer data to calculate the normal load on the rear wheels could provide a better value for the longitudinal force coefficient, as long as the maximum friction is known. Because there are no wheel speed sensors on LocalBug, it is difficult to estimate the surface friction and the wheel slip, and therefore also the forces acting on each tire.

In all tests, the added noise in the feedback loop causes chattering of various strength on the control inputs. This is especially noticeable in figure 7.4. While such chattering can originate from high gains, this can also be part of a solver issue. The simulator uses a fixed time step solver with 1 ms step time, which may not be sufficient to model fast dynamics. In an implementation aspect it should be verified that severe chattering which may cause damage to LocalBug does not occur.

The choice to use feedback linearization as a method of obtaining a desired yaw rate through establishing a sideslip has proven to be effective. When the desired yaw rate is set too high to be feasible for normal driving techniques, a tail-breaking configuration with a high sideslip angle and counter steering for stability is established. In the last test case, it is evident that the feedback linearization controller is able to complete the turn while staying on the road, while the PID controller is not.

Although the simulation of test case four showed that the feedback linearization controller succeeded in completing the turn, there is room for improvement. A skilled driver would execute a series of maneuvers to properly position the car in front of the turn and to establish the sideslip. None of these maneuvers are considered by the designed controller. Other techniques, for instance open loop commands as suggested in Henry and Perrault (2010) and Voser et al. (2010), is therefore required for more gentle excitation of the sideslip dynamics.

Chapter 8

Conclusions

In this thesis, feedback linearization has been used to design a state feedback controller that, by the use of drifting techniques, stabilizes LocalBug at the desired yaw rate. At high angles of sideslip, the forces acting between the tire and the ground are highly nonlinear and the dynamic response of steering and throttle actuators changes.

The controller has proven to be capable of achieving the desired yaw rate with an accuracy of within 5 degrees, and also shown to perform better than conventional PID control. This supports the idea of a new generation active safety systems which does not restrict the vehicle to operate within its linear region. Modern cars are equipped with a variety of sensor systems. Position data from global satellite navigation systems combined with map data, radar and optical sensors could be used to design a yaw rate reference that would keep the vehicle on the road or evade obstacles in front of the car.

The LocalBug simulator has been improved in several areas. Physical values for the moment of inertia of LocalBug and the wheels have been obtained by measurements. The addition of a dc motor model and sensor noise data has significantly increased the realism of the simulator. A verification of the LocalBug simulator was performed by comparing simulation data against logged test data. The results show that the simulator is largely able to accurately reproduce the motions of LocalBug. However, the simulation results diverge from logged data during hard breaking.

Feedback linearization as a control design technique has been successful in describing the coupling of forces and actuator authority that arises from large

variations in sideslip angle. The method is recommended for further use in the derivation of controllers for LocalHawk, which is prone to experience similar effects due to coupling of aileron, elevator and rudder action in different flight conditions.

8.1 Further work

The controller developed in chapter 6 should be implemented on LocalBug and tested. As shown in the simulation results, the controller works satisfactorily in presence of the recorded noise. However, feedback linearization does not guarantee robustness of the system. Should stability issues arise during testing, an extension to sliding mode control could be an appropriate line of action.

Some concerns regarding data logging were found. To ensure data integrity during logging, it should be made sure that each measurement is accurately timestamped. In addition, the logging function should arrange for logging of all sensor data, as this can be of importance for future review of the data. Future controllers may also require more sensor data, especially considering the goal of interchanging hardware and software between LocalHawk and LocalBug.

Due to the lack of knowledge of the wheel speeds on LocalBug, accurately determining the forces acting at the wheel-ground interface is difficult. It should be considered if wheel speed encoders could be added to the vehicle. This would enable calculation of wheel slip, which is the basis for calculating wheel forces and estimating the road surface friction.

Bibliography

Ånnestad, D. C. (2010). Autonomous Bicycle. MSc thesis.

Abdulrahim, M. (2006). On the Dynamics of Automobile Drifting, *SAE 2006 World Congress & Exhibition*.

Ackerman, J. (1997). Robust Control Prevents Car Skidding, *Control Systems, IEEE* **17**(3): 23–31.

Burdick, J. (n.d.). The Moore-Penrose Pseudo Inverse. Accessed 19.05.2011. **URL:** *http://robotics.caltech.edu/ jwb/courses/ME115/handouts/pseudo.pdf*

Croft-White, M. (2006). *Measurement and Analysis of Rally Car Dynamics at High Attitude Angles*, PhD thesis, Cranfield University.

Egeland, O. and Gravdahl, J. T. (2002). *Modeling and Simulation for Automatic Control*, Marine Cybernetics.

Fossen, T. I. (2011). *Handbook of Marine Craft Hydrodynamics and Motion Control*, John Wiley & Sons.

Grip, H. F. (2010). *Topics in State and Parameter Estimation for Nonlinear and Uncertain Systems*, PhD thesis, Norwegian University of Science and Technology.

Grip, H. F., Imsland, L., Johansen, T. A., Kalkkuhl, J. C. and Suissa, A. (2009). Vehicle Sideslip Estimation - Design, implementation and experimental validation, *IEEE Control Systems Magazine* **29**(5): 36–52.

Haugan, J. (1992). *Formler og tabeller*, NKI.

Henry, S. and Perrault, A. (2010). Autonomous RC Car Drifting, *Technical report*, Cornell University.

Hindiyeh, R. Y. and Gerdes, J. C. (2010). Design of a Dynamic Surface Controller for Vehicle Sideslip Angle During Autonomous Drifting, *Advances in Automotive Control*.

HPI-Racing (n.d.). Savage flux hp. Accessed 09.06.2011.
URL: *http://www.hpieurope.com/kit-info.php?lang=en&partNo=104242*

Iaonnou, P. A. and Sun, J. (1996). *Robust Adaptive Control*, Prentice Hall.

Isidori, A. (1995). *Nonlinear Control Systems*, third edn, Springer-Verlag London.

Isidori, A., Marconi, L. and Serrani, A. (2003). *Robust Autonomous Guidance*, Springer-Verlag London.

Jakobsen, J. (2010). LocalBug: Vehicle Simulator with Skidding and Slipping, *Technical report*, Norwegian University of Science and Technology.

Kececi, E. F. and Tao, G. (2006). Adaptive Vehicle Skid Control, *Mechatronics* **16**(5): 291–301.

Khalil, H. K. (2002). *Nonlinear Systems*, third edn, Prentice Hall.

Kiencke, U. and Nielsen, L. (2005). *Automotive Control Systems: For Engine, Driveline and Vehicle*, Springer-Verlag Berlin Heidelberg.

Kooijman, J. D. G. (2006). *Experimental Validation of a Model for the Motion of an Uncontrolled Bicycle*, Master's thesis, Delft University of Technology.

Marino, R. and Tomei, P. (1995). *Nonlinear Control Design*, Prentice Hall International.

Mason, M. T. (2011). Nonholonomic Constraint. Accessed 24.05.2011.
URL: *http://www.cs.cmu.edu/afs/cs/academic/class/16741-s07/www/lecture5.pdf*

Newman, P. M. (2003). C4B - Mobile Robotics. Accessed 02.06.2011.
URL: *http://www.scribd.com/doc/50320116/1/Holonomicity*

Olfati-Saber, R. (2002). Exponential e-Tracking and e-Stabilization of Second-Order Nonholonomic SE(2) Vehicles Using Dynamic State Feedback, *Proceedings of the American Control Conference*.

Rajamani, R. (2006). *Vehicle Dynamics and Control*, Springer.

Slotine, J. E. and Li, W. (1991). *Applied Nonlinear Control*, Prentice Hall.

Veierland, P. M. (2010). *LocalHawk PhoenixII*, Master's thesis, Aberystwyth University.

Velenis, E., Frazzoli, E. and Tsiotras, P. (2010). Stedy-State Cornering Equilibria and Stabilization for a Vehicle During Extreme Operating Conditions, *International Journal of Vehicle Autonomous Systems* **8**(2/3/4): 217–241.

Velenis, E., Tsiotras, P. and Lu, J. (2007). Modeling Aggressive Maneuvers on Loose Surfaces: The Cases of Trail-Breaking and Pendulum-Turn, *Proceedings of the 2007 European Control Conference*. Kos, Greece, July 2-5 2007.

Vik, B. (n.d.). Integrated Satellite and Inertial Navigation Systems.

Vold, J. O. (2010). Guidance System of an AUAV - Local Hawk, *Technical report*, Norwegian University of Science and Technology.

Voser, C., Hindiyeh, R. Y. and Gerdes, C. (2010). Analysis and Control of High Sideslip Manoeuvres, *Vehicle System Dynamics* **48**(Supplement 1): 317–336.

Wenstad, P. (2010). *GPS Guided R/C Car*, Master's thesis, Norwegian University of Science and Technology.

White, F. M. (2008). *Fluid Mechanics*, sixth edn, McGraw-Hill.

Wigestrand, B. J., Breivik, H., Dykesteen, P. A., Melby, H., Norendal, P. S., Veierland, P. M., Samuelsen, E., Evju, O., Jenssen, E., Lilleborge, M., Fuglstad, G. A. and Bhandari, A. (2010a). LocalBug Brukermanual, *Technical report*, Kongsberg Defence Systems.

Wigestrand, B. J., Breivik, H., Dykesteen, P. A., Melby, H., Norendal, P. S., Veierland, P. M., Samuelsen, E., Evju, O., Jenssen, E., Lilleborge, M., Fuglstad, G. A. and Bhandari, A. (2010b). LocalHawk/LocalBug Teknisk Rapport Sommer 2010, *Technical report*, Kongsberg Defence Systems.

Wong, J. Y. (2001). *Theory of Ground Vehicles*, John Wiley & Sons.

Xsens (2010). *MTi-G User Manual and Technical Documentation.* Document MT0137P, Revision H.

Appendix A

Equations for the Linearized Vehicle Model

Equations for the linearized vehicle model (Kiencke and Nielsen 2005)

$$\begin{aligned}\frac{\partial \dot{v}_{cg}}{\partial v_{cg}} &= \frac{1}{m_{cg}}\{-c_{air}A_{front}\rho_{air}v_{cg}\cdot\cos\beta \\ &\quad -\frac{l_{front}\dot{\psi}}{v_{cg}^2}\left(c_{FL}+c_{FR}\right)\cdot\sin\left(\delta-\beta\right) \\ &\quad -\left(c_{RL}+c_{RR}\right)\frac{l_{rear}\dot{\psi}}{v_{cg}^2}\cdot\sin\beta\Bigg\} \end{aligned} \tag{A.1}$$

$$\begin{aligned}\frac{\partial \dot{v}_{cg}}{\partial \beta} &= \frac{1}{m_{cg}}\{\left(F_{L,FL}+F_{L,FR}+c_{FL}+c_{FR}\right)\cdot\sin\left(\delta-\beta\right) \\ &\quad -\left(c_{RL}+c_{RR}+F_{L,RL}+F_{L,RR}-\frac{1}{2}c_{air}A_{front}\rho_{air}v_{cg}^2\right)\cdot\sin\beta \\ &\quad +\left(c_{FL}+c_{FR}\right)\left(\delta-\beta-\frac{l_{rear}\dot{\psi}}{v_{cg}}\right)\cdot\cos\left(\delta-\beta\right) \\ &\quad +\left(c_{RL}+c_{RR}\right)\left(-\beta+\frac{l_{rear}\dot{\psi}}{v_{cg}}\right)\cdot\cos\beta\Bigg\} \end{aligned} \tag{A.2}$$

$$\begin{aligned}\frac{\partial \dot{v}_{cg}}{\partial \dot{\psi}} &= \frac{1}{m_{cg} v_{cg}} \{ l_{front} \cdot \sin(\delta - \beta)(c_{FL} + c_{FR}) \\ &\quad + l_{rear} \cdot \sin\beta (c_{RL} + c_{RR}) \}\end{aligned} \tag{A.3}$$

$$\begin{aligned}\frac{\partial \dot{\beta}}{\partial v_{cg}} &= -\frac{1}{m_{cg} v_{cg}^2} \left\{ (c_{FL} + c_{FR}) \left(\delta - \beta - 2\frac{l_{front}\dot{\psi}}{v_{cg}} \right) \cdot \cos(\delta - \beta) \right. \\ &\quad + (F_{L,FL} + F_{L,FR}) \cdot \sin(\delta - \beta) \\ &\quad - \left(F_{L,RL} + F_{L,RR} + \frac{1}{2} c_{air} A_{front} \rho_{air} v_{cg}^2 \right) \cdot \sin\beta \\ &\quad \left. + (c_{RL} + c_{RR}) \left(-\beta + 2\frac{l_{rear}\dot{\psi}}{v_{cg}} \right) \cdot \cos\beta \right\}\end{aligned} \tag{A.4}$$

$$\begin{aligned}\frac{\partial \dot{\beta}}{\partial \beta} &= \frac{1}{m_{cg} v_{cg}} \left\{ (c_{FL} + c_{FR}) \left(\delta - \beta - \frac{l_{front}\dot{\psi}}{v_{cg}} \right) \cdot \sin(\delta - \beta) \right. \\ &\quad - (c_{FL} + c_{FR} + F_{L,FL} + F_{L,FR}) \cdot \cos(\delta - \beta) \\ &\quad - (c_{RL} + c_{RR}) \left(-\beta + \frac{l_{rear}\dot{\psi}}{v_{cg}} \right) \cdot \sin\beta \\ &\quad - (c_{RL} + c_{RR} + F_{L,RL} + F_{L,RR} \\ &\quad \left. - \frac{1}{2} c_{air} A_{front} \rho_{air} v_{cg}^2 \right) \cdot \cos\beta \right\}\end{aligned} \tag{A.5}$$

$$\begin{aligned}\frac{\partial \dot{\beta}}{\partial \dot{\psi}} &= \frac{1}{m_{cg} v_{cg}^2} \{ l_{rear} (c_{RL} + c_{RR}) \cdot \cos\beta \\ &\quad - l_{front} (c_{FL} + c_{FR}) \cdot \cos(\delta - \beta) \} - 1\end{aligned} \tag{A.6}$$

$$\begin{aligned}\frac{\partial \ddot{\psi}}{\partial v_{cg}} &= \frac{1}{I_z v_{cg}^2} \left\{ l_{front}^2 \dot{\psi} (c_{FL} + c_{FR}) \cdot \cos\delta \right. \\ &\quad + \frac{1}{2} l_{front} \dot{\psi} l_{tread} (c_{FL} - c_{FR}) \cdot \sin\delta \\ &\quad \left. + l_{rear}^2 \dot{\psi} (c_{RL} + c_{RR}) \right\}\end{aligned} \tag{A.7}$$

$$\frac{\partial \ddot{\psi}}{\partial \beta} = \frac{1}{I_z}\{-l_{front}\left(c_{FL}+c_{FR}\right)\cdot\cos\delta - \frac{1}{2}l_{tread}\left(c_{FL}-c_{FR}\right)\cdot\sin\delta + l_{rear}\left(c_{RL}+c_{RR}\right)\Big\} \tag{A.8}$$

$$\frac{\partial \ddot{\psi}}{\partial \dot{\psi}} = \frac{1}{I_z v_{cg}}\left\{-l_{front}^2\left(c_{FL}+c_{FR}\right)\cdot\cos\delta - \frac{1}{2}l_{front}l_{tread}\left(c_{FL}-c_{FR}\right)\cdot\sin\delta - l_{rear}^2\left(c_{RL}+c_{RR}\right)\right\} \tag{A.9}$$

$$\frac{\partial \dot{v}_{cg}}{\partial F_{L,FL}} = \frac{\cos\left(\delta-\beta\right)}{m_{cg}} \tag{A.10}$$

$$\frac{\partial \dot{v}_{cg}}{\partial F_{L,FR}} = \frac{\cos\left(\delta-\beta\right)}{m_{cg}} \tag{A.11}$$

$$\frac{\partial \dot{v}_{cg}}{\partial F_{L,RL}} = \frac{\cos\beta}{m_{cg}} \tag{A.12}$$

$$\frac{\partial \dot{v}_{cg}}{\partial F_{L,RR}} = \frac{\cos\beta}{m_{cg}} \tag{A.13}$$

$$\frac{\partial \dot{v}_{cg}}{\partial \delta} = -\frac{1}{m_{cg}}\{\left(F_{L,FL}+F_{L,FR}+c_{FL}+c_{FR}\right)\cdot\sin\left(\delta-\beta\right) + \left(c_{FL}+c_{FR}\right)\left(\delta-\beta-\frac{l_{front}\dot{\psi}}{v_{cg}}\right)\cdot\cos\left(\delta-\beta\right)\Big\} \tag{A.14}$$

$$\frac{\partial \dot{\beta}}{\partial F_{L,FL}} = \frac{\sin\left(\delta-\beta\right)}{m_{cg}v_{cg}} \tag{A.15}$$

$$\frac{\partial \dot{\beta}}{\partial F_{L,FR}} = \frac{\sin\left(\delta-\beta\right)}{m_{cg}v_{cg}} \tag{A.16}$$

$$\frac{\partial \dot{\beta}}{\partial F_{L,RL}} = \frac{-\sin\beta}{m_{cg} v_{cg}} \tag{A.17}$$

$$\frac{\partial \dot{\beta}}{\partial F_{L,RR}} = \frac{-\sin\beta}{m_{cg} v_{cg}} \tag{A.18}$$

$$\begin{aligned} \frac{\partial \dot{\beta}}{\partial \delta} &= \frac{1}{m_{cg} v_{cg}} \{ (c_{FL} + c_{FR} + F_{L,FL} + F_{L,FR}) \cdot \cos(\delta - \beta) \\ &\quad - (c_{FL} + c_{FR}) \left(\delta - \beta - \frac{l_{front}\dot{\psi}}{v_{cg}} \right) \cdot \sin(\delta - \beta) \Big\} \end{aligned} \tag{A.19}$$

$$\frac{\partial \ddot{\psi}}{\partial F_{L,FL}} = \frac{1}{I_z} \left\{ l_{front} \cdot \sin\delta - \frac{l_{tread}}{2} \cdot \cos\delta \right\} \tag{A.20}$$

$$\frac{\partial \ddot{\psi}}{\partial F_{L,FR}} = \frac{1}{I_z} \left\{ l_{front} \cdot \sin\delta + \frac{l_{tread}}{2} \cdot \cos\delta \right\} \tag{A.21}$$

$$\frac{\partial \ddot{\psi}}{\partial F_{L,RL}} = -\frac{l_{tread}}{2I_z} \tag{A.22}$$

$$\frac{\partial \ddot{\psi}}{\partial F_{L,RR}} = \frac{l_{tread}}{2I_z} \tag{A.23}$$

$$\begin{aligned} \frac{\partial \ddot{\psi}}{\partial \delta} &= \frac{1}{I_z} \{ l_{front} (c_{FL} + c_{FR}) \cdot \cos\delta \\ &\quad - \left[l_{front} (c_{FL} + c_{FR}) \left(\delta - \beta - \frac{l_{front}\dot{\psi}}{v_{cg}} \right) \right. \\ &\quad \left. + \frac{l_{tread}}{2} (F_{L,FR} - F_{L,FL}) \right] \cdot \sin\delta \\ &\quad + \frac{l_{tread}}{2} (c_{FL} - c_{FR}) \cdot \sin\delta \\ &\quad + \left[\frac{l_{tread}}{2} (c_{FL} - c_{FR}) \left(\delta - \beta - \frac{l_{front}\dot{\psi}}{v_{cg}} \right) \right. \\ &\quad \left. + l_{front} (F_{L,FR} + F_{L,FL}) \right] \cdot \cos\delta \\ &\quad - l_{rear} (c_{RL} + c_{RR}) \} \end{aligned} \tag{A.24}$$

Appendix B

Matlab-script for Importing Raw Data

```
1
2  %% clean up
3  close all
4  clear all
5  clc
6
7  %% load data
8  'Loading data...'
9  newData1 = importdata('logg_test_bil_aktiv_06_07_2010.txt');
10
11 % Create new variables in the base workspace from those fields.
12 for i = 1:size(newData1.colheaders, 2)
13     assignin('base', genvarname(newData1.colheaders{i}), newData1.data(:,i
       ));
14 end
15
16 %% fix data
17 'Fixing data orientation...'
18
19 % length of data
20 range = 1:length(seconds);
21
22 % find timestamp
23 time = zeros(length(range),1);
24 for i = 1:length(range)
25
26     time(i) = ...
27         hour(i)*60*60*1000+minute(i)*60*1000+seconds(i)*1000+nano(i)
             *10^-6;
28
29 end
```

```
30
31 % convert time to seconds
32 log.time = (time-min(time))/1000;
33
34 % rotation matrix from sensor frame to vehicle body frame
35 R = [1  0  0;
36      0  1  0;
37      0  0  1];
38
39 % create measurement vectors
40 acc = [acc_x acc_y acc_z]';
41 gyr = [gyr_x gyr_y gyr_z]';
42 vel = [vel_x vel_y vel_z/100]';
43 ori = [euler_roll euler_pitch euler_yaw]';
44
45 % allocate logdata vector size
46 log.acc = zeros(length(range),3);
47 log.gyr = zeros(length(range),3);
48 log.vel = zeros(length(range),3);
49 log.ori = zeros(length(range),3);
50
51 for i = 1:length(range)
52
53     % store inertial velocities
54     log.vel_in(i,:) = (vel(:,i))';
55
56     % store roll, pitch and yaw
57     log.ori(i,:) = ori(:,i)';
58     log.orgheading(i) = ori(3,i);
59
60     % find velocities in sensor frame
61     R_x = [1     0                    0;
62            0     cosd(log.ori(i,1)) -sind(log.ori(i,1));
63            0     sind(log.ori(i,1))  cosd(log.ori(i,1))];
64
65     R_y = [cosd(log.ori(i,2))    0    sind(log.ori(i,2));
66            0                     1    0;
67           -sind(log.ori(i,2))    0     cosd(log.ori(i,2))];
68
69     R_z = [cosd(log.ori(i,3))  -sind(log.ori(i,3)) 0;
70            sind(log.ori(i,3))   cosd(log.ori(i,3)) 0;
71            0                    0                  1];
72
73     log.vel_s(i,:) = ( (R_z*R_y*R_x)' * log.vel_in(i,:)')';
74
75     % rotate velocity and mesaurements from sensor frame to vehicle body
76     % frame
77     log.vel_b(i,:) = ( R * log.vel_s(i,:)' )';
78     log.acc(i,:)   = ( R * acc(:,i) )';
79     log.gyr(i,:)   = ( R * gyr(:,i) )';
80
81     % rotate yaw 180 degrees
82     log.ori(i,3) = ori(3,i) + 180;
83     if abs(log.ori(i,3) > 180)
84        log.ori(i,3) = log.ori(i,3) - 360*sign(log.ori(i,3));
85     end
86 end
87
```

```
% correct servo signal neutral point and convert to simulator
% actuator input (integer between 0-254)
log.steering = (servo1/500 - 1.54)*254 + 127;
log.throttle = (servo2/500 - 1.46)*254 + 127;

% transform longitud, latitude, altitude to NED frame
'Transform position to NED coordinates...'

% preallocate vectors
x_ned = zeros(length(range), 1);
y_ned = zeros(length(range), 1);

llh0 = [lon(1) lat(1) alt(1)];
for i=1:length(range)
   llh = [lon(i) lat(i) alt(i)];

   dned = dllh2dned(llh, llh0);

   x_ned(i) = dned(1);
   y_ned(i) = dned(2);

end

log.pos = [x_ned y_ned];

%% locate test data
'Locating tests in log data...'
numTests = 2;

for test = 1:numTests

    % look up test time range
    switch test
        case 1
            startsAtTime = 323;
            stopsAtTime = 330;
        case 2
            startsAtTime = 480;
            stopsAtTime = 515;
        case 3
            startsAtTime = 520;
            stopsAtTime = 550;
    end

    % find start and stop index
    StartIndex = 0;
    EndIndex = 0;
    i = 1;

    while ~StartIndex
        if log.time(i) > startsAtTime
            StartIndex = i;
        end
        i = i + 1;
    end

    while ~EndIndex
        if log.time(i) > stopsAtTime
```

```
146             EndIndex = i;
147         end
148         i = i + 1;
149     end
150 
151     testData(test).Time      = log.time(StartIndex:EndIndex);
152     testData(test).Acc       = log.acc(StartIndex:EndIndex, :);
153     testData(test).Gyr       = log.gyr(StartIndex:EndIndex, :);
154     testData(test).Vel_b     = log.vel_b(StartIndex:EndIndex, :);
155     testData(test).Vel_s     = log.vel_s(StartIndex:EndIndex, :);
156     testData(test).Vel_in    = log.vel_in(StartIndex:EndIndex, :);
157     testData(test).Ori       = log.ori(StartIndex:EndIndex, :);
158     testData(test).orgheading = log.orgheading(StartIndex:EndIndex);
159     testData(test).Pos       = log.pos(StartIndex:EndIndex, :);
160     testData(test).Throttle  = log.throttle(StartIndex:EndIndex);
161     testData(test).Steering = log.steering(StartIndex:EndIndex);
162 
163 end
164 
165 %% plot log data
166 'Creating plots...'
167 close all
168 
169 % which test to plot
170 test = 2;
171 
172 % plot map
173 figure()
174 plot(testData(test).Pos(:,2), testData(test).Pos(:,1))
175 grid on
176 axis tight
177 axis equal
178 xlabel 'East [m]'
179 ylabel 'North [m]'
180 title ('Map of Testdrive', 'Interpreter','Latex','Fontsize', 14)
181 
182 % add timestamps
183 for i = 1:100:length(testData(test).Pos(:,2))
184     text(testData(test).Pos(i,2), testData(test).Pos(i,1), ...
185         strcat('\leftarrow t=', int2str(testData(test).Time(i)), 's'))
186 end
187 set(gcf,'Position', [600   400   800   600])
188 
189 % plot data
190 figure()
191 
192 % plot velocity body
193 handle(1) = subplot(411);
194 hold on
195 plot(testData(test).Time, testData(test).Vel_b(:,1))
196 plot(testData(test).Time, testData(test).Vel_b(:,2) ,'r')
197 title ('Velocity', 'Interpreter','Latex','Fontsize', 14)
198 grid minor
199 ylabel 'Velocity [m/s]'
200 xlabel 'Time [s]'
201 legend ('V_x','V_y')
202 axis tight
203 
```

```
% plot accelerations
handle(2) = subplot(412);
hold on
plot(testData(test).Time, testData(test).Acc(:,1))
plot(testData(test).Time, testData(test).Acc(:,2) ,'r')
title ('Acceleration', 'Interpreter','Latex','Fontsize', 14)
ylabel 'Acceleration [m/s^2]'
xlabel 'Time [s]'
grid minor
legend ('Acc X','Acc Y')
axis tight

% plot actuator commands
handle(3) = subplot(413);
hold on
plot([testData(test).Time(1) testData(test).Time(end)], ...
    [127 127], 'g' ,'HandleVisibility','off')
plot(testData(test).Time, testData(test).Throttle(:,1))
plot(testData(test).Time, testData(test).Steering(:,1) ,'r')
title ('Control Inputs', 'Interpreter','Latex','Fontsize', 14)
ylabel 'Input Value [0-254]'
xlabel 'Time [s]'
grid minor
legend ('Throttle', 'Steering')
axis tight

% plot heading
handle(4) = subplot(414);
plot(testData(test).Time, testData(test).Ori(:,3) ,'b')
title ('Heading', 'Interpreter','Latex','Fontsize', 14)
ylabel 'Heading [degrees]'
xlabel 'Time [s]'
grid minor
legend 'Heading'
axis ([testData(test).Time(1) testData(test).Time(end) -180 180])

linkaxes(handle, 'x');
set(gcf,'Position', [800   100   707   900])
```

Appendix C

Feedback Linearization Controller Code

Below is the code implemented in the feedback linearization controller in SIMULINK:

```
function tau = fcn(speed, beta, psi_dot, speed_ref, psi_dot_ref, m_car,
  I_z)

m = m_car;
mv = m_car*speed;

% nonlinear system
g = [ cos(beta)/m   sin(beta)/m   0;
     -sin(beta)/mv  cos(beta)/mv  0;
      0             0             1/I_z];
f = [0 -psi_dot 0]';

% linear P-controller
kp_v = 4;
kp_b = 100;

v = [-kp_v*(speed - speed_ref);
     -kp_b*(psi_dot - psi_dot_ref);
      0];

tau = inv(g)*(-f + v);
```

The control allocation block is implemented as:

```
function [th, st]= fcn(tau, delta, beta, psi_dot, v, l_front)

K = [3 0;
     0 6];

T = [1 -sin(delta);
     0  cos(delta);
     0  l_front*cos(delta)];

u = K\(((T'*T)\T')*tau);

th = u(1);

alpha = u(2);
st = (alpha + beta + l_front*psi_dot/v);
```