# Autonomous R/C car

## A robust approach

Master's thesis in Systems, Control and Mechatronics

JOHAN LUND
FREDRIK MACINTOSH

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2018

Master's thesis 2018:EX041

# An investigative report on the use of robust design methods for autonomous R/C car trajectory control

JOHAN LUND

FREDRIK MACINTOSH


CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
*Division of Automation*
*Automatic Control Group*
Chalmers University of Technology
Gothenburg, Sweden 2018

An investigative report on the use of robust design methods for autonomous R/C car trajectory control
JOHAN LUND
FREDRIK MACINTOSH



Supervisor: Balázs Kulcsár
Examiner: Balázs Kulcsár

Master's Thesis 2018:EX041
Department of Electrical Engineering
Division of Automation
Automatic Control Group
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: *Beatitudo est celeritas machina.* Printer ink on plain white paper. A collaboration between Fredrik Macintosh (24) and Johanna Kulcsár (8).

Typeset in LaTeX
Gothenburg, Sweden 2018

An investigative report on the use of robust design methods for autonomous R/C trajectory control

FREDRIK MACINTOSH
JOHAN LUND
Department of Electrical Engineering
Chalmers University of Technology

# Abstract

Usually (nominal) models used for controller design are simplified neglecting many of the components of the real system that is described. This calls for design techniques where the controller ensures stability and performance for a set of (uncertain system) models rather than a single one.

This M.Sc. thesis aimed at using techniques that are intrinsically designed to meet robust design conditions. We implement robust $\mathcal{H}_\infty$ control design and experimentally test it on small-scale radio controlled cars that autonomously track a pre-defined trajectory.

First, in order to understand how robustness can be interpreted, the theory of $\mathcal{H}_\infty$, the principles of optimal control and their relation to signal and system norms are described.

Second, by using an experimental setup for self driving R/C cars, uncertain vehicle models are derived and controller objectives for tracking are defined. Disturbance, performance, and uncertainties (in terms of dynamic weightings) are added to the nominal design when creating an augmented system model. Both pure disturbance rejecting and reference tracking dynamic output feedback robust design concepts are used to develop 1- and 2- DoF (Degree of Freedom) controllers. Results are evaluated in a comparative experimental setup. This means that the designed $\mathcal{H}_\infty$-controllers are compared to the more common PID- and LQR-controllers, examining if there is any clear gain using a more complex theory. Data from real world experiments as well as simulations are analysed. Results show that the $\mathcal{H}_\infty$ control strategy is highly beneficial in terms of tracking performance at higher velocities and other variations of vehicle characteristics.

Keywords: $\mathcal{H}_\infty$, Robust control, Autonomous drive, Trajectory tracking, Uncertain systems.

# Acknowledgements

We would like to give special thanks to our supervisor Balázs Kulcsár for his constant desire to teach and great ability to do so.

To the Bachelor thesis group EENX15-18-23, consisting of Patrick Engström, Daniel Fredriksson, Ahmed Hamdy, Oscar Olesen, David Olsson and Sanna Sandberg. Thank you for the valuable insight and laughs.

To David Frisk for creating the LaTeX template used to format this report.

Finally to Johanna Kulcsár for the beautiful colouring of the front page.

Johan Lund, Gothenburg, June 2018
Fredrik Macintosh, Gothenburg, June 2018

# Contents

# List of Figures

# List of Tables

# 1
# Introduction

## 1.1 Background

Tracking performance of controllers is a highly sought after trait, after all, why use a controller in the first case if it doesn't yield desirable results? When mentioning control theory, one of the first things that comes to mind is probably the Proportional, Integral, Derivative- controller, known as the PID. PID control is the most common form of controller type in the world. In fact, around 95% of the world's control loops are PID (or rather PI) [1], and there's a reason for that. It's simple and versatile. A PID is for example a very suitable control type for cruise control in a car.

The PID controller typically has a large trade-off between performance and robustness. It is difficult to achieve both[2]. It performs badly under uncertain dynamics of a system. As is found, this can have devastating effects.

For more advanced systems, such as autonomous drive, Model Predictive Control (MPC) is currently the norm. MPC is however computationally heavy and might not be suited for all such applications. MPC can be designed to be robust[3], but again at the cost of computation. The method also relies of non-linear strategies, which might be undesirable simply because of the inherent complexity level. For this reason, a linear robust controller can often be an effective way to tackle the problem as it allows for performance while maintaining robustness.

## 1.2 Scope of Thesis

- The thesis aims to develop a miniature radio controlled (R/C) platform on which to evaluate the use of robust control. The goal is to design controllers that maintain a high tracking performance under increased velocities of the R/C car.
- The R/C car is restricted to movement in the plane, although 3D changes (e.g *pitch*) are partially accounted for.
- To ease testing and design, the vehicle platform is restricted to indoor use only and under very consistent traction conditions.
- All controller designs are strictly linear. Implementations include some very minor non-linearities.

## 1.3 Outline

We will start by describing the required theory for the project, this includes a few concepts of linear algebra, a description of the mathematical model used in the design and simulation. Next follows an introductory description of robust control as well as the more specific LQR (Linear Quadratic Regulator) and $\mathcal{H}_\infty$ parts. The thesis then continues with the description of the vehicle platform, what it includes, what it doesn't include and how it is realised. This includes descriptions of the hardware as well as the software. After that, results are given consisting of comparisons between simulations and reality for the different controllers, as well as the actual robustness. Finally, a small discussion of the newly found results is presented that ends with a conclusion, which should wrap up the thesis.

# 2
# Theory

## 2.1 Linear Algebra

The following section covers a few concepts of linear algebra that are very useful for control theory. They are also necessary in order to fully understand the following chapters on control theory.

### 2.1.1 Signals and System Norms

Comparing different signals is no different than comparing other different entities with each other, they have to be compared against the same framework. For signals (vectors) and systems (transfer functions) alike, it turns out that norms are an excellent way of normalising them. A norm is a function that yields a strictly positive size of a vector or matrix. The $p$-Norm is here, as in most other uses, denoted $||*||_p$ where $p$ is the dimension. The definition of the $p$-norm is as follows,

$$||*||_p \equiv \left(\sum_{i=1}^{n} |*_i|^p\right)^{\frac{1}{p}} \tag{2.1}$$

For a function applied to a quantity, vector, scalar, matrix, or transfer function, $X$ to be considered a norm, the following conditions have to hold

$$\boxed{\begin{aligned} ||X||_p &\geq 0 \text{ - Positive} && (2.2)\\ ||X||_p &= 0 \Leftrightarrow u(t) = 0,\ \forall t \text{ - Positive definite} && (2.3)\\ ||kX||_p &= |k|||X||,\ \forall k \in \mathbb{R}_{finite} \text{ - Scalability} && (2.4)\\ ||X+Y||_p &\leq ||X|| + ||Y|| \text{ - Triangle inequality} && (2.5) \end{aligned}}$$

Note that the fourth condition (and thereby norms) is only defined for $p > 1$. While norms may exist for infinitely many dimensions, a lot of them are simply impossible to calculate, so most often the $p = 1, 2$ and $\infty$ -norms are used. Following (2.1), the 2-norm for a signal $x(t)$ is denoted as a fraction of vector norms:

$$||x(t)||_2 = \left(\int |x(t)|^2 dt\right)^{\frac{1}{2}}. \tag{2.6}$$

The perceptive reader will note that this is indeed the definition of Euclidean distance.

A highly beneficial way of describing system norms is via a *induced* p-norm. Simply meaning that the system norm is induced by the signal norm in the following way

$$||A||_p := \sup_{x \neq 0} \frac{||Ax||_p}{||x||_p} \tag{2.7}$$

The induced $\mathcal{H}_\infty$ norm has a slightly different notation. It is the worst case gain of all possible $||A||_2$ gains. That is

$$||A||_\infty := \sup_{\substack{x \neq 0 \\ x \in \mathcal{L}_2}} \frac{||Ax||_2}{||x||_2} \tag{2.8}$$

where $\mathcal{L}_2$ is the set of all square integrable signals.

The reason for this being beneficial for the description of system behaviour is its similarity to the notion of a transfer function from an input to an output.

$$G(s) = \frac{Y(s)}{U(s)}$$

This makes it possible to interpret the system norm as a form of gain, i.e. the amplification of one selected input on another selected output, a metric crucial in robust control.

An important note on the matrix norm is that it's definition is slightly more strict than that of the vector norm. In addition to the 4 conditions stated above, the so called *consistency equation* has to be satisfied[4].

$$||AB||_p \leq ||A||_p \cdot ||B||_p \tag{2.9}$$

### 2.1.2 Singular Value

Whilst the matrix norm is a useful measure of absolute maximum gain of a system (matrix), it is unsuitable for any type of frequency analysis. For scalar systems (SISO, Single Input, Single Output), the bode plot is crucial for such analysis and is fairly straight forward. For matrices (MIMO, Multi Input, Multi Output) however, the bodeplot might not be sufficient. While it is perfectly valid, it analysis the system piece by piece. $Input_1 \rightarrow Output_1$, $Input_2 \rightarrow Output_1$, ..., $Input_n \rightarrow Output_m$ etc. (for $n$ inputs and $m$ outputs). In effect, it handles each and every subsystem as a SISO system, meaning it can't capture possible dynamics in between them.

The singular values of a matrix are also good measure of the size of a matrix, but unlike the matrix norm, they can be made frequency dependent by calculating the singular values of the frequency response of a system. The frequency dependent singular values of a matrix can be seen as an extension of the bode analysis, for MIMO systems.

Given a matrix $A \in \mathbb{R}^{m \times n}$ or $\in \mathbb{C}^{m \times n}$, it can be shown[4] that there exists unitary matrices

$$V \in \mathbb{R}^{n\times n} \text{ or } \in \mathbb{C}^{n\times n} \tag{2.10}$$
$$U \in \mathbb{R}^{m\times m} \text{ or } \in \mathbb{C}^{m\times m} \tag{2.11}$$
$$A = U\Sigma V^T. \tag{2.12}$$

Where

$$\Sigma = \begin{bmatrix} \Sigma_\sigma & 0 \\ 0 & 0 \end{bmatrix}, \Sigma_\sigma = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_k \end{bmatrix} \tag{2.13}$$

and

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k \geq 0,\ k = min(m, n) \tag{2.14}$$

$\sigma_i$ are the so-called *singular values* of $A$.

Now, if $A$ is given as a transfer function matrix, the singular values can be calculated for each and every desired frequency and plotted as such to form the MIMO version of the bodeplot and the same input-output gain relations can be analysed.

### 2.1.3 Spectral Radius

The spectral radius of a matrix (system) $G$ is defined as the magnitude of the largest eigenvalue of said matrix $G$.

$$\rho(G) = max|\lambda_i(G)| \tag{2.15}$$

While it resembles a norm, it is not a norm as it does not satisfy conditions 3 of (2.2) or (2.9). Even though it is not a norm, it has a useful property. It sets a lower bound on any matrix norm.

### 2.1.4 Linear Fractional Transformations

Linear fractional transformations (LFT) as it turns out, are useful for interconnecting matrices with each other.

Consider the matrix A, with dimensions $(n_1 + n_2) \times (m_1 + m_2)$ partitioned as follows

$$P = \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix} \tag{2.16}$$

Now let two other matrices, called $\Delta$ and $K$ (the somewhat peculiar choice of names becomes apparent later) with dimension $m_1 \times n_1$ and $m_2 \times n_2$ respectively. The linear fractional transformation is now defined as follows

$$F_l(P,K) \triangleq P_{11} + P_{12}K(I - P_{22}K)^{-1}P_{21} \tag{2.17}$$

$$F_u(P,\Delta) \triangleq P_{22} + P_{21}\Delta(I - P_{11}\Delta)^{-1}P_{12} \tag{2.18}$$

where $u$ and $l$ denote the upper and lower half of $P$. Note that the invertability of $(I - P_{22}K)$ and $(I - P_{11}\Delta)$ is a strict criteria for the LFT to exist.

## 2.2 Dynamic Vehicle Model

Controller design doesn't necessary need a detailed model, however robust control in the form of $\mathcal{H}_\infty$ depends upon it. A proper model also provides a firm basis for analysis able to identify various aspects of the system that affects the controller design and performance. The ideal model should of course describe the real system as good as possible. However, since the tools used for analysis and calculation of $\mathcal{H}_\infty$-controllers are based on linear systems, the model must also be based on linear equations. The model should if possible also make use of parameters that are readily available in the actual system so that they can be easily measured. Any model is also easier to understand and therefore analyse if the physical equation are somewhat intuitive.

Since vehicle dynamics is a well explored area, the models described in *Vehicle Dynamics and Control* by Rajesh Rajamani [5] were used as foundation.

### 2.2.1 Bicycle Model of Lateral Vehicle Dynamics

To describe the dynamic motion of the vehicle a Two degree of freedom (2DoF) bicycle model is used as described by Rajamani [5]. Such a model describes the lateral and rotational dynamics of the vehicle under the assumption of constant forward velocity and small steering, and thereby also small slip angles. It also assumes dry conditions, meaning a friction coefficient of 1.

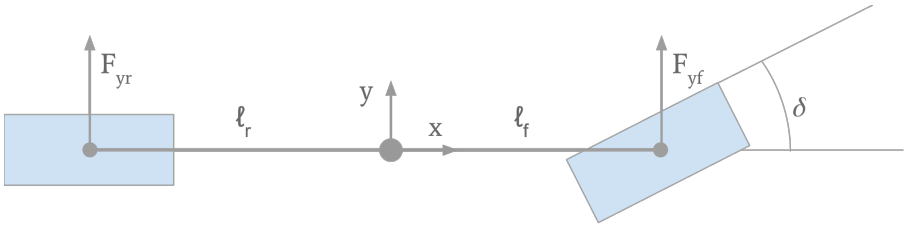



**Figure 2.1:** 2DoF bicycle model

The model consists of a front and a rear tire. It has a mass $m$ and the distance to the CoG from the front and rear axis are $\ell_f$ and $\ell_r$ respectively. The front wheel is steerable with the angle $\delta$. To give the bicycle the ability to rotate, $\ell_f$ and $\ell_r$ are seen as levers and the friction between the tires and the road is the force acting upon them under the assumption of no skid[6].

$$ma_y(t) = F_{yf}(t) + F_{yr}(t) \tag{2.19}$$

$a_y(t)$ is composed of the lateral acceleration $\ddot{y}(t)$ and the centripetal acceleration $V_x\dot{\psi}(t)$, where $V_x$ is the longitudinal velocity and $\dot{\psi}$ is the yaw rate of the vehicle. $F_{yf}(t)$ and $F_{yr}(t)$ are the lateral tire forces.

$$m(\ddot{y}(t) + V_x\dot{\psi}(t)) = F_{yf}(t) + F_{yr}(t) \tag{2.20}$$

Using moment balance around the z axis results in the following description for the yaw dynamics, where the difference in force between front and rear tires yields a torque around the CoG (center of gravity).

$$I_z\ddot{\psi}(t) = \ell_f F_{yf}(t) - \ell_r F_{yr}(t) \tag{2.21}$$

Since the tire forces are non-linear, a linear approximation has to be used. For small slip angles, the tire force is proportional to the slip angle. The slip angle of the tire is the difference between the steering angle and the heading of the tire, the tire will resist the twisting motion depending on its stiffness. In figure 2.2 the difference for the front wheel is calculated as

$$\alpha_f(t) = \delta(t) - \theta_{Vf}(t).$$

The slip of the rear tire is similar, except it can't be turned.

$$\alpha_r(t) = -\theta_{Vr}(t),$$

where $\theta_{Vf}(t)$ and $\theta_{Vr}(t)$ are the heading angles of the tires with respect to the longitudinal axis of the vehicle.

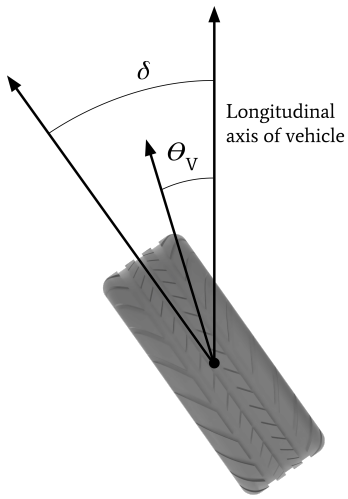



**Figure 2.2:** Illustration of tire slip angle

The non-linear equations for the tire heading angle are as follows

$$tan(\theta_{Vf}(t)) = \frac{V_y(t) + \ell_f\dot{\psi}(t)}{V_x} \tag{2.22}$$

$$tan(\theta_{Vr}(t)) = \frac{V_y(t) - \ell_r\dot{\psi}(t)}{V_x} \tag{2.23}$$

which for small angles simply yield

$$\theta_{Vf}(t) = \frac{V_y(t) + \ell_f \dot{\psi}(t)}{V_x} \tag{2.24}$$

$$\theta_{Vr}(t) = \frac{V_y(t) - \ell_r \dot{\psi}(t)}{V_x} \tag{2.25}$$

The angle of the tires velocity angle is hence the ratio of the lateral velocity and an addition from the yaw rate of the vehicle, and its longitudinal velocity.

The only thing left to calculate the force is the tire stiffness, $C_\alpha$.

$$F_{yf}(t) = 2C_{yf}(\delta(t) - \theta_{Vf}(t)) = 2C_{yf}\alpha_f(t). \tag{2.26}$$

$$F_{yr}(t) = 2C_{yr}(-\theta_{Vr}(t)) = 2C_{yr}\alpha_r(t). \tag{2.27}$$

where the 2 is to account for the fact that the bicycle model has to be augmented with additional wheels in order du more correctly model a car.

Linearising around $V_y(t) = 0$, $\dot{\psi}(t) = 0$ and a constant velocity $V_x$, the state space form of the bicycle model is as follows,

$$\begin{bmatrix} \dot{y}(t) \\ \ddot{y}(t) \\ \dot{\psi}(t) \\ \ddot{\psi}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{2C_{\alpha f}+2C_{\alpha r}}{mV_x} & 0 & -V_x - \frac{2C_{\alpha f}\ell_f - 2C_{\alpha r}\ell_r}{mV_x} \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{2C_{\alpha f}\ell_f - 2C_{\alpha r}\ell_r}{I_z V_x} & 0 & -\frac{2C_{\alpha f}\ell_f^2 + 2C_{\alpha r}\ell_r^2}{I_z V_x} \end{bmatrix} \begin{bmatrix} y(t) \\ \dot{y}(t) \\ \psi(t) \\ \dot{\psi}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{2C_{\alpha f}}{m} \\ 0 \\ \frac{2C_{\alpha f}\ell_f}{I_z} \end{bmatrix} \delta(t) \tag{2.28}$$

.

**Table 2.1:** States of the lateral dynamics bicycle model

| | |
|---|---|
| $\delta(t)$ | Steering angle of the wheel(s). [*radians*] |
| $y(t)$ | The vehicle lateral position is the perpendicular distance from the center of mass of the car to the tangent of the road curvature. [$m$] |
| $\psi(t)$ | Yaw angle; orientation angle of the vehicle with respect to global X axis. [*radians*] |

This model is the starting point of the model used. A more suitable definition is however described next which allows for easier introduction of reference signals.

### 2.2.2 Dynamic Model in Terms of Error with Respect to the Road

To be able to follow a desired track, the model is expressed in terms of errors measuring the deviation from the track [5]. Just as the dynamic lateral vehicle model expressed in terms of position and heading, the error based model is linearised around a constant velocity $V_x$. This model uses the same assumptions and equations as the model described in 2.2.1 but expresses the states as deviations from the reference path. Since the desired yaw rate is described by the track, with it's own

dynamics, it is separated into its own matrix description showing how it enters the dynamics.

**Table 2.2:** How lateral error and yaw error are derived

| | | |
|---|---|---|
| $e_1(t)$ | Lateral position error with respect to road. | $\ddot{e}_1(t) = \ddot{y}(t) + V_x(\dot{\psi}(t) - \dot{\psi}_{des})$ |
| $e_2(t)$ | Yaw angle error with respect to road. | $e_2(t) = (\psi(t) - \psi_{des})$ |
| $\dot{\psi}_{des}$ | Desired yaw rate determined from road radius $R$. | $\dot{\psi}_{des} = \frac{V_x}{R}$ |

$$\dot{e}(t) = A_e e(t) + B_{e1}\delta(t) + B_{e2}\dot{\psi}_{des} \tag{2.29}$$

$$\begin{aligned}\begin{bmatrix}\dot{e}_1(t)\\ \ddot{e}_1(t)\\ \dot{e}_2(t)\\ \ddot{e}_2(t)\end{bmatrix} &= \begin{bmatrix}0 & 1 & 0 & 0\\ 0 & -\frac{2C_{\alpha f}+2C_{\alpha r}}{mV_x} & \frac{2C_{\alpha f}+2C_{\alpha r}}{m} & \frac{-2C_{\alpha f}l_f+2C_{\alpha r}l_r}{mV_x}\\ 0 & 0 & 0 & 1\\ 0 & -\frac{2C_{\alpha f}l_f-2C_{\alpha r}l_r}{I_zV_x} & \frac{2C_{\alpha f}l_f-2C_{\alpha r}l_r}{I_z} & \frac{2C_{\alpha f}l_f^2+2C_{\alpha r}l_r^2}{I_zV_x}\end{bmatrix}\begin{bmatrix}e_1(t)\\ \dot{e}_1(t)\\ e_2(t)\\ \dot{e}_2(t)\end{bmatrix}\\ &+ \begin{bmatrix}0\\ \frac{2C_{\alpha f}}{m}\\ 0\\ \frac{2C_{\alpha f}l_f}{I_z}\end{bmatrix}\delta(t) + \begin{bmatrix}0\\ -\frac{2C_{\alpha f}l_f-2C_{\alpha r}l_r}{mV_x} - V_x\\ 0\\ -\frac{2C_{\alpha f}l_f^2-2C_{\alpha r}l_r^2}{I_zV_x}\end{bmatrix}\dot{\psi}_{des}\end{aligned} \tag{2.30}$$

This error model now has the benefit of an added reference signal $\dot{\psi}_{des}$. As seen when comparing the error based model, (2.30), with the local lateral model, (2.2.1), the input matrix for $\delta(t)$ ($B_1$) remains the same in both models. This is a natural consequence since, even though the description changes somewhat, the input dynamics of the system remains the same.

### 2.2.3 Simplified Yaw and Yaw Rate Based Model

As a result of the track description used, see section 3.5.1, the lateral distance is no longer important, since any *path* between way points is allowed. In practice this means that the lateral error will always be zero, resulting in the removal of the lateral states. Which results in the following state space representation

$$\begin{aligned}\begin{bmatrix}\dot{e}_2(t)\\ \ddot{e}_2(t)\end{bmatrix} &= \begin{bmatrix}0 & 1\\ \frac{2C_{\alpha f}l_f-2C_{\alpha r}l_r}{I_z} & \frac{2C_{\alpha f}l_f^2+2C_{\alpha r}l_r^2}{I_zV_x}\end{bmatrix}\begin{bmatrix}e_2\\ \dot{e}_2\end{bmatrix}\\ &+ \begin{bmatrix}0\\ \frac{2C_{\alpha f}l_f}{I_z}\end{bmatrix}\delta(t) + \begin{bmatrix}0\\ -\frac{2C_{\alpha f}l_f^2-2C_{\alpha r}l_r^2}{I_zV_x}\end{bmatrix}\dot{\psi}_{des}.\end{aligned} \tag{2.31}$$

With this simplified model only the dynamic related to yaw and yaw rate are retained. While this makes it's impossible to directly control the position, it does provide a way to see the overall movement of the vehicle as a result of the current heading and it's relation to waypoints.

### 2.2.4 Track Representation within the Model

Using a block diagram to illustrate the system described by the expression in (2.31), in feedback with a controller, it is clear that the track reference affects the error state directly but also that all the errors evolve in a in a fashion dependant upon each other. This is to be expected, a yaw error will, for example, lead to a position error over time. This means it is keeping the car close to the desired position by using only the orientation.

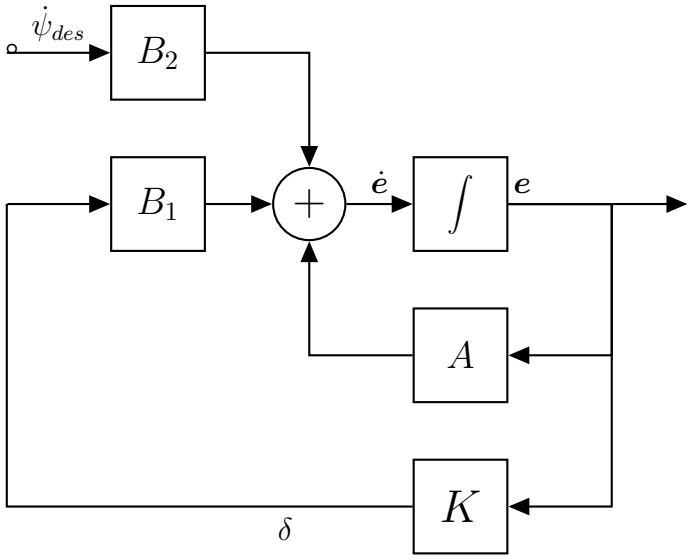



**Figure 2.3:** Block diagram of the error based model

When the controller in implemented in a real world application the description of how the reference enters the system can be described in a different way. The error is still described by $\dot{e}(t) = Ae(t) + B_1\delta(t) + B_2\dot{\psi}_{des}$ but is sampled as the actual estimated states of the vehicle subtracted from a reference orientation described by the track. This representation of the system is illustrated by the block diagram in Figure 2.4, where $x(t)$ is a vector describing the orientations, $t$ is the track described in the same global coordinate frame and $T$ describes how the error vector $e(t)$ is made up from a comparison of the vehicle orientation and the desired track orientation. $P$ is the car model found in section 2.2.1, but that too stripped of its two lateral states. Note that the system $T$ is not described in any detail. It just represents that the track itself can be seen as dynamics shaping the output of the dynamics of the car into new states described as errors from the wanted states.

Even though $\dot{\psi}_{des}$ can be seen as a reference signal, it enters the system as a disturbance that affects the states as described by matrix $B_2$. This means that the feedback controlled system $(A - B_1K)$ will not be able to converge the tracking errors to zero [5].

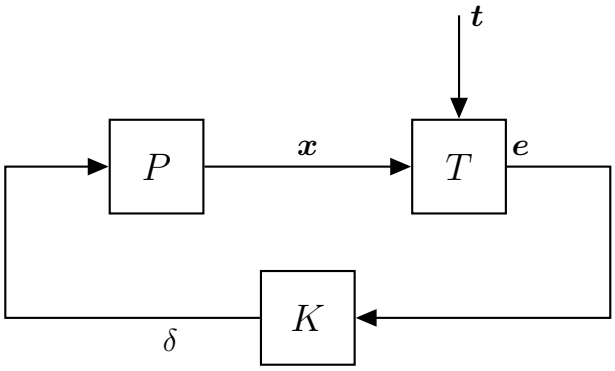



**Figure 2.4:** Block diagram illustrating how the track is described as an error. Where the dynamics from the track $\boldsymbol{t}$ to the states $\boldsymbol{e}(t)$ lies within $T$.

The following figure provides a more visual representation of the system in terms of local and global coordinates and the track.

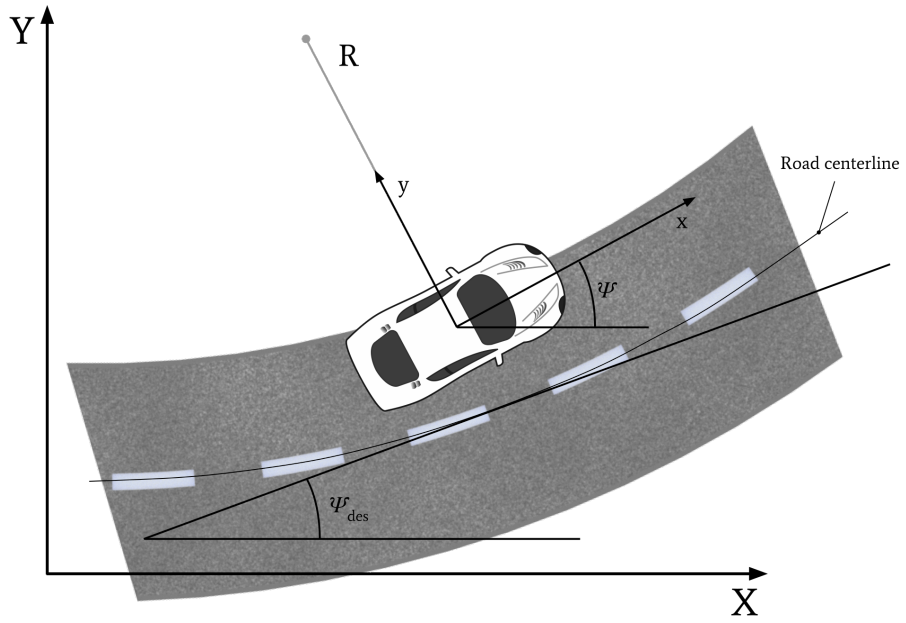



**Figure 2.5:** A representation of the vehicles local coordinate frame in a global frame with track description included

## 2.3 Robust Control

The principle of robust control is an area of control that targets uncertainties and/or uncertain dynamics of systems. These could range from simple measurement noise to the more extreme case of completely broken components of a system. Different types of controllers yield different levels of robustness, even controllers that are not robust by nature (by nature here meaning by mathematical design) can be robust under certain circumstances. While this level of robustness might be sufficient for many applications, there is no way of guaranteeing tracking performance.

Consider the exemplary explanation of the impact of uncertain parameters and performance criteria below. To keep it simpler, it uses a single performance weight along with the parametric uncertainties as well as a weighted disturbance input. Should it be required, more can of course be added.

The effect of uncertain parameters can be seen as a multiplicative (or additive) error on either the input or the output of a nominal plant $P_{nom}$. Output multiplicative uncertainty is fairly intuitive as it is simply the (hopefully) expected nominal output and some relative errors added, forming the following

$$P_{real} = (I + W_1 \Delta W_2) P_{nom}. \tag{2.32}$$

For simplifying purposes, consider a scalar plant, $p_{real}$ is in that case a signal comprised of the nominal output, $p_{nom}$, and a weighted additional response added to it, $(w_1 \Delta w_2) p_{nom}$. $w_1$ and $w_2$ shapes the additional response in both frequency and magnitude and $\Delta$ norm bounds the signal ($||\Delta||_\infty \leq 1$), as stated in 2.1.1, allowing signals to be compared regardless of type (this notion makes more sense in a MIMO/SIMO situation where the state outputs often are different units). Finally an additional weight $w_2$ can be added to further shape the signal if desired.

The Block Schematics of this is the following figure 2.6

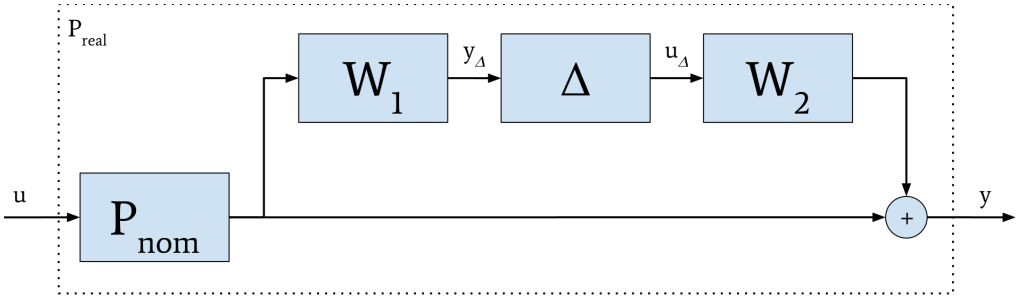



**Figure 2.6:** Nominal plant with parametric uncertainty described as multiplicative output error

The weights $W_1$ and $W_2$ are design criteria, but they should strictly match the expected parametric uncertainty or the controller will either not be able to handle the errors or it will over-compensate, which might result in a controller difficult to realise. This requires the weights to match the resulting frequency behaviour of the parametric errors as well as magnitude discrepancies.

Having defined the parametric uncertainty of the system, focus is now on performance. While the controller primarily has to remain stable for all possible perturbations resulting from the changes in parameters, some level of performance is desired. To analyse performance, some or all outputs are chosen as so-called performance outputs. This output is weighted according to desired levels of performance for each signal, with added (weighted) disturbance and added to the system, see figure 2.7.

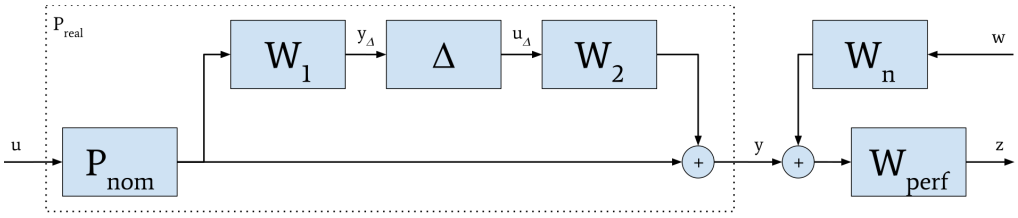



**Figure 2.7:** Plant $P_{nom}$ with added parametric uncertainties, performance output and disturbance weighting

For the next couple of steps, a new structure is defined, the so-called $\Delta PK$ structure. Given the perturbations $\Delta$ above, it is possible to 'extract' these signals and give them their own block. Knowing that

$$||\Delta||_\infty \leq 1$$

, the desired stability analysis of the system can then be done without violating the small gain theorem. To do this, a slight adjustment to the plant given in figure 2.7 is made, see figure 2.8

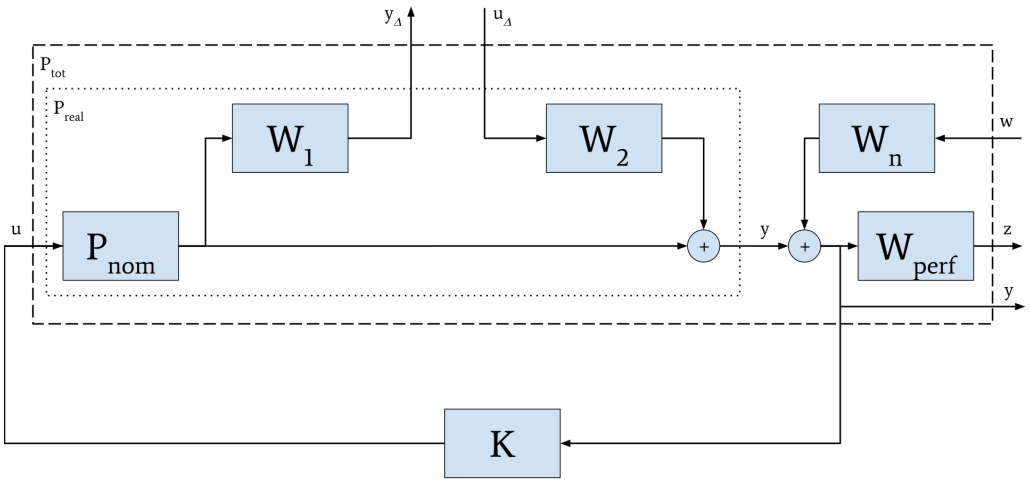



**Figure 2.8:** Plant $P_{tot}$

As evident from the figure, the $\Delta$-block has been extracted, and its input and output are now instead included into $P_{tot}$. A controller $K$ has also been added to the description. Inputs to the plant are the normalised perturbations $u_\Delta$, the exogenous disturbances $w$ and the control signal $u$. Outputs of the plant are the weighted perturbations $y_\Delta$, the so-called performance outputs $z_n$ and the measured output $y$.

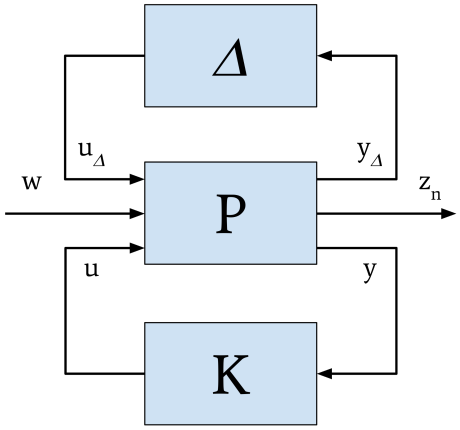



**Figure 2.9:** $\Delta PK$ representation of a system

To analyse this in terms of robustness, yet another structure is used, the $N\Delta$-structure. This structure considers the control signal $u$ to be an internal signal and thus creates a more suitable basis of analysis for the robustness as the control signal is excluded from the loop transfers.

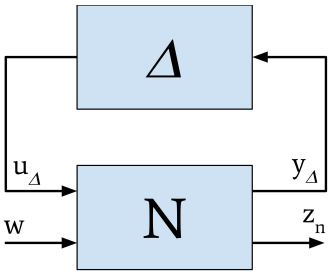



**Figure 2.10:** $\Delta N$ representation of a system

The $N$ is derived by deriving all the loop transfer functions of figure 2.8 above while treating the control signal as an internal signal (ergo not an input), yielding

$$N = \begin{bmatrix} N_{11} & N_{21} \\ N_{12} & N_{22} \end{bmatrix} = \begin{bmatrix} -W_1 P_{nom} K (I + P_{nom} K)^{-1} W_2 & -W_1 K (I + P_{nom} K)^{-1} W_n \\ -W_{perf} P_{nom} K (I + P_{nom} K)^{-1} W_2 & -W_{perf} P_{nom} K (I + P_{nom} K)^{-1} W_n \end{bmatrix} \tag{2.33}$$

The same result is achieved via LFT, $N = F_l(P_{tot}, K)$, see section 2.1.4, given the inputs

$$\begin{bmatrix} u_\Delta \\ w \end{bmatrix}$$

(the control signal is a part of the loop and hence not considered an input to the system) and the outputs

$$\begin{bmatrix} y_\Delta \\ z \end{bmatrix}.$$

To verify and guarantee performance and stability, a few metrics are defined, namely:

- **NS - Nominal stability.**
  - Given the plant $P_{tot}$ (perturbations $\Delta = 0$), is the controller $K$ providing internal stability?
  - Verification: Internal stability of $N$.
    *Note: Stability analysis can of course be done on $P_{nom}$ directly, as the added weight simply scales the output and thus does not affect stability*
- **NP - Nominal performance**
  - Given NS, are the performance criteria met for the nominal plant $P_{nom}$?
  - Verification: $||N_{22}||_\infty < 1$
- **RS - Robust stability**
  - Given the uncertain plant, effectively a set of plants, $P_{unc}$, is the controller $K$ providing internal stability?
  - Verification: $||N_{11}||_\infty < 1$
- **RP - Robust performance**
  - Given RS, are the performance criteria still met?
  - Verification: $||N||_\infty \leq 1$

### 2.3.1 The Standard Linear Quadratic Regulator

Given a state space representation of a time-invariant system on the form

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \tag{2.34}$$

where $x(t)$ are the states and $y(t)$ are the measured outputs. A is the transition matrix describing the linear or linearised differential equations of the modelled system. B is the input transition matrix, it is formed in the same way as A, but describes how the input propagates through the states. C is the output matrix, and describes how the states, together with the input and the feedthrough matrix D effects the output of the system.

In order to achieve the desired states, the control input can be designed to do so. This is done by ensuring that the control policy of the system relocates the poles (eigenvalues) of the closed loop system, $(A - BK)x(t)$, to satisfactory positions.

Placing these poles turns out to be an excellent use for optimisation, specifically Lagrangian optimisation. By creating a cost functional on the form

$$J = \frac{1}{2}\int_0^\infty x(t)^T Q x(t) + u(t)^T R u(t) dt = \int_0^\infty V(x(t), u(t)), \tag{2.35}$$

the poles can be located optimally in the sense of $Q$ and $R$.

$Q \in \mathcal{R}^{n\times n}$ and sets the level of white noise process noise, this is what yields the robustness of the LQR. It can perhaps more intuitively be seen as the cost of having the state incorrect where as $R \in \mathcal{R}^{m\times m}$ sets the cost of control signal use. $n$ being the number of states and $m$ being the number of inputs.

The task is now to minimise $J$ by the location of poles. This is done by solving the Euler-Lagrange equation,

$$\left[\frac{\partial V}{\partial x(t)}\right]^* - \frac{d}{dt}\left[\frac{\partial V}{\partial \dot{x}(t)}\right]^* = 0,\ \forall t \in [t_0, t_f] \tag{2.36}$$

where $t_0$ is the initial time and $t_f$ is the final time.

To solve this equation, or rather the optimisation problem that it describes, first a few assumptions have to be made regarding the system:

- The control signal is unconstrained, that is to say $u(t) \in [-\infty, \infty]$
- The initial conditions $x_0 = x(0)$ are given.
- The two weighting matrices $Q$ and $R$ are symmetric and positive semi-definite. $R$ also has to be strict positive definite.
- The LTI system has to be detectable (any unobservable states of the system must be asymptotically stable) and stabilisable (any uncontrollable states of the system must be asymptotically stable).

Under these assumptions, and additionally the assumption of an optimal solution the Hamiltonian of the system is formed,

$$\begin{aligned} &H^*(x(t), u(t), \lambda(t)) = \\ &\frac{1}{2}x^{*T}(t)Qx^*(t) + \frac{1}{2}u^{*T}(t)Ru(t) + \lambda^{*T}(t)(Ax^*(t) + Bu^*(t)) = \\ &V(x^*(t), u^*(t)) + \lambda^T(t)(Ax^*(t) + Bu^*(t)) \end{aligned} \tag{2.37}$$

where $\lambda(t)$ is the well known Lagrange multiplier, here denoted co-states.

To find an optimal solution, there are a few conditions that need to be fulfilled[7].

$$\left(\frac{\partial H}{\partial u(t)}\right)^* = 0 \tag{2.38}$$

$$\left(\frac{\partial H}{\partial \lambda(t)}\right)^* = \dot{x}^* \tag{2.39}$$

$$\left(\frac{\partial H}{\partial x(t)}\right)^* = -\dot{\lambda}^*(t) \tag{2.40}$$

$$\left[H^* + \left(\frac{\partial S}{\partial t}\right)^*\right]_{t_f} \delta t_f + \left[\left(\frac{\partial S}{\partial x(t)}\right)^* - \lambda^*(t)\right]_{t_f}^T \delta x_f = 0 \tag{2.41}$$

The last condition is for this case fulfilled automatically. The $S$-term relates to the more general case of LQR control where the system has a desired final state and time and cost can be applied to those as well. Since this particular case is a so called infinite-time optimal control problem, this term is 0 (there can be no cost on a final state if there is none).

The optimal controller is given from the first condition as a function of the states and the co-states as:

$$\left(\frac{\partial H}{\partial u(t)}\right)^* = Ru^*(t) + B^T\lambda^*(t) = 0 \implies u^*(t) = -R^{-1}B^T\lambda(t). \tag{2.42}$$

This optimal controller is now inserted into the second condition, yielding the following

$$\left(\frac{\partial H}{\partial \lambda(t)}\right)^* = \dot{x}^*(t) = Ax^*(t) + Bu^*(t) = Ax^*(t) - B\underbrace{R^{-1}B^T\lambda^*(t)}_{u^*} \tag{2.43}$$

And finally, the third condition results in

$$-\lambda^*(t) = \left(\frac{\partial H}{\partial x}\right)^* = Qx^*(t) + A^T\lambda^*(t). \tag{2.44}$$

In (2.42) the optimal controller is expressed as a function of the co-state $\lambda(t)$. Since however a state feedback controller is desired, this is merely a help along the way. What is needed is a way to relate the co-state $\lambda(t)$ and the state $x(t)$. Therefore a linear transformation matrix is assumed to exist between the optimal state and the optimal co-state,

$$\lambda^*(t) = P(t)x^*(t) \tag{2.45}$$

Next step is to differentiate the new co-state equation and repeat the previously stated optimality conditions.

$$\dot{\lambda}^*(t) = \dot{P}(t)x^*(t) + P(t)\dot{x}^*(t) \tag{2.46}$$

$$\dot{x}^*(t) = Ax^*(t) - BR^{-1}B^T\underbrace{\lambda^*(t)}_{P(t)x^*(t)} = Ax^*(t) - BR^{-1}B^TP(t)x^*(t) \tag{2.47}$$

$$\dot{\lambda}^*(t) = -Qx^* - A^T\lambda^*(t) = -Qx^*(t) - A^TPx^*(t) \tag{2.48}$$

Substituting (2.47) and (2.48) into (2.47) yields the following form,

$$-Qx^*(t) - A^TP(t)x^*(t) = \dot{P}(t)x^*(t) + P(t)\left(Ax^*(t) - BR^{-1}B^TPx^*(t)\right) \tag{2.49}$$

By means of algebra gymnastics, this can be written as

$$x^*(t)\left(\dot{P}(t) + P(t)A + A^TP(t) + Q - P(t)B^{-1}B^TP(t)\right) = 0 \tag{2.50}$$

This is still however dependent on the state, what is desired is a solution valid for arbitrary states $x^*$. To achieve this, a solution to the following nonlinear differential equation has to be found

$$\dot{P}(t) + P(t)A + A^TP(t) + Q = P(t)BR^{-1}B^TP(t) \tag{2.51}$$

which is indeed on the form of a matrix Riccati[8] equation, in fact, it is called the *matrix differential Riccati equation*.

$P(t)$ is obviously time-dependent, it can however be shown[7] that $P(t)$ remains near constant while $t << t_{final}$. In the case of the very common infinite-time control problem, $t_{final} = \infty$ meaning that any and all lengths of time can be considered $<< t_{final}$ and therefor the solution to $P(t)$ is constant. Effectively, $\lim_{t\to\infty} P(t) = P$.

The controller $K$ is then given by

$$K = R^{-1}B^T P \tag{2.52}$$

Finally yielding the state feedback controller as

$$\begin{aligned} \dot{x}(t) &= Ax(t) - BKy(t) + Br(t) \\ y(t) &= Cx(t) - DKy(t) + Dr(t) \\ u(t) &= Ky(t) \end{aligned} \tag{2.53}$$

### 2.3.2 The $\mathcal{H}_\infty$ Control Problem

To ease the transition to $\mathcal{H}_\infty$ controller which works on the basis of norms, the LQR controller can be described in terms of a norm minimisation, namely the 2-norm. Following (2.35) and (2.1), the cost functional can be defined as

$$J = \frac{1}{2}\left(||Q^{\frac{1}{2}}x(t)||_2^2 + ||R^{\frac{1}{2}}u(t)||_2^2\right) \tag{2.54}$$

Also working off of the basis of norms is the $\mathcal{H}_\infty$-controller. The sole purpose of the $\mathcal{H}_\infty$ synthesis is to find a stabilising controller that, given the uncertain plant, ensures

$$\begin{aligned} &||N||_\infty \leq \gamma_{min}. \\ &||N||_\infty = ||F_\ell(P,K)||_\infty = \max_{w(t)\neq 0} \frac{||z(t)||_2}{||w(t)||_2} \end{aligned} \tag{2.55}$$

The effect of this is that for any and all perturbations, exogenous disturbances and measurement noise within the specified bounds will be attenuated to such a degree that the overall gain of the system (the $\mathcal{H}_\infty$-norm) never exceeds $\gamma_{min} < 1$. Where $\gamma_{min}$ is the cost of the optimal controller. That is to say, the system remains stable while performance criteria are met.

Finding such an optimal controller is in theory great, but in reality often not necessary or computationally difficult. Instead, a sub-optimal $\mathcal{H}_\infty$-controller is calculated where $\gamma > \gamma_{min}$ and $\gamma < 1$ is still a requirement to maintain stability according to the small gain theorem.

The task of finding the suboptimal controller is easier to explain if the system is given on a more standard state-space representation.

If a plant $P$ is given as a transfer function matrix (weights are often comfortably described as transfer functions) it first has to be translated to state-space form. If $P$ is a state space (weights are defined as state-spaces), it is possible to partition the plant giving the following structure,

$$P = \left[\begin{array}{c|cc} A & B_1 & B_2 \\ \hline C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{array}\right]$$

Given this standard structure, the task of finding a controller depends on a fulfilling a few criteria. It can be shown [4][9] that such a controller exists if and

only if **I, II and III** are met.

- **(I)** $X_\infty \geq 0$ is a solution to the algebraic riccati equation:
  - $A^T X_\infty + X_\infty A + C_1^T C_1 + X_\infty(\gamma^{-2} B_1 B_1^T - B_2 B_2^T)X_\infty = 0$, such that $Re\ \lambda_i(A + (\gamma^{-2} B_1 B_1^T - B_2 B_2^T)X_\infty) < 0$. Where $\lambda_i(\cdot)$ are the eigenvalues.
- **(II)** $Y_\infty \geq 0$ is a solution to the algebraic riccati equation:
  - $A^T Y_\infty + Y_\infty A + B_1 B_1^T + Y_\infty(\gamma^{-2} C_1^T C_1 - C_2^T C_2)Y_\infty = 0$, such that $Re\ \lambda_i(A + Y_\infty(\gamma^{-2} C_1^T C_1 - C_2^T C_2)X_\infty) < 0$.
- **(III)** The spectral radius of $X_\infty Y_\infty$ never exceeds $\gamma^2$
  - $\rho(X_\infty Y_\infty) < \gamma^2$

From this a controller structure is created,

$$K = \left[\begin{array}{c|cc} A_\infty & -Z_\infty L_\infty & Z_\infty B_2 \\ \hline F_\infty & 0 & I \\ -C_2 & I & 0 \end{array}\right]$$

Where

$$F_\infty = -B_2^T X_\infty \tag{2.56}$$
$$L_\infty = -Y_\infty C_2^T \tag{2.57}$$
$$Z_\infty = (I - \gamma^{-2} Y_\infty X_\infty)^{-1} \tag{2.58}$$
$$A_\infty = A + \gamma^{-2} B_1 B_2^T X_\infty + B_2 F_\infty + Z_\infty L_\infty C_2. \tag{2.59}$$

Given $K$, all possible solutions to the controller problem are then created from $K_\infty = LFT(K, Q)$ where $Q$ is a stable LTI matrix where $||Q||_\infty < \gamma$. $Q$ is an additional design parameter that can be used to further tune the behaviour of the controller. The simplest solution satisfying all the criteria is to simply set $Q = 0$ (what system is more stable than one that does nothing?).

With a $Q = 0$ the solution for the controller $K_\infty$ is,

$$K_\infty = -F_\infty(sI - A_\infty)^{-1} Z_\infty L_\infty \tag{2.60}$$

$K_\infty$ is now a controller containing an observer part and a state feedback part. Extruding the observer, it can be seen that is is very similar in structure to the Kalman filter, this can be used to create robust filtering (not covered in this thesis), although it of course has different content and also an added part estimating the worst case disturbance.

$$\dot{\widetilde{x}}(t) = A\widetilde{x}(t) + B_1\gamma^{-2} B_1^T X_\infty \widetilde{x}(t) + B_2 u(t) + Z_\infty L_\infty(C_2\widetilde{x}(t) - y(t)) \tag{2.61}$$

the feedback controller part is given by

$$u(t) = F_\infty \widetilde{x}(t). \tag{2.62}$$

Now that all the conditions for finding and constructing a stabilising controller are given, what remains is to find a suitable algorithm for doing this. Manual

calculations would prove to be extremely time consuming, with the use of a computer the calculations can be done in a couple of seconds.

Since the goal is to find the smallest possible $||N||_\infty < \gamma$ given the above stated criteria, an iterative search over ever decreasing values of $\gamma$ is a rather simple solution. The Bisection search algorithm is commonly used and fairly straightforward, hence it is quickly summarised below.

**Algorithm 1** Bisection search algorithm

1: Give an initial $\gamma$, a step size $d$ and a tolerance $\epsilon$.
2: **if I**,**II** and **III** satisfied **then**
3: Decrease $\gamma$ by $d$.
4: **if** $\gamma - \gamma_{old} \leq \epsilon$ **then**
5: Done.
6: **else**
7: Go to 2.
8: **end if**
9: **else**
10: Increase $\gamma$ by $\frac{d}{2}$
11: Set $d = \frac{d}{2}$
12: Go to 2.
13: **end if**

It is obvious that $(\gamma + \epsilon) > \gamma_{min}$ hence the controller will be suboptimal to a degree depending on *epsilon*. In reality, commercial software such as MATLAB is used to calculate a controller. `hinfsyn()` is an example of a command available in Matlab that given the partitioned plant $P_{tot}$ calculates a sub-optimal controller.

## 2.4 Position and Direction Estimation

All feedback based control are dependant on measurements. In the case of controlling a vehicle, position, velocity, yaw and yaw rate are all quantities of interest. While some of these can be measured directly, such as position by the camera, or the yaw rate by the IMU, there is still a need to merge these measurements and calculate the quantities not directly available. The reliability of the measurements available is also something that needs to be taken into consideration. While numerous different strategies exist, the de facto standard[10] for position estimation is the extended Kalman filter. The reason being the ability to use non-linear models of motion, relatively low complexity and ease of implementation.

### 2.4.1 Extended Kalman Filter

The extended Kalman filter is a variation of the Kalman filter allowing the use of nonlinear models. The underlying process is the linearisation about the current estimate of the mean and covariance. Even though the extended Kalman filter is not the only flavour of Kalman filters able to perform nonlinear state estimation,

is has been considered to be the standard way[10], as long as the transition models used is well defined. It is also the standard when dealing with problems similar to the one at hand such as navigation systems and GPS.[11]

Implementation wise it can be practical to describe the Kalman filter algorithm in two parts, prediction and update. The prediction part updates the state as they would evolve according to the model and the update evaluates the measurements and updates the states with the measurements weighted according how likely they are to be correct. Following a standard formulation of the EKF such as described in [12], with slightly changed notation for readability (with the knowledge that Wikipedia would be a first stop for those not acquainted with Kalman filtering), the algorithm works as follows.

| | |
|---|---|
| $f()$ | the state-transition model |
| $h()$ | the observation model |
| $\boldsymbol{Q}_k$ | the covariance of the process noise |
| $\boldsymbol{R}_k$ | the covariance of the observation noise |

The prediction part estimates the state $\hat{\boldsymbol{x}}$ by propagating the current state through the model $f()$. The covariance matrix $\boldsymbol{P}$ also changes to reflect the increased uncertainty caused by predicting the next state without measurement data.

$$\begin{aligned} \text{Predicted state estimate } \hat{\boldsymbol{x}}_{k|k-1} &= f(\hat{\boldsymbol{x}}_{k-1|k-1}, \boldsymbol{u}_k) \\ \text{Predicted covariance estimate } \boldsymbol{P}_{k|k-1} &= \boldsymbol{F}_k \boldsymbol{P}_{k-1|k-1} \boldsymbol{F}_k^T + \boldsymbol{Q}_k \end{aligned} \tag{2.63}$$

The measurement part compares the actual measurements $\boldsymbol{z}$ with the expected measurement from the model $h()$. This is then used to calculate the Kalman gain $\boldsymbol{K}$ (not optimal since the extended Kalman filter isn't optimal), which is used to update the states and the covariance matrix.

$$\begin{aligned} \text{Innovation or measurement residual } \tilde{\boldsymbol{y}}_k &= \boldsymbol{z} - h(\hat{\boldsymbol{x}}_{k|k-1}) \\ \text{Innovation (or residual) covariance } \boldsymbol{S}_k &= \boldsymbol{H}_k \boldsymbol{P}_{k|k-1} \boldsymbol{H}_k^T + \boldsymbol{R}_k \\ \textit{Near-optimal} \text{ Kalman gain } \boldsymbol{K}_k &= \boldsymbol{P}_{k|k-1} \boldsymbol{H}_k^T \boldsymbol{S}_k^{-1} \\ \text{Updated state estimate } \hat{\boldsymbol{x}}_{k|k} &= \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k \tilde{\boldsymbol{y}}_k \\ \text{Updated covariance estimate } \boldsymbol{P}_{k|k} &= (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k) \boldsymbol{P}_{k|k-1} \\ \text{Where } \boldsymbol{F}_k = \left.\frac{\partial f}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}_{k-1|k-1}, \boldsymbol{u}_k} &\text{ and } \boldsymbol{H}_k = \left.\frac{\partial h}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}_{k|k-1}}. \end{aligned} \tag{2.64}$$

# 3
# Methods

## 3.1 Prototype Platform

One of this project goals is having a functioning, relatively easy to set up prototype platform to see continued use in labs and other control evaluating experiments. The goal is for the software of the platform to be highly modularised. Modularisation allows for each and every part to be designed individually, so long as the communication between parts is standardised, more on this later.

### 3.1.1 Hardware

The prototype platform is based on a number of vehicles as well as a track to run them on. It also includes a vision system and a few micro processors and sensors. All of the control is done from a laptop computer and control signals are then sent to the car. This introduces time delays to the system, but ensures that there is enough processing power for all calculations.

#### 3.1.1.1 Vehicles

To verify the functionality of a designed controller on real R/C cars, cars are needed. For this purpose the platform uses two Kyosho Mini-Z MR-03S[13] RWD RC 1:27 scale RC cars.

#### 3.1.1.2 Micro Processor

To send data from the car to the main computer, a NodeMCU development[14] kit is used. Communication is easily achieved using the built in ESP8266-12E WiFi module. It also supports the use of ROS topics via the `rosserial_arduino`[15] package. WiFi support is however not natively supported by the `rosserial` package and additional libraries[16] have to be installed in order to serve as a translating bridge between the WiFi and the serial communication.

#### 3.1.1.3 Sensors

The vision system consists of a Point Grey FL3-U3-13Y3M-C [17] camera, capable of 150 frames per second at a resolution of 1280x1024 pixels, yielding a very high quality image. This camera is coupled with a lens with a focus length of 4.5 mm and an angle of view of $79° \times 59.4°$ giving a good picture at the intended distance to target ($\sim 2.3m$, height of a standard ceiling).

Onboard each of the cars is also one MPU-6050 IMU used to collect local motion data from the car to be used in later sensor fusion.

#### 3.1.1.4 Controller

To be able to send control signals to the cars, the included controller has been modified so that instead of a user physically controlling the potentiometers on the controller, a micro processor now handles those signals. This is done due to somewhat limited space, and weight restrictions of the cars. Control signals are sent from the main computer to an Arduino Uno[18] prototype board via USB which in turn sends signal to a controller shield that uses a pair of MCP4877 2-channel DACs to control the voltage in place of the potentiometers. A circuit diagram of the two controllers, the two DACs and the arduino is found in Figure 3.1

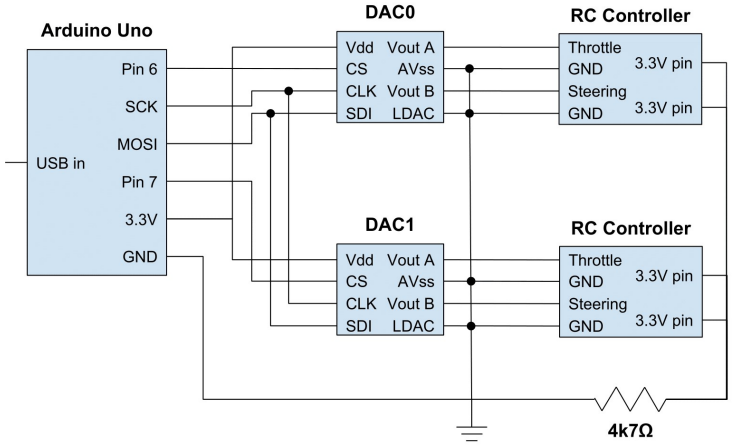



**Figure 3.1:** Circuit diagram of the controllers

#### 3.1.1.5 Tracking LEDs

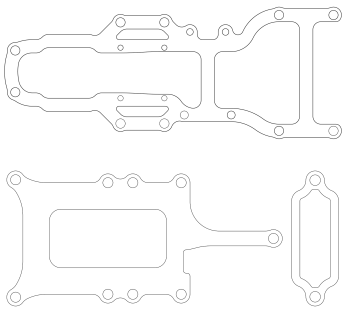
Mounted on top of each of the cars are IR LEDs in very specific arrangements, see figure 3.4. The center LEDs are used as a way of identifying which car that is detected and the outer triangularly placed LEDs are used to track the position and heading of the car.

#### 3.1.1.6 Power

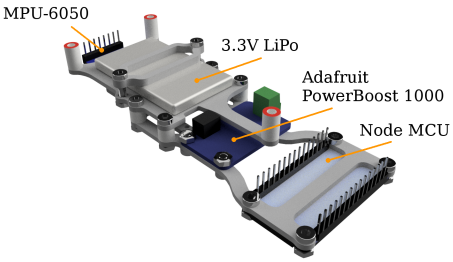
In order to power the onboard sensors and IR LEDs, a battery solution is designed using a single 3.3v LiPi battery and a Adafruit PowerBoost 1000[19]. The PowerBoost 1000 allows for charging of the 3.3V battery during use, which is of great help during testing of the tracking. It also regulates the 3.3V up to 5.2V allowing use with the Node MCU board.

#### 3.1.1.7 Mounting Superstructure for the Kyosho Vehicles

To augment the cars with extra sensors, power supply and Infrared LEDs for the vision system, they need to be mounted on the chassis. To do this physically, such hardware is designed in Autodesk Fusion 360[20] and then manufactured using a laser cutter. The result of this is found in figure 3.2.



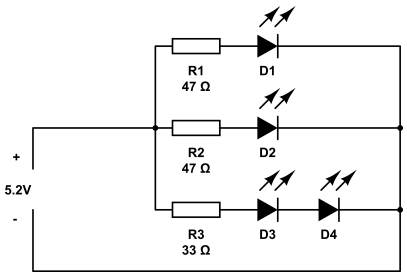
**(a)** Drawings of the three lasercut components.



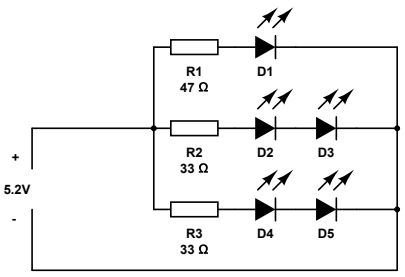

**(b)** Rendered picture of the assembly, including electronics (board for the LED is not pictured, although its mounting points are marked in red.)

**Figure 3.2:** CAD of the superstructure mounted on the cars.

Furthermore, the LEDs require the correct setup to function properly. This is achieved through the use of a circuit board, which consists of a stripboard and a few appropriate components. See figure 3.3.




**(a)** Circuit diagram for car 0




**(b)** Circuit diagram for car 1

**Figure 3.3:** Circuit diagram of the stripboard configuration for the two different cars

### 3.1.2 Software

Different hardware components of the platform require different types of software to function as intended. Some of this software is readily available and some of it has to be written from scratch.

#### 3.1.2.1 ROS - Robot Operating System

Contrary to what the name suggests, ROS is not an operating system, it is a framework for writing software, specifically targeted towards robots. It aims to distribute

tasks in so called nodes. A node is a program, an executable, that uses the ROS framework to send and receive data from other nodes. The node can of course also be written to also perform any other external task, controlling a servo or receiving data from a sensor. The communication is done via Topics, predefined message buses. Unless a user explicitly demands it, nodes don't know who's subscribing to its data or who's sending the data it's publishing to and it doesn't have to. All of that is handled by the master (The main ROS core node). This makes modularisation and distribution of systems very easy.

Since ROS is completely open source, there's often a lot of readily available packages that users are free to use. This often makes it possible to get set up fairly quickly with developing a package for use with a desired project.

#### 3.1.2.2 Tracking Node

The tracking of cars is mainly done visually via the camera. To do this effectively, specific software has to be written. This is handled via a ROS node called `blob tracker` which makes use of the camera video feed and OpenCV libraries to detect and track the IR LEDs mounted on the cars. The code functions as follows:

**Algorithm 2** Track cars

1: Fetch a single frame from the video feed.
2: As a simple filter threshold the image, sorting it into white and black parts.
3: Dilate all white parts (make them larger).
4: Find the contours of all white areas.
5: **if** minArea < Area of contour < maxArea **then**
6: Contour accepted
7: **else**
8: Contour discarded
9: **end if**
10: Find the center of the accepted white areas
11: Create a sub-image of the non-dilated image at the newly found centers.
12: **if** Number of detected contours == 4 **then**
13: Find triangularly spaced dots.
14: Get the 2D pose of the found dots and set car ID to 0.
15: **else if** Number of detected contours == 5 **then**
16: Find triangularly spaced dots.
17: Get the 2D pose of the found dots and set car ID to 1.
18: **else**
19: Do nothing.
20: **end if**
21: Publish each of the 2D poses on their own topics.
22: Go to 1.

Creating the sub images and doing the tracking in them is a way of further trying to make sure that the areas detected actually belong to a car and aren't parts of something else.

To make best use of the camera, this loop needs to run at 150Hz as to not lose any frames and thereby position data of the cars.

**Figure 3.4:** IR LED arrangement for car 0 (left) and car 1 (right)

Local orientation of the car is achieved via the onboard IMU, which is fused together with the camera measurements to form a better tracking estimate than any of the two would achieve by itself.

#### 3.1.2.3 Extended Kalman Filter Node

Considering the assumption that even though the position measurements from the camera tracking are accurate, the update frequency is around 150 Hz limited by the camera. This means that the filters primary function in this case is to estimate the intermediate values using the motion model. In order to get a smoother position, the EKF node is run at 500 Hz. The node can subscribe to the position measured by the tracking node or the IMU data or both. Using these measurements in conjunction with the motion model it estimates position, velocity, acceleration, yaw and yaw rate which it then publishes.

#### 3.1.2.4 Controller Node

**3.1.2.4.1 Steering Control** Having designed the controller in MATLAB it has to be implemented on the car. To realise this, the $\mathcal{H}_\infty$ controller, discretised and in state space form is simply read by the controller code and combined with the measurements of the car to form the next control signal. This is done through the use of a few matrix multiplications,

$$\begin{aligned} x_c(k+1) &= A_c x_c(k) + B_c x(k) \\ u(k+1) &= C_c x_c(k) + D_c x(k). \end{aligned} \tag{3.1}$$

Where $x_c(k)$ are the current internal states of the controller, $x(k)$ are the current measured states of the car, and $A_c$, $B_c$, $C_c$ and $D_c$ are the state space matrices of the controller.

The control signal $u(k+1)$ is then sanity checked (if it's above maximum, set it to maximum vice versa for minimum) and sent to the Arduino connected to the controller hardware.

Similar strategies are used for the LQR and PID controller. The LQR is a simple vector multiplication and the PID uses the yaw to form a yaw error and uses the actual yaw rate form the Kalman filter instead of the forward Euler approximation at each time instance for the differentiating pat.

The connection between the ROS node on the Arduino and the Controller node is done via TCP/IP. It is actually a serial communication running over a USB cable, but the information is packaged as a TCP/IP package. This communication has a maximum baudrate which must not be exceeded. If the rate is exceeded, the connection will start queuing messages (because of the delivery guarantee of the TCP/IP package) potentially causing huge latency problems with the control signals. To combat this, the control node is run fairly slowly at 100Hz.

**3.1.2.4.2 Velocity Control** In the model used for description of the vehicle dynamics, the velocity is assumed constant. This has the effect that the model used to calculate steering control cannot be directly applied to full control of both steering and velocity of the vehicle. Since the velocity needs some form of control a simple PID is implemented and tuned using the Ziegler-Nichols method[21].

While a PID designed this way does have a good tracking performance of reference, this velocity reference has to be decided. A simple strategy is to let the vehicle reach maximum accepted velocity for straight parts of the track and reduce velocity before corners. This is achieved by calculating the angle between the angle of the vehicle and the next two way points of the track.

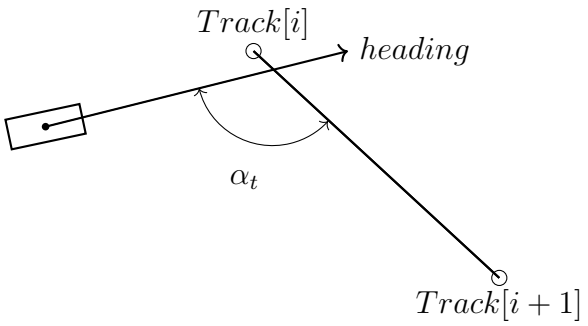



**Figure 3.5:** Angle between vehicle and future way points

As shown by the illustration in figure 3.5, the maximum possible angle is $\pi$ radians. This allows for a normalisation of the angle by dividing by $\pi$ giving ratio with a maximum value of 1 if the heading of the vehicle is identical to the track. Using this normalised value a reference can be shaped for suitable velocity depending on the curvature of the path.

In practice this is made by scaling according to a linear equation on the form $y = kx+m$ where $m$ is the minimum wanted velocity, $k+m$ is the maximum wanted velocity and $x$ is the normalised angle $\alpha_t$. This yields a reference signal $y$ linearly

dependent on the track. Of course, should a constant velocity be desirable, $y$ can be set accordingly.

#### 3.1.2.5 Visualisation Node

As it is useful to be able to keep track of the virtual representation of the car a simple visualisation is implemented. In figure 3.6 presented below, both position and yaw, as well as angle error and reference point can be seen.

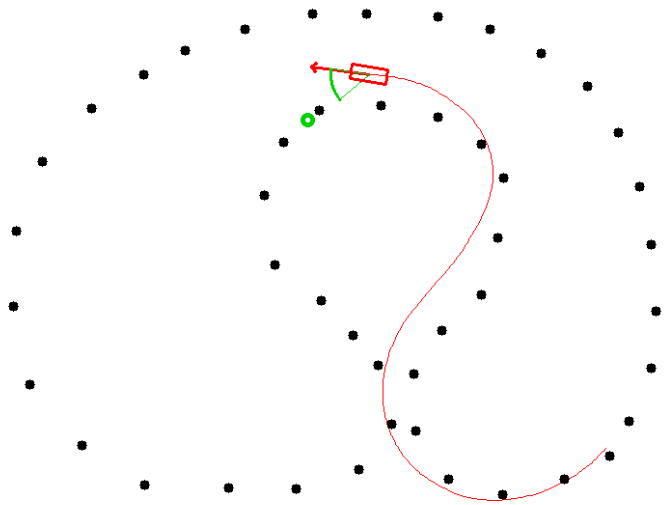


**Figure 3.6:** A visualisation of position, yaw, angle error to track as well as current reference point, track way points and $\approx$ 2 seconds of travelled path

## 3.2 Parameter Analysis

In order for any valid design of robust control to be made, the parametric uncertainties of the system have to be defined. To do this, the uncertain parameters are created using the MATLAB command `ureal()`, where nominal value, range and type of uncertainty is specified. This yields a set of models, which in turn can be used to analyse the impact of errors in the different parameters.

### 3.2.1 Frequency Response Analysis of Error Model

To find the parameters causing the largest change in dynamics, singular value plots of the open loop system, see Figure 2.3, are compared for different uncertain parameters and since the model is linearised around a constant velocity, also different velocities. The nominal parameters are given in table 3.1 below.

**Table 3.1:** Model parameters

| | |
|---|---|
| $m = 0.207$ | Mass of the vehicle. $[kg]$ |
| $\ell_f = 0.0495$ | Distance from centre of mass to front axis. $[m]$ |
| $L = 0.102$ | Wheel base. $[m]$ |
| $\ell_r = L - \ell_f = 0.0525$ | Distance from centre of mass to rear axis. $[m]$ |
| $\ell_w = 0.06$ | Width of the vehicle body $[m]$ |
| $I_z = \frac{m}{12}(L^2 + \ell_w^2) = 0.00024$ | Rotational inertia of vehicle, z-axis. $[kg \cdot m^2]$ |
| $C_{\alpha f} = 20$ | Tire stiffness constant, front wheel. $[N/rad]$ |
| $C_{ratio} = \frac{\ell_f}{\ell_r} = 0.9429$ | Tire stiffness ratio (cornering stiffness is load dependent) |
| $C_{\alpha r} = C_{\alpha f} C_{ratio} = 18.86$ | Tire stiffness constant, rear wheel. $[N/rad]$ |
| $V_x = 1.8$ | Velocity used for linearisation. |

The choice of $V_x$ is done because simulations show that linearising around a higher velocity makes for a model with wider range of suitable velocities. That is to say, a model linearised around a higher velocity has more reasonable dynamics at lower velocities than one that has been linearised at a low velocity and run at a higher.

The parameters that yield a significant cause of concern, or rather those that require either extra attention when designing a robust controller or extra carefulness when measuring them are found using singular values.

#### 3.2.1.1 Vehicle Mass

The mass of the vehicle is measured using a kitchen scale, this is assumed to have a few % error. On top of that, the mass might change over the lifetime of the car. Different batteries weigh differently from other brands, parts might be exchanged etc. Therefore another couple of % is added yielding a total expected deviation of $\pm 5\%$. Such a deviation from nominal mass of the vehicle yields the sigma plot below, showing that the undesirable changes, albeit small, are located in higher frequencies.

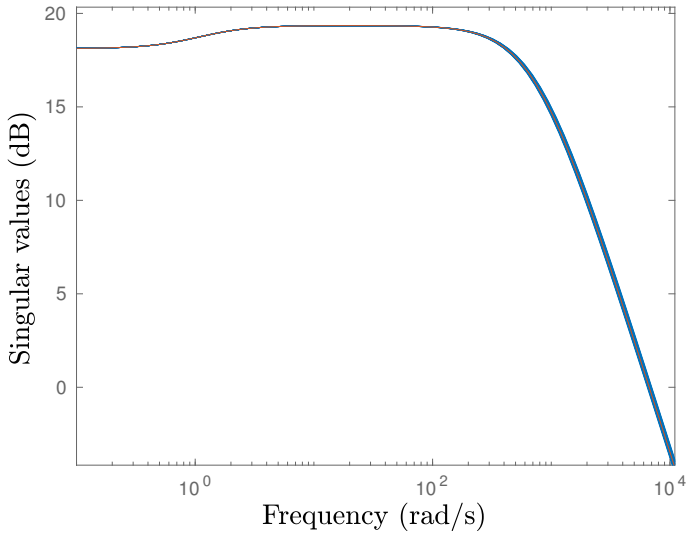



**Figure 3.7:** Sigma plot containing 20 samples of $G$ with uncertain mass. Nominal plant shown in red.

#### 3.2.1.2 Centre of Gravity

Next uncertainty in the system is the distance from centre of gravity (CoG) to the front axle (distance to rear axis being the remainder of $L - \ell_a$, where $L = 0.102m$ is a certain and known value). Centre of gravity is fortunately easy to measure for the stationary vehicle, but the mounting solution of the battery means that it might change slightly between runs. Moreover, the fact the CoG is not at the same height as the wheel axes, the distance from CoG to front axle (and rear) will differ as the vehicle brakes and accelerates, inducing pitch as visualised in figure 3.8. The maximum possible deviation is easily measured yielding a good approximation for the magnitude of the uncertainty.

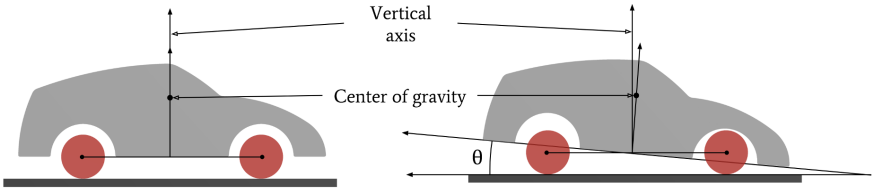



**Figure 3.8:** Visualisation of brake induced pitch motion of the vehicle

The pitch angle $\theta$ is measured to be a maximum of 1.68° and the vertical distance from the longitudinal axis through the wheels to the CoG ($l = 15mm$) (Note that this value too is fairly uncertain and adds to the total uncertainty of the CoG) yielding an expected maximum deviation in the longitudinal direction of

$$CoG_\Delta = 15sin(\theta) = 0.48mm \tag{3.2}$$

This is equal to $\frac{0.48}{\ell_f} = \frac{0.48}{49.5} \sim 1\%$. This is doubled to account for uncertainties

in the vertical position of the CoG yielding a final uncertainty of 2%. Setting this uncertainty yields the results found in figure 3.9.

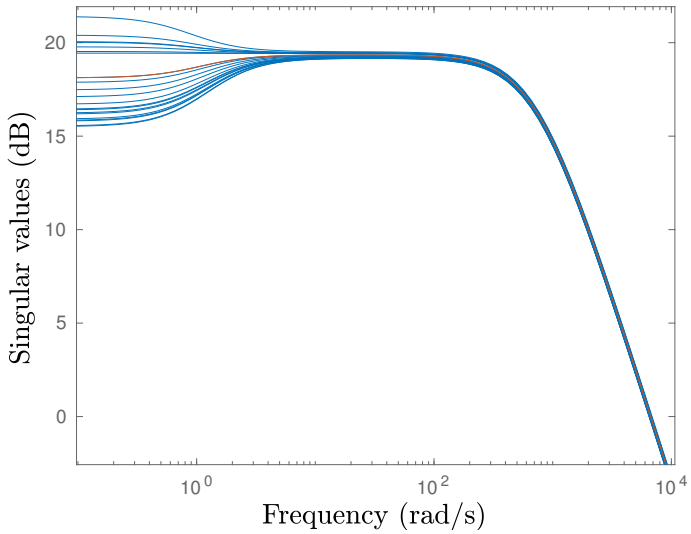



**Figure 3.9:** Sigma plot containing 20 samples of $G$ with uncertain centre of gravity. Nominal plant shown in red.

#### 3.2.1.3 Tire Stiffness

The nominal value for this is taken from an article[22] that measures this value for slightly larger R/C vehicles. Since the rubber compound, surface and not to mention size of the tires differ from the R/C car used, this parameter is highly uncertain. As figure 3.10 verifies, changes in this parameter causes shifts in the dynamics at higher frequencies.

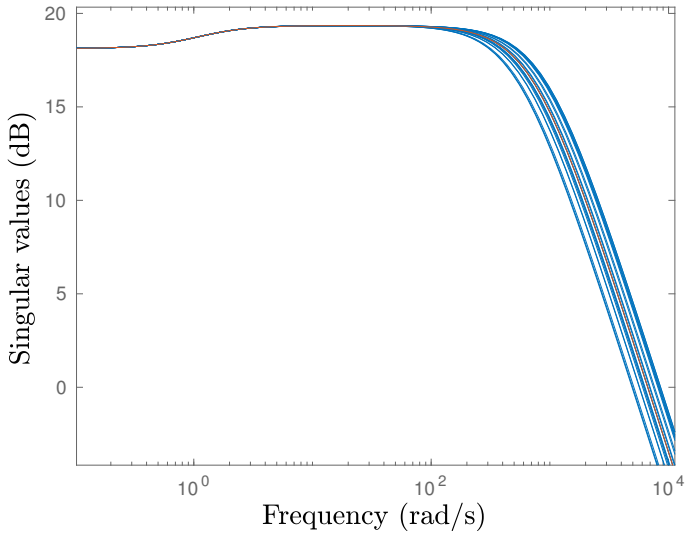



**Figure 3.10:** Sigma plot containing 20 samples of $G$ with uncertain tire stiffness. Nominal plant shown in red.

#### 3.2.1.4 Tire Stiffness Ratio

Since the rear and front tires have differing width, radius, and load, it makes sense that they don't share the same tire stiffness. A ratio is therefore set between them based on the ratio of the load on the front and rear axes, see (3.3) below. To account for difference in radius and width differences an error of $\pm 5\%$ from the nominal value is specified.

$$stiffnessRatio = \frac{\ell_f}{\ell_r} \tag{3.3}$$

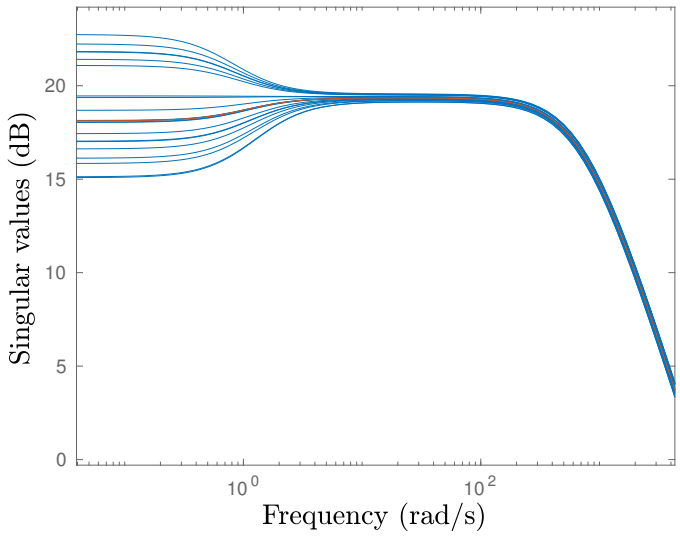



**Figure 3.11:** Sigma plot containing 20 samples of $G$ with uncertain tire stiffness ratio. Nominal plant shown in red.

#### 3.2.1.5 Longitudinal Velocity

What perhaps is most disconcerting is what happens when the car is driven at velocities other than the one used to linearise the model.

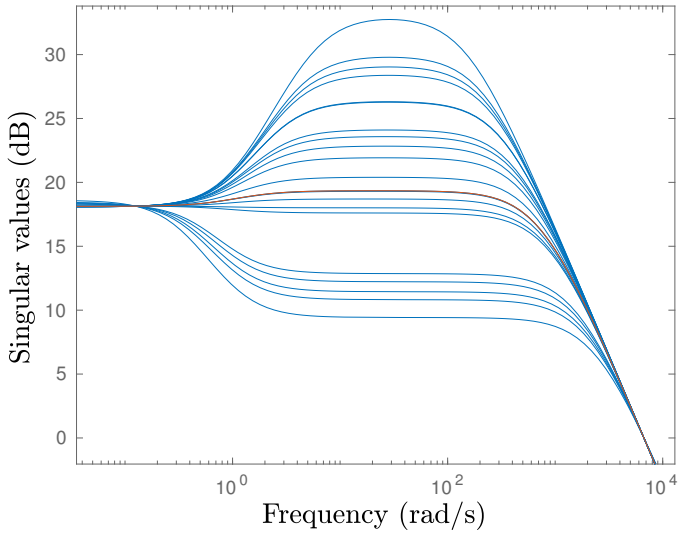



**Figure 3.12:** Sigma plot containing 20 samples of $G$ with uncertain longitudinal velocity used for linearisation. Nominal plant shown in red.

The deviation for this test is defined as a set of $[0.1,\ 4]m/s$. With such a large uncertainty in velocity, it would be difficult to realise a controller that is not far too "*careful*".

#### 3.2.1.6 Total Uncertainty

The total deviation from nominal plant can also be shown if all parameters are uncertain at the same time, which is of course somewhat more realistic. Linearisation velocity is here omitted because setting the velocity as an uncertain parameter could possibly yield a too restrictive controller.

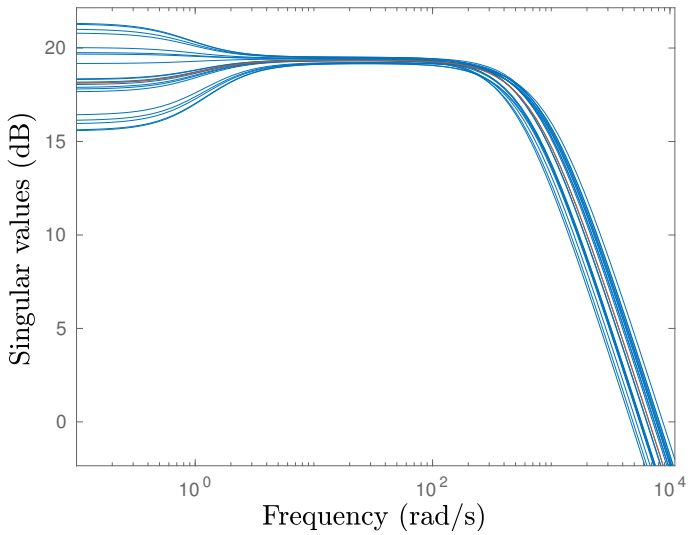



**Figure 3.13:** Sigma plot containing 20 samples of $G$ with all uncertain parameters simultaneously. Nominal plant shown in red.

### 3.2.2 Estimation of Physical Steering Delay

When sending a steering input to the car, it would be naïve to assume that the wheels instantly assume the desired angle. Although it turns out that the change is indeed fast, it is far from instant.

To measure this delay and design a suitable transfer function, the steering is set to it's minimum steering angle of $-25°$ and the steering linkage is filmed with a frame rate of 240 using a smartphone. The steering is then set to the maximum steering angle of 25°, the number of frames required for the wheel angle to switch from $-25°$ to 25° are counted and from that a time is calculated. It is found that the wheels require an average of 8 frames to reach it's desired angle, yielding a $\frac{8}{240} = 1/30 \approx 33ms$ delay. Adding a simple time delay would be the easy way out, but what does not capture the dynamics very well. Instead a simple first order transfer function is designed and added as an input dynamic to the system. While the correctness of this transfer function is difficult to verify due to the lack of measurements, it is assumed to be as close as possible with the information available.

A transfer function on the form,

$$\frac{1}{s/100 + 1}, \tag{3.4}$$

yields the step response found in Figure 3.14 below.

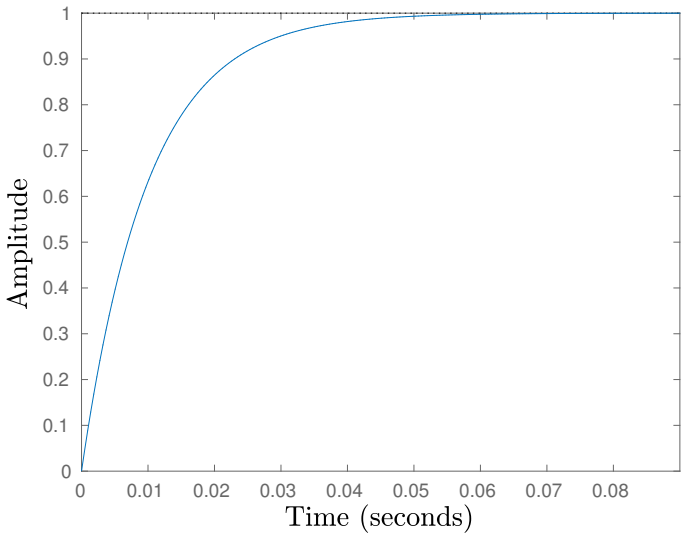



**Figure 3.14:** Step response of the designed steer delay transfer function

## 3.3 Controller Synthesis

Using what is described in section 2, four types of controllers for direction control are designed in order to evaluate performance of the platform. Due to lack of the longitudinal dynamics of the car, described by the model used for controller design, the velocity control of the car is handled by a PID controller separate from the steering. This velocity control implementation is the same regardless of direction controller implementation.

### 3.3.1 Directional Controlling PID

The PID controller is a simple non-filtered one, designed according to the Ziegler-Nichols method[21] and a bit of manual tweaking. It also implements a very simple anti-windup. Such a controller results in the following parameters:

**Table 3.2:** PID controller parameters

| | |
|---|---|
| $K_p$ | 1900 |
| $K_i$ | 500 |
| $K_d$ | 150 |

The PID is simply an output feedback controller and is not based off of a model description, hence designing it towards an uncertain plant is no different than doing it towards a nominal one. Due to the specific software implementaion the PID control signal output differs from the other controller by not being in degrees. Instead $u \in (-1600\ 1600)$ which corresponds to the voltage range in millivolt of the hardware controller described in section 3.1.1.4.

### 3.3.2 Linear Quadratic Controller

The first form of robust controller is the LQR. The parameters for this controller is the expected gaussian process noise and the input cost, since the longitudinal velocity is not included in the state, the controller is only designed to handle the steering and will have to do so under varying velocity conditions.

$$Q = \begin{bmatrix} 130 & 0 \\ 0 & 100 \end{bmatrix}, R = 6700 \tag{3.5}$$

signifying an expensive control strategy, yielding a state feedback vector on the form

$$K_{LQR} = \begin{bmatrix} 0.0622 & 0.0549 \end{bmatrix} \tag{3.6}$$

The LQR controller is based off of a model, but it cannot take the uncertain parameters into consideration at the same level as the $\mathcal{H}_\infty$-synthesis, thus, it's robustness is limited. In fact, the robustness of the LQR is only described by adequate phase and gain margins.

### 3.3.3 1DoF $\mathcal{H}_\infty$ Output Feedback Controller

Designing a controller often gets more difficult the more demands on performance you include, knowing that, the first controller is a pure disturbance rejecting one. Since the controller is decoupled from the longitudinal dynamics, the controller can not account for it in a robust way, but as with the LQR, the heading and turn rate of the vehicle are robustly controlled.

The weights used for the 1DoF design are listed and explained below

**Table 3.3:** System weights for the 1DoF $\mathcal{H}_\infty$-control design

| | |
|---|---|
| $W_1$ | Uncertainty shaping |
| $W_2$ | Uncertainty penalisation |
| $W_u$ | Performance weight, penalises the control signal use |
| $W_d$ | Disturbance shaping |
| $W_{perf}$ | Performance weight, penalises the error in $\dot{\Psi}$ |
| $W_n$ | Noise shaping |

The weights described in Table 3.3 are designed to achieve this as follows:

Following the analysis in 3.2, $W_1$ and $W_2$ describe the parametric uncertainty and are created using the MATLAB function `ucover()`[23] which fits an uncertain system to a set of LTI responses. The function also outputs the multiplicative uncertainty as weights, $W_1$ and $W_2$. The structure of the `ucover()` output sets $W_2 = 1$ and $W_1$ is shaped according to the relative error from the nominal model caused by the parametric errors.

Figure 3.15 below shows the result of the `ucover()`, the maximum relative difference from nominal behaviour over frequencies.

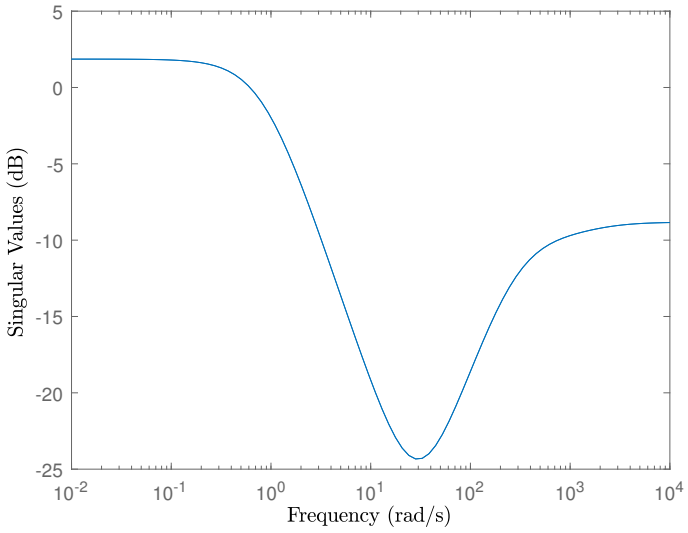



**Figure 3.15:** Relative errors of uncertain plant, $W_1$

$W_u$ penalises the use of control input. This weight is set fairly low to allow for more control signal. Frequency wise it is set according to figure 3.16 to ensure that jittery behaviour is limited by penalising higher frequencies.

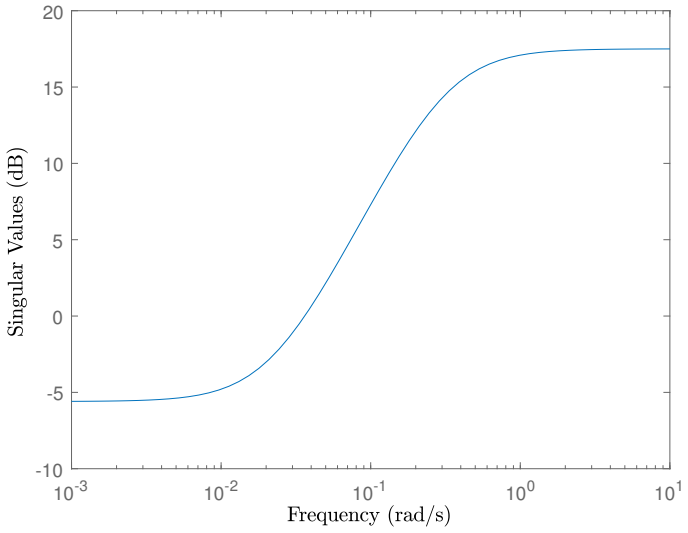



**Figure 3.16:** Performance weight for the control signal, $W_u$

The weight assigned to the disturbance $W_d$ should resemble the actual disturbance as close as possible. This can be achieved by collecting data from simulations runs. As the yaw rate $\dot{\psi}$ can be calculated as $\frac{Velocity}{Radius}$, the desired yaw rate is easily obtainable from the current velocity of the vehicle and the current calculated radius of the track. The resulting desired yaw rates can then be analysed with for example a fast Fourier transform. This analysis is however tricky in the sense that the

frequency and magnitude content varies with different controllers. Given this slight catch 22, empirical results proved that a Gaussian noise with a magnitude of 1.9 gave the best performance.

$W_{perf} = W_{p1}$ is a weight that penalises the state, in this case the errors. It's designed to keep performance high up until a bit below the bandwidth of the input (the control loop is run at $100Hz \approx 628rad/s$), but is fairly relaxed to keep control use down (the weights influence each other). See figure 3.17.

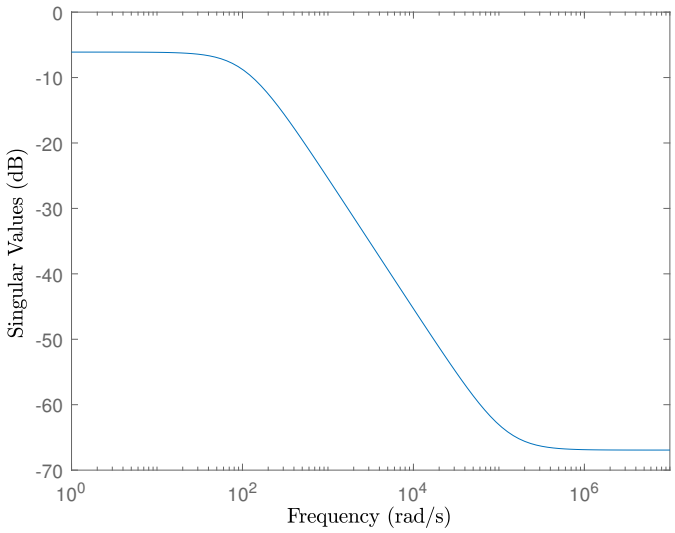



**Figure 3.17:** Performance weight for the error $\dot{\Psi}$, $W_{p1}$

$W_n$ is just as expected a description of the noise that enters the measurements. It is designed as a static matrix with the expected co-variance of the measurement noise. These values are easily measured.

$$\begin{bmatrix} 0.02 & 0 \\ 0 & 0.5 \end{bmatrix} \tag{3.7}$$

The weights are then used to build a system with the structure shown in figure 3.18,

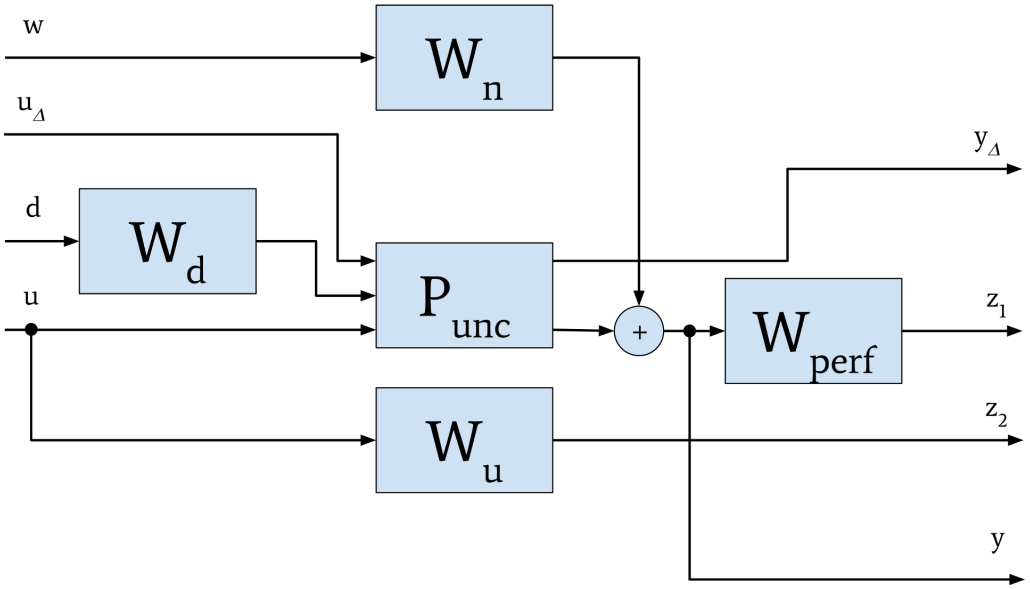



**Figure 3.18:** $P_{real}$ with added design weights for 1DoF $\mathcal{H}_\infty$ design

The goal is then to assure

$$||T_{dz}||_\infty = \sup \frac{||z||_2}{||d||_2} < \gamma < 1. \tag{3.8}$$

In this case, since the controller is purely disturbance rejecting, the reference signal is seen as a disturbance. This means that what actually is assured is

$$||T_{rz}||_\infty = \sup \frac{||z||_2}{||r||_2} < \gamma < 1. \tag{3.9}$$

Since the performance state $z$ in this case is the errors, what is minimised is the reference signals impact onto the errors.

The number of states in the final controller is directly proportional to the number of signals, the number of weights and the number of outputs. The 1DoF $\mathcal{H}_\infty$ controller ends up being 10 states. The same number of states as in $P_{tot}$.

### 3.3.4 2DoF $\mathcal{H}_\infty$ State Feedback Controller

In the case of the purely disturbance rejecting 1DoF controller, even the reference signal is considered a disturbance signal. Although this might yield satisfactory results, it is possible to further augment the controller with reference input as well. This results in a so called 2DoF controller. It has knowledge about both the system and the reference signal.

Designing a plant for the 2DoF system is very similar to the 1DoF, but the reference signal is now introduced to the system in a different way. Instead of the reference signal being treated as a disturbance, it is being treated in the more common way of comparing it to the output of the plant, weighted and treated as a performance metric.

$$z(t) = e_z(t)W_{perf} = (u(t)P_{uc} - r(t))\,W_{perf} \tag{3.10}$$

The weights used for the 2DoF design are similar in design, but now weighs other metrics.

**Table 3.4:** System weights

| | |
|---|---|
| $W_1$ | Uncertainty shaping. |
| $W_2$ | Uncertainty penalisation. |
| $W_u$ | Control weight, penalises the control signal use. |
| $W_d$ | Disturbance shaping. |
| $W_{ideal}$ | Reference behaviour shaping. |
| $W_{p1}$ | Performance weight, penalises the difference between $\Psi_r$ and $\Psi$ |
| $W_{p2}$ | Performance weight, penalises the yaw rate $\dot{\Psi}$ |
| $W_n$ | Noise shaping |

Again, the $W_1$ and $W_2$ are related to the actual plant and therefore unchanged.

$W_u$ still penalises the use of control input. But it has a slightly different weighting than in the 1DoF case, see figure 3.19, to counteract the effects of the other added weights.

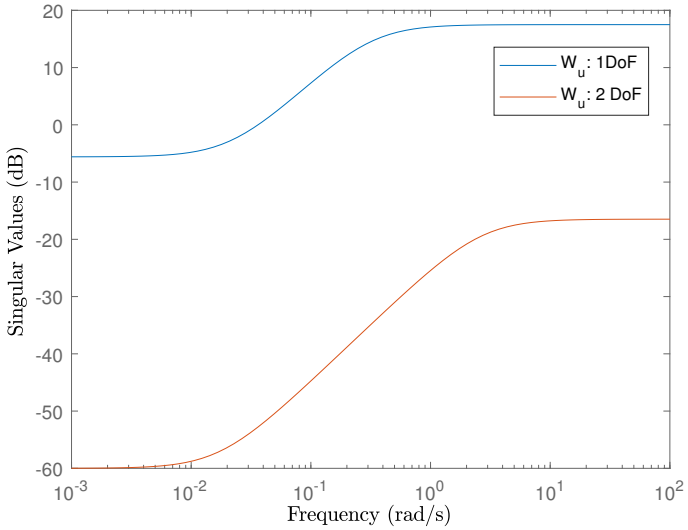



**Figure 3.19:** Difference in control signal performance weight between the 1DoF and 2DoF design

$W_d$ remains identical to the 1DoF design for the same, the input from the track doesn't differ.

$W_{p1}$ still penalises errors in $\dot{\Psi}$. See figure 3.17.

$W_{p2}$ penalises the difference between the state behaviour and the desired behaviour reference, $W_{ideal}$, where the reference behaviour is on the yaw, $\Psi$. Note that since the output of $P_{real}$ are the errors, the reference in yaw is always 0, See figure

3.20 to see the frequency design of $W_{p2}$. This shape denotes that it is relatively expensive to have low frequency errors, meaning we force a low frequency behaviour of the reference tracking.

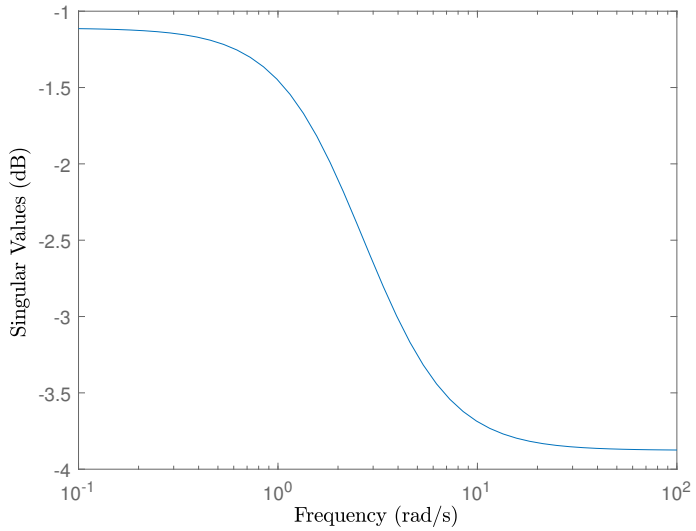



**Figure 3.20:** Performance weight for the deviation from reference signal

$W_{ideal}$ represents the desired plant response to a reference. The fact that for this case the reference signal is constantly 0 makes this part some what non-intuitive. But if seen as a behaviour shaper it should dawn, that while having an instant tracking of reference would be great, it's not realistic and might cause the controller to have very erratic behaviour. Because of this and the fact that the reference is really on the desired error, the ideal behaviour is described as a first order lead compensator.

$$W_{ideal} = \frac{s + 0.00045}{s + 0.045}$$

The step response of in figure 3.21 this shows that error is allowed to begin with, but should quickly decline towards 0.

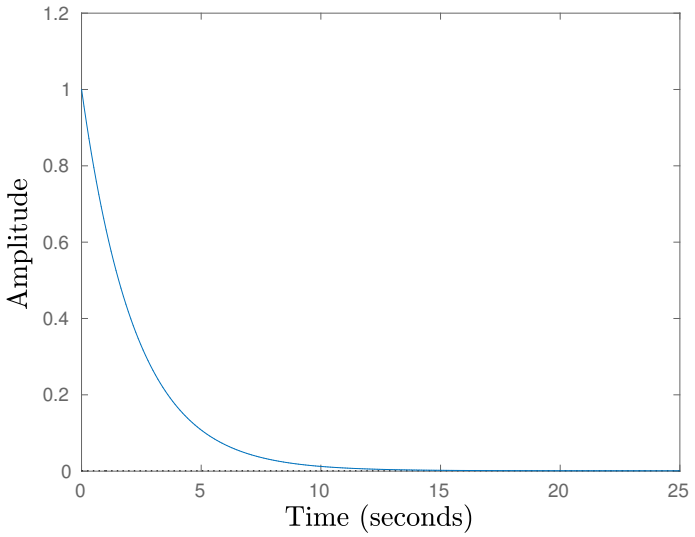



**Figure 3.21:** Step response of the desired reference behaviour

Using these weights, the following structure is designed, see figure 3.22.

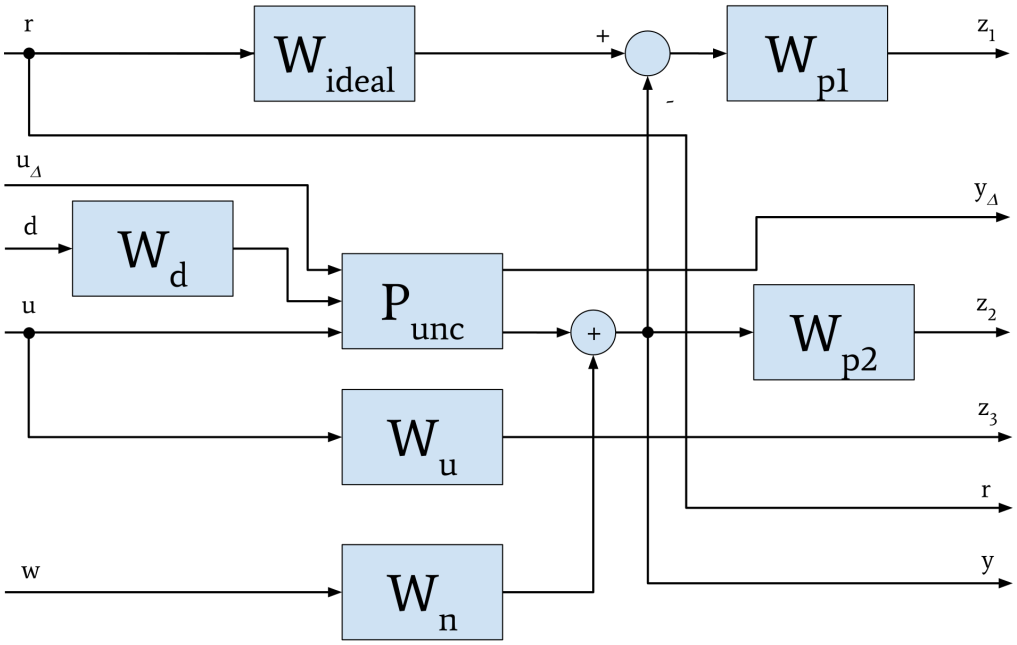



**Figure 3.22:** $P_{real}$ with added design weights for 2DoF $\mathcal{H}_\infty$ design

What makes the 2DoF controller a 2DoF is that it not only receives information about the plant output, it also is fed with the reference signal (which can be weighted if needed) giving it more information about the system and thus should yield higher performance.

The 2DoF $\mathcal{H}_\infty$ controller ends up being 11 states. Again, the same number of states as in $P_{tot}$

### 3.3.5 Velocity Control PID

The PID used to control velocity is a simple non-filtered one, manually tuned. A simple anti-windup is again implemented. Since there is no feedback on the actual rotation of the wheels a rate limiter is added to the output to decrease the risk of 'burn out's. To ensure that the rate limit doesn't interfere with the integrating part of the controller, or perhaps the other way around, the integral error is limited to $0.1m/s$. To be able to brake more freely, the integrating error is not increasing if we are braking. The controller parameters are as follows,

**Table 3.5:** Velocity PID controller parameters

| | |
|---|---|
| $K_p$ | 560 |
| $K_i$ | 150 |
| $K_d$ | 63 |

and the rate limiter is set to ensure that the throttle can't increase more than 4‰ of maximum per time sample.

The output of the velocity controller relates to a millivolt scale that corresponds to the throttle controls input which span from 0 to 3200.

## 3.4 Extended Kalman Filter Implementation

### 3.4.1 Coordinated Turn Motion Model

To describe the movement of the vehicle a model that can capture the motion behaviour is needed. While simple models such as constant acceleration usually can describe most motion, the limited turn radius and the behaviours this give rise to, encourages the use of the motion model known as coordinated turn[24]. It is suitable in describing the motion as it describes position changes as a consequence of the velocity and the changes in bearing of the vehicle. The model, as it is usually described consists of five states $p_x, p_y, v_x, \psi, \dot{\psi}$, but since the vehicle is equipped with an accelerometer within the IMU, the acceleration in the local x-axis coordinate frame, $a_x$, is added providing a possible simplification of the measurement model seen later. Since the heading of a vehicle isn't the same as the angle of the body, the vehicle slip angle $\beta$ is added. It is calculated using the current angle of the wheels relative to the body combined with some physical measurement of the distance of the wheel axles to the centre of gravity. However, the kinetic model used for controller calculation does not make use of the heading of the vehicle $(\psi + \beta)$[5] but the yaw angle of the vehicle body, $\psi$, as seem in section 2.2.2.

Using the assumption that a known control input of wheel angle will have a known effect, the $\beta$ is considered to be known and therefore isn't estimated by the Kalman filter but rather added directly from the output of the controller.

$$\beta = \tan^{-1}\left(\frac{\ell_r \tan\delta_f + \ell_f \tan\delta_r}{\ell_r + \ell_f}\right) \tag{3.11}$$

$$
\begin{aligned}
p_x[k+1] &= p_x[k] + v_x[k] \cdot \cos\left(\psi[k] + \beta\right) \cdot dt \\
p_y[k+1] &= p_y[k] + v_x[k] \cdot \sin\left(\psi[k] + \beta\right) \cdot dt \\
v_x[k+1] &= v_x[k] + a_x[k] \cdot dt \\
a_x[k+1] &= a_x[k] + q_{a_x}[k] \\
\psi[k+1] &= \psi[k] + \dot{\psi}[k] \cdot dt \\
\dot{\psi}[k+1] &= \dot{\psi}[k] + q_{\dot{\psi}}[k]
\end{aligned}
\tag{3.12}
$$

Jacobian

$$
\begin{bmatrix}
1 & 0 & \cos\left(\psi[k] + \beta\right) \cdot dt & 0 & -v_x[k] \cdot \sin\left(\psi[k] + \beta\right) \cdot dt & 0 \\
0 & 1 & \sin\left(\psi[k] + \beta\right) \cdot dt & 0 & v_x[k] \cdot \cos\left(\psi[k] + \beta\right) \cdot dt & 0 \\
0 & 0 & 1 & dt & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & dt \\
0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\tag{3.13}
$$

### 3.4.2 Camera Measurement Model

The camera measurements of position and heading are states that is directly represented in the state space used by the motion model. This means that these states can be updated directly from measurements. Since the camera is also considered to have a high degree of accuracy, the associated covariance values describing the noise is set to very small values. Calculating the resolution of the camera, each pixel sees $3 \times 3$ mm providing a lower limit. The distance between the IR LED combined with the resolution in each axis of 3 mm gives a lower limit to yaw resolution of 0.055 radians.

$$
\begin{aligned}
h(\hat{\boldsymbol{x}}_{k|k-1}) &= \begin{bmatrix} p_x \\ p_y \\ \psi \end{bmatrix} \\
\boldsymbol{z} &= \begin{bmatrix} p_{x_{cam}} \\ p_{y_{cam}} \\ \psi_{cam} \end{bmatrix} \\
\boldsymbol{H}_k = \left.\frac{\partial h}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}_{k|k-1}} &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}
\end{aligned}
\tag{3.14}
$$

### 3.4.3 Accelerometer Measurement Model

The IMU is capable of measuring acceleration in the local x-,y- and z-axis coordinate frame. It is also capable of measuring the velocity of rotations around these axis. With the simplification of considering the vehicle as moving in a two dimensional plane with no local *sideway* movement, only the x-axis acceleration and the yaw rate around the z-axis needs to be measured.

$$
\begin{aligned}
h(\hat{\boldsymbol{x}}_{k|k-1}) &= \begin{bmatrix} a_x \\ \dot{\psi} \end{bmatrix} \\
\boldsymbol{z} &= \begin{bmatrix} a_{x_{IMU}} \\ \dot{\psi}_{IMU} \end{bmatrix} \\
\boldsymbol{H}_k = \left.\frac{\partial h}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}_{k|k-1}} &= \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
\tag{3.15}
$$

### 3.4.4 Desired Outcome of the Kalman Filter

The desired outcome of the Kalman filter implementation is of course a noise free and accurate estimation of position, velocity and yaw angle. In this case the constructed filter is also designed to provide a higher sample rate than possible from the camera, upsampling from the 150 Hz camera measurements to 500 Hz. The control loop is run at only 100 Hz, but the sync of the measurements and the control loop cannot be guaranteed, hence the need for higher resolution in the positional data.

Since the filter is dependent on design parameters, the filter needs to be decided and evaluated to guarantee performance. The evaluation of the filter is made by comparing the Kalman filter output with the camera data. Eyeballing the data should be enough to evaluate both noise reduction and time delays in the filtered data compared to the camera measurements, making sure the filter output both agrees with the camera position as well s being able to predict and fill in the *missing* positions in between measurements. The indirect measurements such as velocity are evaluated with regard to the physical limitation of velocity changes decided by the vehicle itself. In practice this means tweaking the parameters until the position agrees with the camera measurements, the intermediate positions *look* reasonable and that the velocity calculated by the filter is both smooth and reasonable in magnitude.

## 3.5 Track Description

The track that the vehicle follows is described using a ordered list of position coordinates. To ease the construction of the track a MATLAB script is used where mouse pointer input is translated to a track that can be used both in simulation as well as loaded to the actual control system. There are no real constraints in the creation of a track, however, physical limitations such as the turning radius of the vehicle will cause some tracks to be impossible for the vehicle to successfully follow.

### 3.5.1 Reference Generation from Track

As the track is described using positions defined as $X$- and $Y$-coordinates, and the controller relies on errors in yaw and yaw rate of the vehicle, the associated errors are calculated. The difference in the yaw of the vehicle compared to where it should be headed, is calculated using a vector description of vehicle yaw and a line through the centre of gravity of the vehicle and the next reference point.

The error in yaw rate is calculated through the description of a circle passing through the vehicle centre of gravity, a point slightly in front of the vehicle and the reference point. Using this circle and the current velocity the desired yaw rate is obtained.

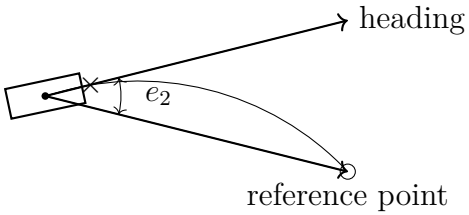



**Figure 3.23:** Yaw error describing the difference in heading and angle to reference point.

As the controller performance is dependant on the frequency behaviour of the disturbance, a simple algorithm to handle reference points in relation to the track is implemented. This lets the reference point move ahead of the vehicle in a predictable fashion, sliding between track points at a distance dependent on the vehicle velocity.

## 3.6 Restrictions and Reductions

### 3.6.1 Model

The model used for control design is the simplified version described in section 2.2.2 in equation (2.31) where the only two states are the yaw and yaw rate.

### 3.6.2 Comparison between Simulation and Reality

As much of the collected data comes from simulation runs, a comparison with real world performance gives valuable insight on how much the simulation runs can actually tell us. By iterative runs it also enables the fine-tuning of parameters that are hard to estimate or measure. However, as no model, especially a linearised one as the one used, can be expected to represent the real system the discrepancies between the simulations and reality are expected.

### 3.6.3 Exclusion of IMU Measurements

Due to latency issues in the communication protocol used to send the IMU data, the IMU remains unused throughout all testing. This turns out to not cause any issues regarding position estimation, but one would assume that such measurements would be even more exact with more data available.

### 3.6.4 Number of Cars

Since the verification of robustness and performance in terms of tracking isn't dependent on the number of cars, the decision was made to focus the work on a single

car. If a racing aspect of it is to be considered, that is still possible simply by comparing lap times between controllers assuming that the track is less forgiving (if the controller has too poor tracking abilities it shouldn't be able to complete a lap because it would be off the track).

## 3.7 Performance Evaluation

### 3.7.1 Parameter Variation

As the controllers are supposed to handle uncertain parameters to a varying degree, changing the physical parameters of the vehicle is the obvious way to evaluate the robustness in this aspect.

Since single values (not to be confused with a *singular* value) don't tell much about the system behaviour, the trajectories of the vehicles are also presented for each case.

#### 3.7.1.1 Velocity Variation

As the model uses a constant velocity and analysis show that any deviations from this value have a high impact on the system, it is suitable to evaluate performance with different reference velocities. For this, three different maximum desired velocities for the PID speed controller are used; $1.5m/s$, $2.0m/s$ and $2.5m/s$. This yields a more 'racing' aspect to the test where the car brakes in corners and accelerates on the straights.

Also tested is the system response to a constant velocity, such that the tracking performance is not impacted by the velocity control to the same extent. For this test, the reference velocities are $1.0m/s$, $1.5m/s$ and $1.7m/s$. The choice of the highest value at $1.7m/s$ becomes very apparent later.

#### 3.7.1.2 Mass and CoG Variation

For the real experiments, a mass is added to the rear of the frame increasing the total mass to $.222kg$ (7.4% increase from Nominal) and shifting the CoG backwards by $2mm$ (4% of maximum). Note that this shift in CoG and mass is 4 and 1.5 times above the estimated difference respectively, so the uncertainty levels are technically out of range of what the controller is designed for. We however do know from figure 3.15 that the relative errors of the system remains fairly low, so an increase above this should not cause too many issues (although the robust performance can of course no longer be guaranteed). Since the mass and position of CoG influence all the parameters, the parameters for this case are updated as follows in table 3.6

Simulation gives a bit more freedom in the ability to change parameters, so a worst case parameter scenario is also tested using the same constant velocities as for the experiments.

**Table 3.6:** Parameters used for the real world uncertainties

| | |
|---|---|
| $m = 0.22$ | Mass of the vehicle. $[kg]$ |
| $\ell_f = 0.0515$ | Distance from centre of mass to front axis. $[m]$ |
| $L = 0.102$ | Wheel base. $[m]$ |
| $\ell_r = L - \ell_f = 0.055$ | Distance from centre of mass to rear axis. $[m]$ |
| $\ell_w = 0.06$ | Width of the vehicle body $[m]$ |
| $I_z = \frac{m}{12}(L^2 + \ell_w^2) = 0.00027$ | Rotational inertia of vehicle, z-axis. $[kg \cdot m^2]$ |
| $C_{\alpha f} = 20$ | Tire stiffness constant, front wheel. $[N/rad]$ |
| $C_{ratio} = \frac{\ell_f}{\ell_r} = 1.019$ | Tire stiffness ratio (cornering stiffness is load dependent) |
| $C_{\alpha r} = C_{\alpha f} C_{ratio} = 20.4$ | Tire stiffness constant, rear wheel. $[N/rad]$ |

### 3.7.2 Test Setup

Data is collected by saving all the available information related to position, velocity and yaw. That is, the output of the Kalman filter is sampled at its output frequency of 500 Hz. The output of the controller is also collected consisting of steering and throttle signal, as well as which track indices that are positioned behind and in front of the vehicle. These track indices can then be used to calculate the vehicle distance to the track, $\mathbf{E_d}[i]$, for each collected sample $i$. $\mathbf{E_d}$ is then analysed to further understand the behaviour of different controllers during the different scenarios. It should be noted that the indices sampled might in cases of index update be misaligned, causing a slight amount of outliers in the data.

The performance is evaluated by using the following metrics.

**Table 3.7:** The different metrics collected from the data.

| | |
|---|---|
| $\mathbf{M(E_d)}$ | Mean distance from track. $[m]$ |
| $\mathbf{M(V_x)}$ | Mean velocity $[m/s]$ |
| $\mathbf{Mode(E_d)}$ | Mode of the lateral error (Most common value) $[m]$ |
| $\mathbf{M(E_d)/M(V_x)}$ | Mean distance from track divided by mean velocity $[s]$ |
| $\mathbf{M(T)}$ | Mean time to complete one lap of the track $[s]$ |
| $\mathbf{D}$ | Total distance travelled by the vehicle $[m]$ |

**Table 3.8:** The test setup

| | | | | |
|---|---|---|---|---|
| Number of laps: | 4 | | | |
| Total track length (for 4 laps): | 45.0817 m | | | |
| Velocity | | | | |
| $V_{max}$: | 1.5 $m/s$ | 2.0 $m/s$ | 2,5 $m/s$ | |
| $V_{constant}$ : | 1.0 $m/s$ | 1.5 $m/s$ | 1.7 $m/s$ | |
| Controllers: | PID | LQR | 1 DoF | 2 DoF |
| System characteristic: | Nominal | Added fixed weight | | |

The track setup remains identical for each of the simulated responses as well as for the experiments. As for the initial states of the car, they too remain near

identical for each of the runs. The car is placed at a fix position and orientation on the track and always starts from 0 velocity. In an effort to keep the friction coefficient similar during testing, the track is cleaned using a brush and any dust on the tires of the car is wiped off. Although these things would count toward uncertainties in the system, quantifying them and designing for them would require the possibility of measuring the differing tire forces. This is a measurement unavailable to us.

Noticing that the battery levels have an impact on the acceleration of the car, efforts where made to change the batteries often and also to change the order of tested controller as to spread the variation over multiple controllers, not giving any a clear advantage.

# 4
# Results

## 4.1 Position Estimation Verification

### 4.1.1 Test and Evaluation of Kalman filter

Sampling data from both the camera and the Kalman filter, a rough performance estimation may easily be made. Since the Kalman filter is well known, the process of parameter tuning is omitted. The filter was is this case tuned to allow for unexpected movement such as sideways skidding not possible in the motion model by increasing *trust* in the camera measurements. This leads to a small amount of noise remaining in the measurements even after filtering. As illustrated by figure 4.1 the filter estimates closely track the camera measurement but manages to provide intermediate values by up sampling as well as reducing the level of noise to reasonable levels.

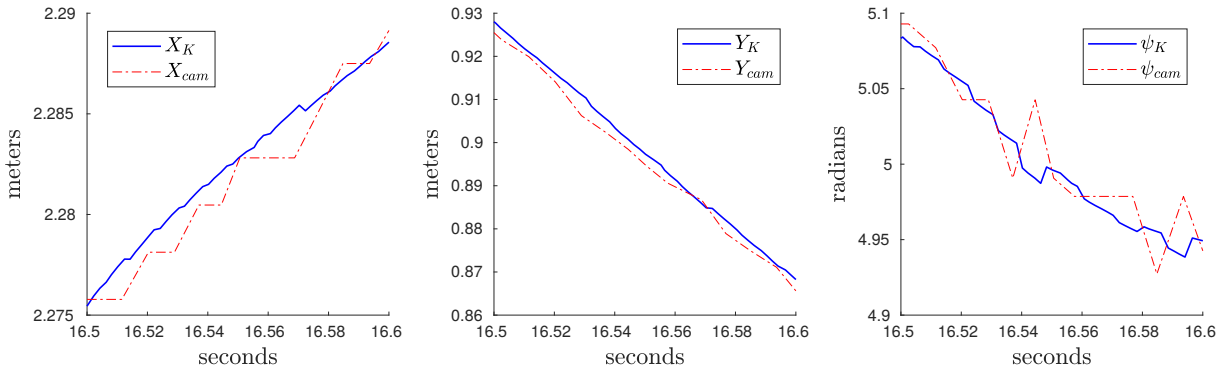
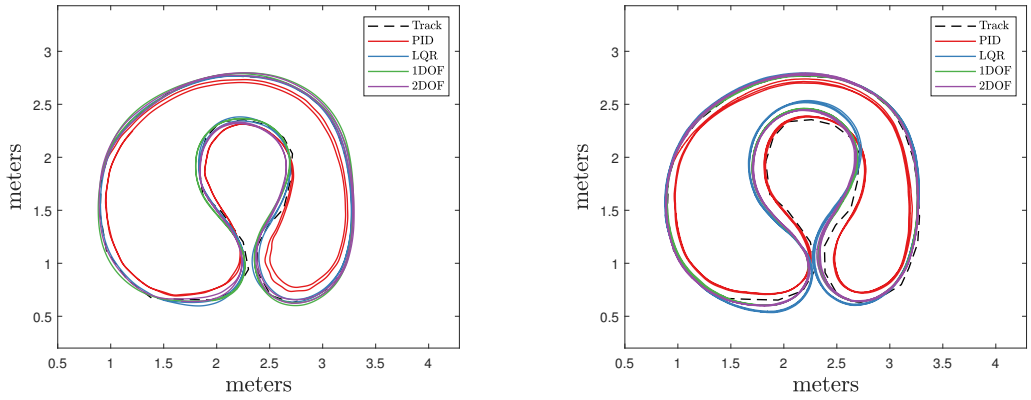



**Figure 4.1:** Estimated states from the Kalman filter together with camera measurements

## 4.2 Model Verification

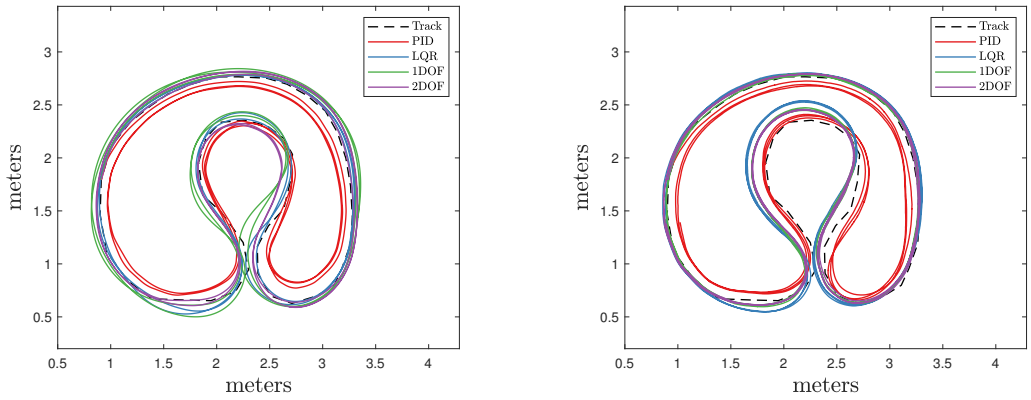
To verify that the used nominal model is sufficient to describe the real world dynamics, simulated results are compared with nominal real world experiments. To keep the simulations tests closer to reality, they use the same velocities as the real experiments (velocities at each given time point are stored). One key difference between real dynamics and that of the model is that the model can't skid, it always has infinite grip. Knowing this, the high speed performance of the model might

surpass that of the real experiment. To avoid unnecessary clutter in the plots, all simulations are only the first 20 seconds.



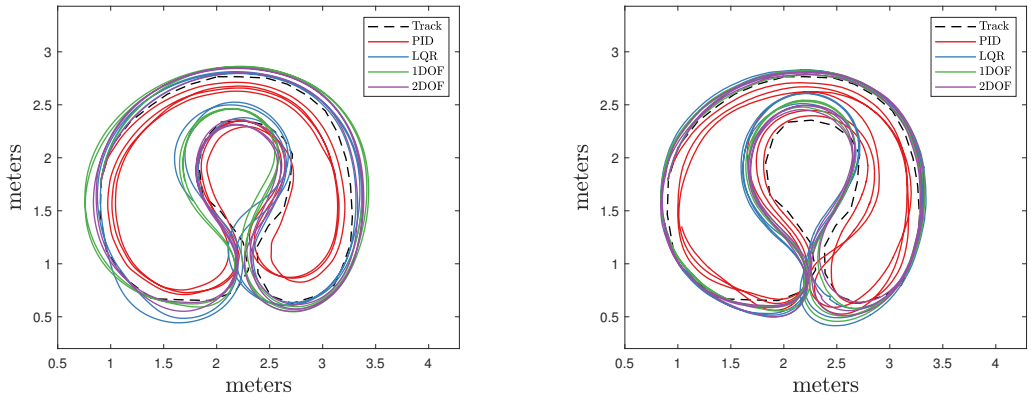

**(a)** Nominal simulation **(b)** Nominal experiment

**Figure 4.2:** The 4 different controllers in simulation and real experiment for 1.5 m/s desired velocity




**(a)** Nominal simulation **(b)** Nominal experiment

**Figure 4.3:** The 4 different controllers in simulation and real experiment for 2.0 m/s desired velocity




**(a)** Nominal simulation **(b)** Nominal experiment

**Figure 4.4:** The 4 different controllers in simulation and real experiment for 2.5 m/s desired velocity

Although the tracking is not identical, the general behaviour is recognisable from the simulated response. Interesting to note is that the real experiments are a lot more consistent for each lap. The car is let run for four laps, and often the car drives *wheel-in-wheel*. The differences in jittery behaviour could emerge due to the fact that very fast changes in direction are not possible due to lack of grip at the front wheels, larger steps in steering angle could cause larger levels of understeer. In effect, the tires of the simulated car can exert more force thus torque, causing the car to turn faster.

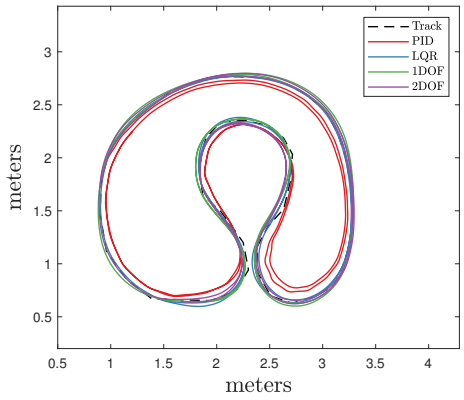
The most important results taken from this test is that although the exact tracking behaviour is not transferable to real world application, the behaviour stays similar, which is helpful for further design.

## 4.3 Performance in Simulation

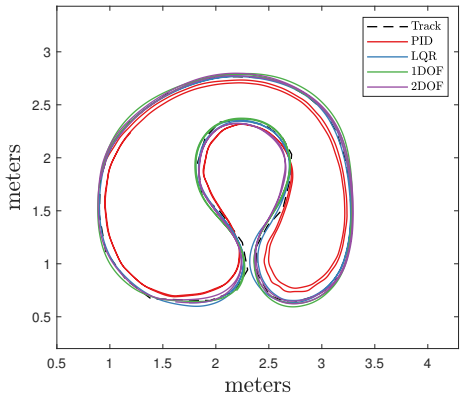
In order to verify the controllers performance in simulation, they are again simulated using the same velocity as for the real tests.

### 4.3.1 Varying Velocities

To verify that the controllers can achieve nominal performance and stability as well as their robust counterparts, the different controllers are simulated at the three different velocities found in the real experiments. The nominal parameters used for the vehicle are found in section 2.2.2 and the uncertain values used are found in section 3.7.1.2. The results of these experiments are presented below. The left figures represent the nominal parameters (they are identical to the left side plots in of the previous section) and the right side represent the uncertain parameters for each given velocity.

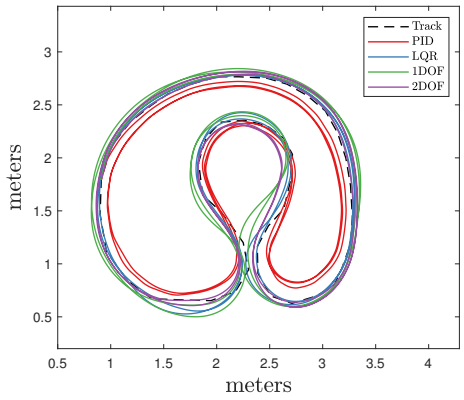



**(a)** Nominal parameters, 1.5 m/s

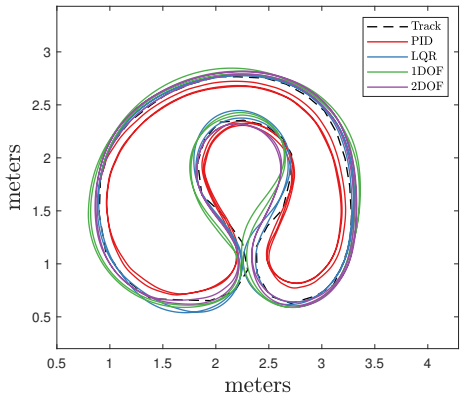



**(b)** Varied parameters, 1.5 m/s

**Figure 4.5:** The 4 different controllers in simulation for 1.5 m/s desired velocity

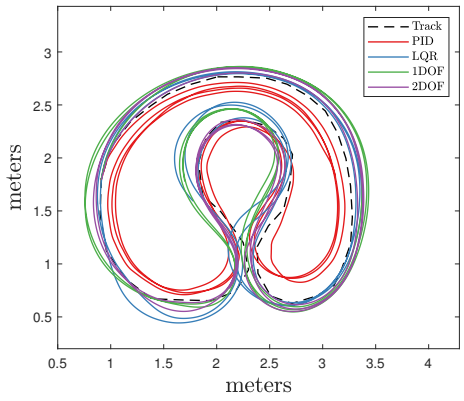



**(a)** Nominal parameters, 2.0 m/s

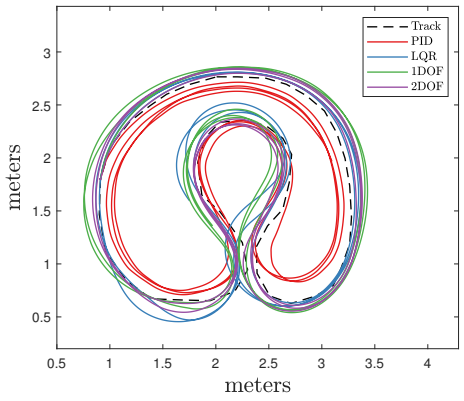



**(b)** Varied parameters, 2.0 m/s

**Figure 4.6:** The 4 different controllers in simulation for 2.0 m/s desired velocity




**(a)** Nominal parameters, 2.5 m/s




**(b)** Varied parameters, 2.5 m/s

**Figure 4.7:** The 4 different controllers in simulation for 2.5 m/s desired velocity

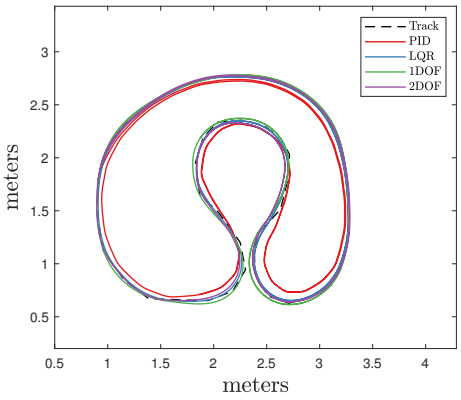
From the simulated results it is clear that the parameter change does not alter the performance of any controller in a noteworthy way. The differences are small enough to most likely be contributed to the different velocities over the two experiments. What is however very clear is that the PID controller has a significant decline in tracking performance with increasing velocity, a decline that is not as pronounced for the robust controllers.

### 4.3.2 Constant Velocity under Worst Case Uncertainty

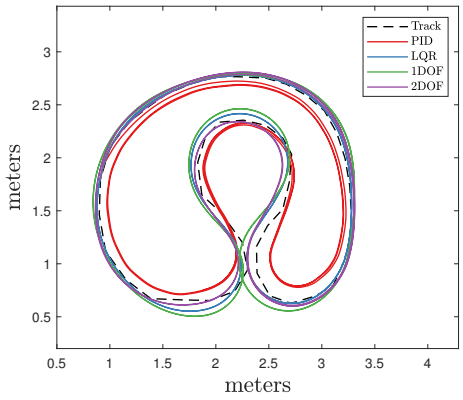
Since the possibility of changing the parameters in a controlled way is somewhat limited in real life, the simulation is used to verify stability for the more extreme cases. In this case, extreme means simulations with the parameters at values which cause the largest relative error. Such parameters are given with the help of the MATLAB command `wcgain()` which return the highest gain of the system and what values of the uncertain parameters that yield that increase. The parameters used for this test are found in table 4.1 The velocity is kept constant as to not skew the results by having different velocities for each given controller.

**Table 4.1:** Parameters used for the worst case uncertainties

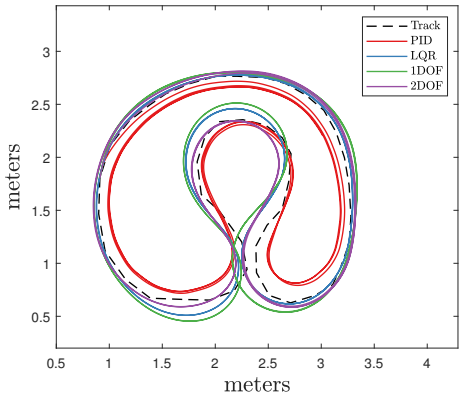
| | |
|---|---|
| $m = 0.2136$ | Mass of the vehicle. $[kg]$ |
| $\ell_f = 0.05$ | Distance from centre of mass to front axis. $[m]$ |
| $L = 0.102$ | Wheel base. $[m]$ |
| $\ell_r = L - \ell_f = 0.052$ | Distance from centre of mass to rear axis. $[m]$ |
| $\ell_w = 0.06$ | Width of the vehicle body $[m]$ |
| $I_z = \frac{m}{12}(L^2 + \ell_w^2) = 0.00025$ | Rotational inertia of vehicle, z-axis. $[kg \cdot m^2]$ |
| $C_{\alpha f} = 14$ | Tire stiffness constant, front wheel. $[N/rad]$ |
| $C_{ratio} = \frac{\ell_f}{\ell_r} = 0.99$ | Tire stiffness ratio (cornering stiffness is load dependent) |
| $C_{\alpha r} = C_{\alpha f} C_{ratio} = 13.86$ | Tire stiffness constant, rear wheel. $[N/rad]$ |




**(a)** Worst case simulation, 1.0 m/s




**(b)** Worst case simulation, 1.5 m/s




**(c)** Worst case simulation, 1.7 m/s

**Figure 4.8:** The 4 different controllers simulated using (left to right, top to bottom) $1.0m/s$, $1.5m/s$ and $1.7m/s$ constant reference velocity

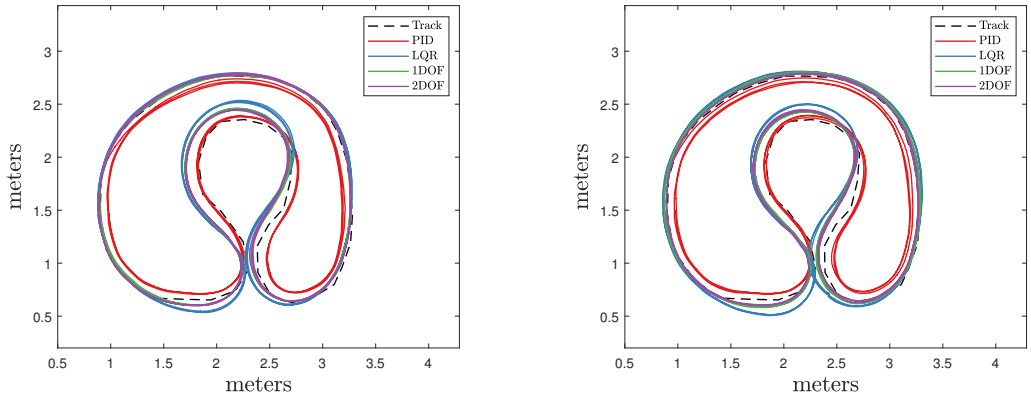
For these parameters, the difference in performance is not as apparent. Although the robust controllers fair better, they still see a similar relative increase in error. This can most likely be contributed to the fact that the parameters are at their designed limits, so the robust controllers remain stable, but they start exhibiting some unwanted behaviour. As stated before, the actual tracking of the controllers is very likely not the same as the real world experiments. As it turns out, this is exactly the case, with a significant change for the PID controller under the same criteria of constant velocity.

## 4.4 Performance in Real World Application

Perhaps more important, or rather more interesting, than the simulation results are the real world results.

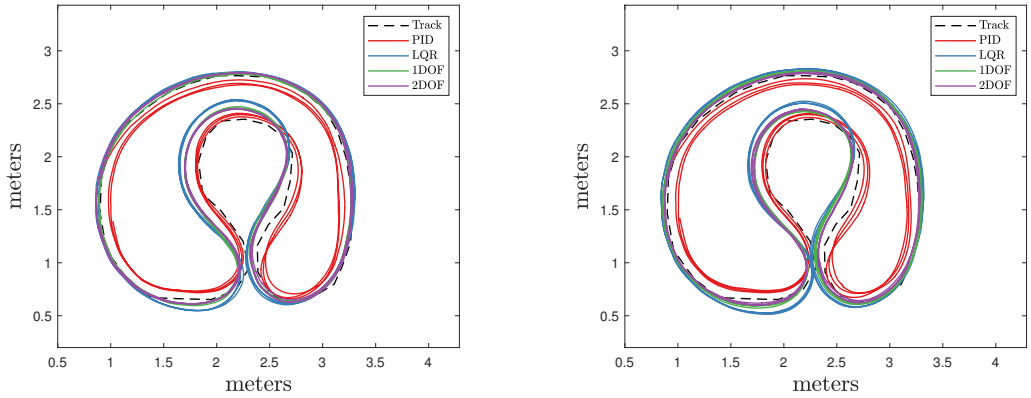
### 4.4.1 Varying Velocities

The different controllers are all run at the same track with the same starting point. Three different velocities are used, 1.5 m/s, 2.0 m/s and 2.5 m/s as maximum reference velocity for straights and sharing a minimum reference velocity of 0.2 m/s (so that we don't risk coming to a full stop in sharp turns). To gather data about robustness the vehicles are tested both nominally and with an added weight, changing both the centre of gravity and the mass, see section 3.7.1.2.



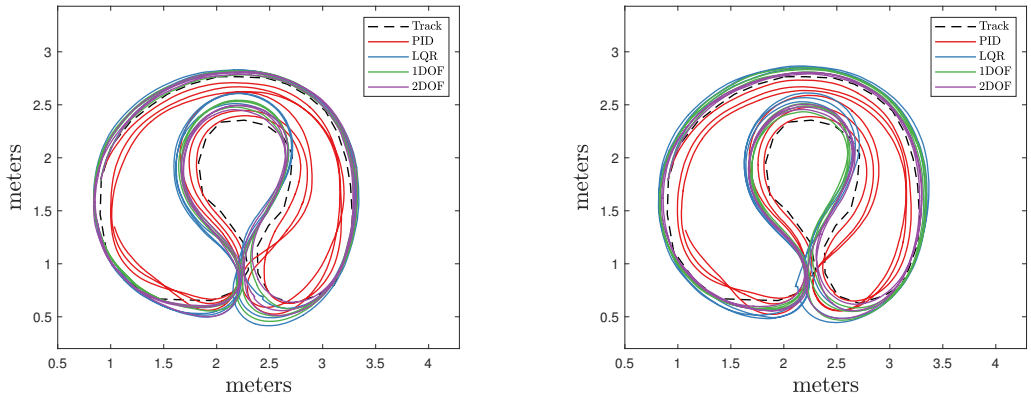

**(a)** Nominal parameters, 1.5 m/s **(b)** Varied parameters, 1.5 m/s

**Figure 4.9:** The 4 different controllers in real experiments for 1.5 m/s desired velocity




**(a)** Nominal parameters, 2.0 m/s **(b)** Varied parameters, 2.0 m/s

**Figure 4.10:** The 4 different controllers in real experiments for 2.0 m/s desired velocity



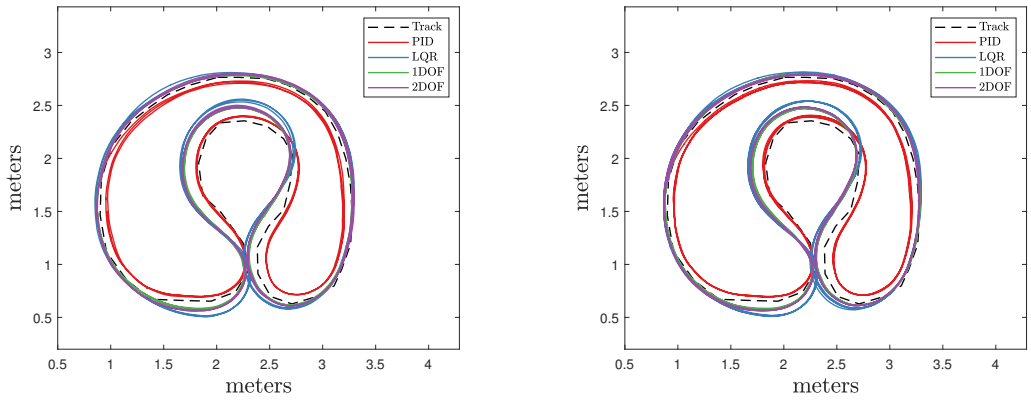

**(a)** Nominal parameters, 2.5 m/s **(b)** Varied parameters, 2.5 m/s

**Figure 4.11:** The 4 different controllers in real experiments for 2.5 m/s desired velocity

The results are very similar to that of the simulation, that the difference between nominal and uncertain parameters are not really significant. Deciding on a few measurable performance metrics, it is however clear that the robust controllers are showing a more consistent behaviour over the range of velocities than the PID.

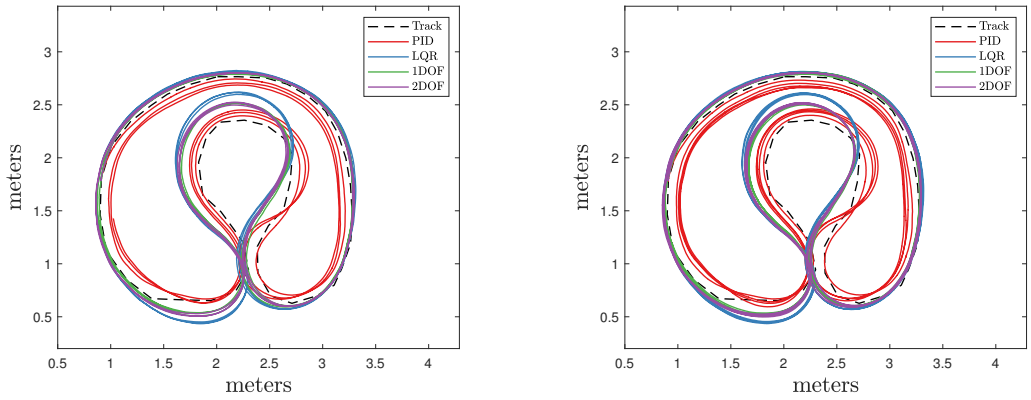
### 4.4.2 Constant Velocities

As done in the simulations the controllers are run at a constant velocities for the real test as well. This no longer biases the velocity control towards poor tracking and allows for better analysis of the ability to reject such a signal.



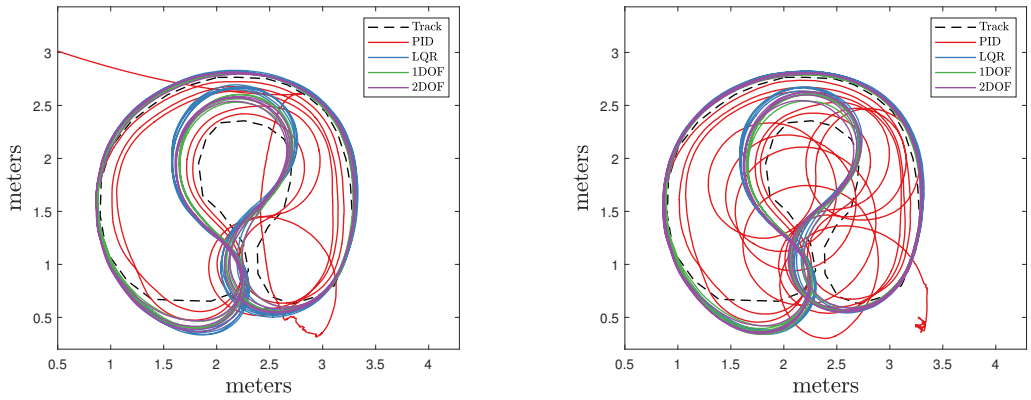

**(a)** Nominal parameters, 1.0 m/s **(b)** Varied parameters, 1.0 m/s

**Figure 4.12:** The 4 different controllers in real experiments for 1.0 m/s desired velocity




**(a)** Nominal parameters, 1.5 m/s **(b)** Varied parameters, 1.5 m/s

**Figure 4.13:** The 4 different controllers in real experiments for 1.5 m/s desired velocity




**(a)** Nominal parameters, 1.7 m/s **(b)** Varied parameters, 1.7 m/s

**Figure 4.14:** The 4 different controllers in real experiments for 1.7 m/s desired velocity

As evident by the figures, these results truly prove the superiority of the robust controllers. All of the robust controllers complete their 4 laps, while the PID can't handle the higher velocity and runs off the track.

## 4.5 Numerical Analysis of Test Data

### 4.5.1 Varying Velocities

Analysing table 4.2 below, where perhaps the $Mode(E_d)$, the most common value in the data set, and $Mode(E_d)/M(V_X)$ are the most important ones. Before calculating the mode, the values are rounded to the closest cm. The reason for this is to limit the amount of unique values to a more suitable amount. Without this method, the $Mode$ result would be more random than conclusive. Because of the way the

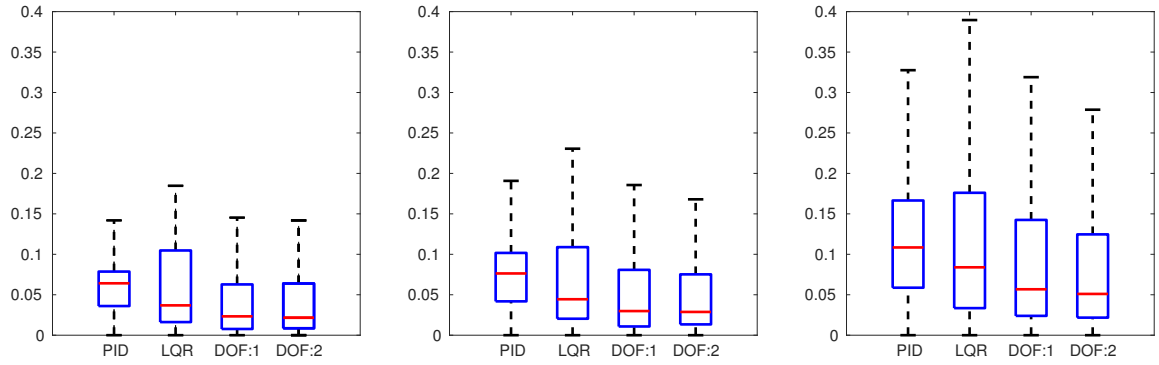
velocity reference is created, worse tracking performance can lead to favourable reference velocities thus higher overall velocities and better lap times. Diving the $Mode(E_d)$ by the mean velocity, gives a measurement on how much the errors scale with velocity indicating the robustness of the controller to velocity. As for the $Mode$, it shows that the PID controllers most frequent error is a lot higher than that of the other controllers.

**Table 4.2:** PID, LQR, 1DoF, 2DoF. Nominal parameters, with track dependent reference velocity

| $V_{max}$ 1.5 m/s | PID | LQR | 1DoF | 2DoF |
|---|---|---|---|---|
| $M(E_d)$ | 0.0588 | 0.0613 | 0.0405 | 0.0391 |
| $Mode(E_d)$ | 0.07 | 0.02 | 0.01 | 0.01 |
| $M(V_X)$ | 1.0984 | 1.0922 | 1.0898 | 1.0791 |
| $Mode(E_d)/M(V_X)$ | 0.0637 | 0.0183 | 0.0092 | 0.0093 |
| $M(T)$ | 9.5655 | 10.8791 | 10.4603 | 10.5513 |
| $D$ | 42.0277 | 47.5325 | 45.6054 | 45.5556 |
| $V_{max}$ 2.0 m/s | | | | |
| $M(E_d)$ | 0.0733 | 0.0711 | 0.0496 | 0.0487 |
| $Mode(E_d)$ | 0.1 | 0.02 | 0.01 | 0.01 |
| $M(V_X)$ | 1.3918 | 1.3952 | 1.3883 | 1.3857 |
| $Mode(E_d)/M(V_X)$ | 0.0719 | 0.0143 | 0.0072 | 0.0072 |
| $M(T)$ | 7.5723 | 8.5065 | 8.1926 | 8.2160 |
| $D$ | 42.1595 | 47.4796 | 45.4955 | 45.5519 |
| $V_{max}$ 2.5 m/s | | | | |
| $M(E_d)$ | 0.1163 | 0.1085 | 0.0835 | 0.0722 |
| $Mode(E_d)$ | 0.1 | 0.01 | 0.01 | 0.01 |
| $M(V_X)$ | 1.6468 | 1.6519 | 1.6529 | 1.6420 |
| $Mode(E_d)/M(V_X)$ | 0.0607 | 0.0061 | 0.0060 | 0.0061 |
| $M(T)$ | 6.8801 | 7.5776 | 7.3158 | 7.2267 |
| $D$ | 45.3290 | 50.0756 | 48.3754 | 47.4538 |

Noticing the difference between the mean and the median of the sampled distance errors $E_d$, box plots are used to illustrate the underlying distribution, see figure 4.15. To give a short introduction to box plots, the central mark indicates the median, and the bottom and top edges of the box indicate the 25th and 75th percentiles. The whiskers extend to the most extreme data points not considered outliers.

What can be seen from the box plots is that each controller has a distinct behaviour when regarding the distance to the track. Where the robust controllers it is also clear that the robust controllers have and maintain a lower error.



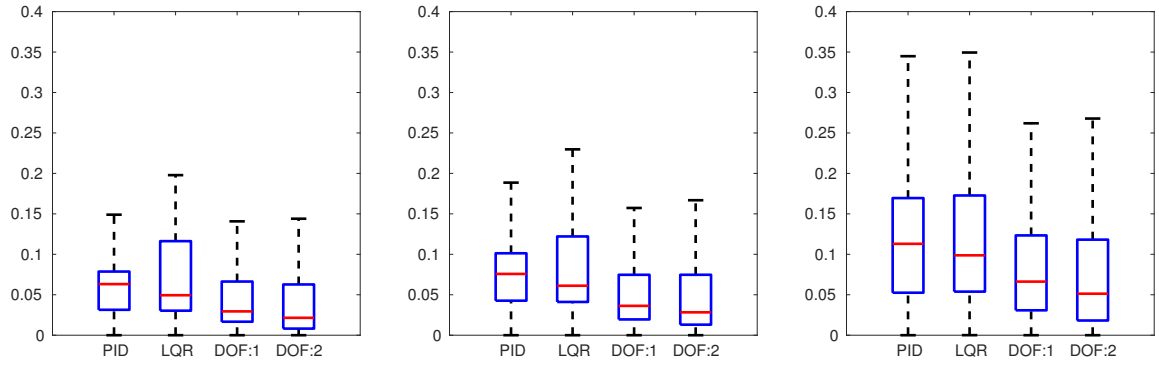

**(a)** Nominal parameters, $V_{max} = 1.5m/s$ **(b)** Nominal parameters, $V_{max} = 2.0m/s$ **(c)** Nominal parameters, $V_{max} = 2.5m/s$,

**Figure 4.15:** Box plots over the sampled errors $E_d$, with track dependant reference velocity, nominal

All of the controllers are performing increasingly worse, but the PID is the one showing the most significant decrease in performance between different velocities. From figures 4.4.1, 4.10 and 4.11 it is also very apparent that the PID is consistently cheating by running a shorter track, contributing to it's faster lap times.

**Table 4.3:** PID, LQR, 1DoF, 2DoF. Altered parameters, with track dependant reference velocity

| $V_{max}$ 1.5 m/s | PID | LQR | 1DoF | 2DoF |
|---|---|---|---|---|
| $M(E_d)$ | 0.0574 | 0.0676 | 0.0416 | 0.0386 |
| $Mode(E_d)$ | 0.07 | 0.04 | 0.02 | 0.01 |
| $M(V_X)$ | 1.0963 | 1.0867 | 1.0820 | 1.0811 |
| $Mode(E_d)/M(V_X)$ | 0.0639 | 0.0368 | 0.0185 | 0.0092 |
| $M(T)$ | 9.5853 | 11.0714 | 10.6150 | 10.5412 |
| $D$ | 42.0314 | 48.1335 | 45.9455 | 45.5922 |
| $V_{max}$ 2.0 m/s | | | | |
| $M(E_d)$ | 0.0721 | 0.0783 | 0.0511 | 0.0472 |
| $Mode(E_d)$ | 0.09 | 0.05 | 0.03 | 0.01 |
| $M(V_X)$ | 1.3877 | 1.3988 | 1.3986 | 1.3825 |
| $Mode(E_d)/M(V_X)$ | 0.0649 | 0.0357 | 0.0214 | 0.0072 |
| $M(T)$ | 7.5819 | 8.6285 | 8.1994 | 8.2458 |
| $D$ | 42.0921 | 48.2737 | 45.8783 | 45.6060 |
| $V_{max}$ 2.5 m/s | | | | |
| $M(E_d)$ | 0.1193 | 0.1144 | 0.0831 | 0.0761 |
| $Mode(E_d)$ | 0.14 | 0.07 | 0.01 | 0.01 |
| $M(V_X)$ | 1.6485 | 1.6705 | 1.6595 | 1.6311 |
| $Mode(E_d)/M(V_X)$ | 0.0849 | 0.0419 | 0.006 | 0.00610 |
| $M(T)$ | 6.9355 | 7.6017 | 7.2281 | 7.2895 |
| $D$ | 45.7346 | 50.8128 | 47.9902 | 47.5614 |




**(a)** Varied parameters, $V_{max} = 1.5m/s$ **(b)** Varied parameters, $V_{max} = 2.0m/s$ **(c)** Varied parameters, $V_{max} = 2.5m/s$,

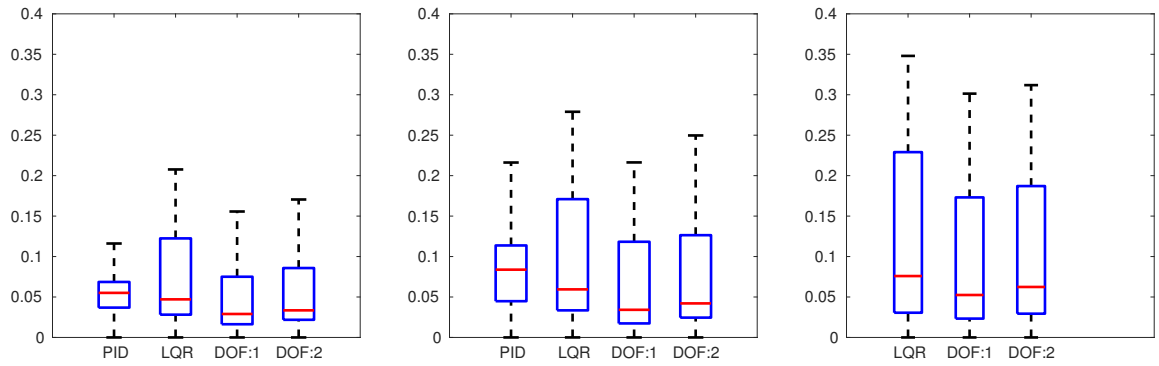
**Figure 4.16:** Box plots over the sampled errors $E_d$, with track dependant reference velocity, uncertain

### 4.5.2 Constant Velocities

The different controllers are all run at the same track with the same starting point. Three different speeds are used, $1.0m/s$, $1.5m/s$ and $1.7m/s$ and the velocity control strives to keep this constant. This is done to ensure that none of the controllers can benefit from poor tracking. The maximum velocity of $1.7m/s$ is chosen as it is slightly below the maximum velocity at which the 2DoF $\mathcal{H}_\infty$ is able to complete the track.

**Table 4.4:** PID, LQR, 1DoF, 2DoF. Nominal parameters, constant reference velocity

| $V$ 1.0 m/s | PID | LQR | 1DoF | 2DoF |
|---|---|---|---|---|
| $M(E_d)$ | 0.0528 | 0.0720 | 0.0472 | 0.0544 |
| $Mode(E_d)$ | 0.06 | 0.03 | 0.02 | 0.03 |
| $M(V_X)$ | 0.9517 | 0.9561 | 0.9539 | 0.9529 |
| $Mode(E_d)/M(V_X)$ | 0.0630 | 0.0314 | 0.0210 | 0.0315 |
| $M(T)$ | 11.2114 | 12.7877 | 12.2497 | 12.4086 |
| $D$ | 42.6793 | 48.9137 | 46.7474 | 47.3053 |
| $V$ 1.5 m/s | | | | |
| $M(E_d)$ | 0.0805 | 0.0975 | 0.0650 | 0.0729 |
| $Mode(E_d)$ | 0.1 | 0.04 | 0.02 | 0.03 |
| $M(V_X)$ | 1.4128 | 1.4232 | 1.4176 | 1.4158 |
| $Mode(E_d)/M(V_X)$ | 0.0708 | 0.0281 | 0.0141 | 0.0212 |
| $M(T)$ | 7.8098 | 8.8289 | 8.4095 | 8.4725 |
| $D$ | 44.1424 | 50.2645 | 47.6818 | 47.9879 |
| $V$ 1.7 m/s | | | | |
| $M(E_d)$ | n/a | 0.1257 | 0.0943 | 0.1031 |
| $Mode(E_d)$ | n/a | 0.01 | 0.03 | 0.04 |
| $M(V_X)$ | n/a | 1.6100 | 1.5981 | 1.5988 |
| $Mode(E_d)/M(V_X)$ | n/a | 0.0062 | 0.0188 | 0.0250 |
| $M(T)$ | n/a | 8.1303 | 7.7851 | 7.8230 |
| $D$ | n/a | 52.3701 | 49.7686 | 50.0363 |

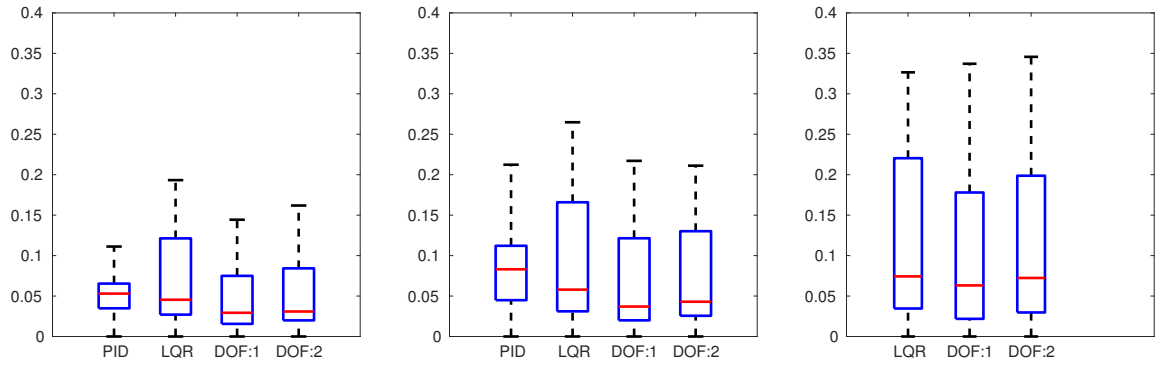



**(a)** Nominal parameters, $V_{ref} = 1.0m/s$ **(b)** Nominal parameters, $V_{ref} = 1.5m/s$ **(c)** Nominal parameters, $V_{ref} = 1.7m/s$,

**Figure 4.17:** Box plots over the sampled errors $E_d$, constant reference velocity, nominal

**Table 4.5:** PID, LQR, 1DoF, 2DoF. Altered parameters, constant reference velocity

| $V$ 1.0 m/s | PID | LQR | 1DoF | 2DoF |
|---|---|---|---|---|
| $M(E_d)$ | 0.0507 | 0.0699 | 0.0458 | 0.052 |
| $Mode(E_d)$ | 0.06 | 0.03 | 0.02 | 0.03 |
| $M(V_X)$ | 0.9512 | 0.9587 | 0.9571 | 0.9609 |
| $Mode(E_d)/M(V_X)$ | 0.0631 | 0.0313 | 0.0209 | 0.0312 |
| $M(T)$ | 11.2469 | 12.7240 | 12.1888 | 12.2451 |
| $D$ | 42.7955 | 48.7970 | 46.6667 | 47.0676 |
| $V$ 1.5 m/s | | | | |
| $M(E_d)$ | 0.0796 | 0.0951 | 0.0662 | 0.0735 |
| $Mode(E_d)$ | 0.11 | 0.03 | 0.03 | 0.03 |
| $M(V_X)$ | 1.4129 | 1.4268 | 1.4231 | 1.4263 |
| $Mode(E_d)/M(V_X)$ | 0.0779 | 0.0210 | 0.0211 | 0.0210 |
| $M(T)$ | 7.8256 | 8.7495 | 8.3653 | 8.4212 |
| $D$ | 44.2362 | 49.9433 | 47.6210 | 48.0475 |
| $V$ 1.7 m/s | | | | |
| $M(E_d)$ | n/a | 0.1222 | 0.1030 | 0.1122 |
| $Mode(E_d)$ | n/a | 0.03 | 0.01 | 0.03 |
| $M(V_X)$ | n/a | 1.6097 | 1.6061 | 1.6109 |
| $Mode(E_d)/M(V_X)$ | n/a | 0.0186 | 0.0062 | 0.0186 |
| $M(T)$ | n/a | 8.0420 | 7.8315 | 7.8734 |
| $D$ | n/a | 51.7898 | 50.3197 | 50.7363 |



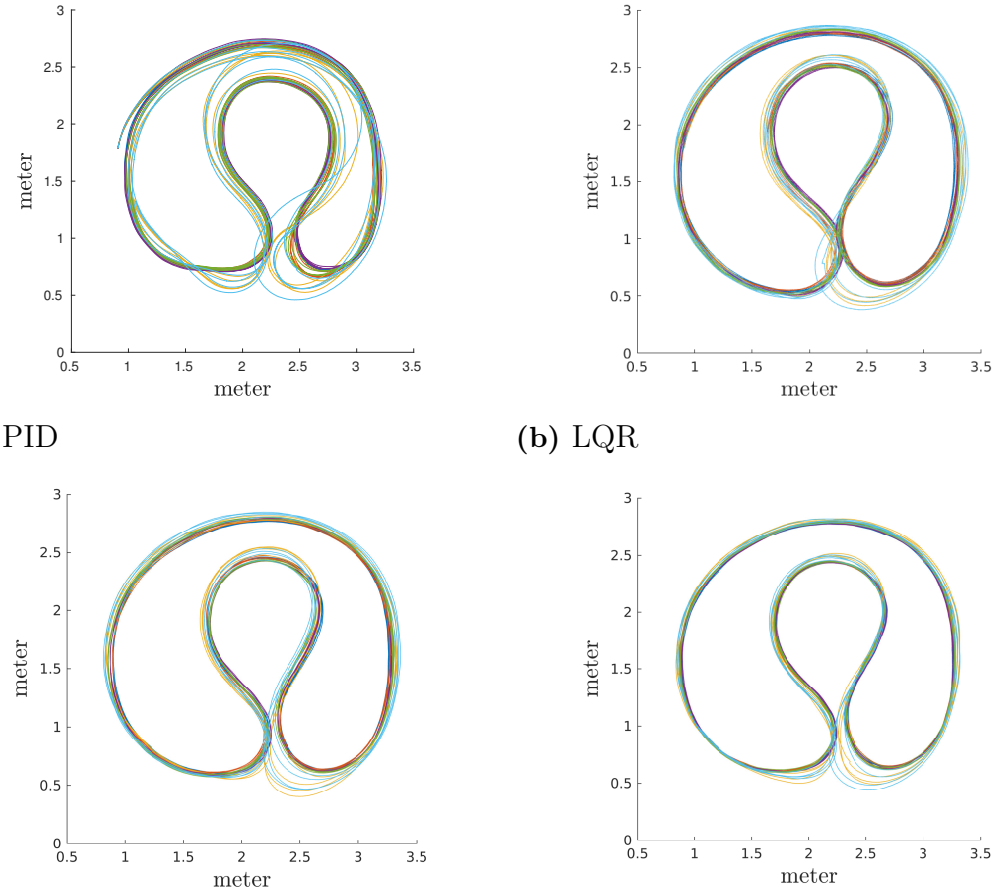

**(a)** Varied parameters, $V_{ref} = 1.0m/s$ **(b)** Varied parameters, $V_{ref} = 1.5m/s$ **(c)** Varied parameters, $V_{ref} = 1.7m/s$,

**Figure 4.18:** Box plots over the sampled errors $E_d$, constant reference velocity, uncertain

Both table 4.4 and 4.5 lack data for the PID controller at $1.7m/s$, caused by the PID controllers inability to complete the track at this velocity. It literally ran into the wall. The trend in the errors are very similar to that of previous test, with updated reference velocities. The PID sees a significant decrease in performance until it ultimately fails to track entirely. It still has the fastest lap times thanks to its cheating.

Although what is presented in the above chapter is definitely satisfactory, it might also be a bit difficult to grasp. Therefore, one final metric is presented. Robustness is now seen as level of repeatability. How does the performance of the controller differ over varying system dynamics?

In figure 4.19 and 4.20. What is shown are plots of the trajectories for each controller over all the experiments (where the controller did not fail), that is to say nominal system, uncertain system and each of the different velocities of those systems, all in the same plot.



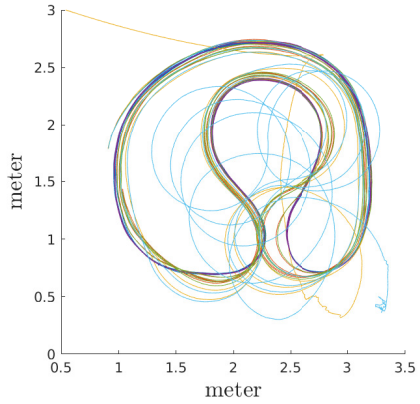

**(a)** PID

**(b)** LQR

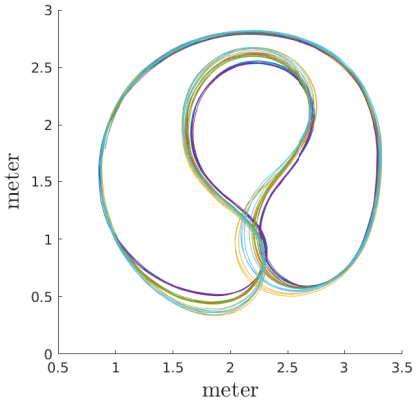
**(c)** 1DoF

**(d)** 2DoF

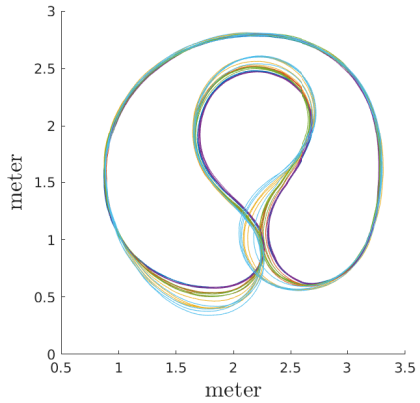
**Figure 4.19:** All the trajectories at different velocities. With track dependant reference velocity.



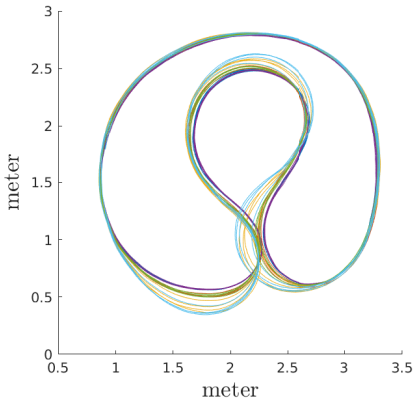

**(a)** PID




**(b)** LQR




**(c)** 1DoF




**(d)** 2DoF

**Figure 4.20:** All the trajectories at different velocities. With constant reference velocity.

What becomes very apparent is that no matter what we throw at the robust controllers, their behaviour remains very similar for all experiments. What this means is that the robustness of the controller is designed very well, but the performance aspect of it needs a bit of tuning. In other words, the variance of the trajectory is minimised, now it just needs to be shaped.

## 4.6 Comments

Trying to make sense of all the measurements presented seems at a first glance to be a daunting task. What must be asked is which aspects of the controllers that are examined. Some numbers might at a first seem to indicate that the PID controller is the highest performing one judging on average lap times, $M(T)$ in tables 4.2 to 4.5. However, looking at the total distance travelled for each scenario it is shown that the PID travels a shorter path than any of the other controllers. This of course adds a fair bit of doubt to the PIDs seemingly superior performance in terms of lap time. This clearly differing behaviour from the other controllers, can be explained by the underlying kinetic motion model which both the LQR- and the two $\mathcal{H}_\infty$ -

controllers use. The PID is more or less blind to anything else than pure input, so is technically also the LQR. But the $\mathcal{H}_\infty$ controllers have a *knowledge* of the motion of the vehicle. The PID was also designed using a hardware in the loop approach, using a constant input signal to the engine. While performing more than adequate close to the velocity in which is was designed the tests show a pronounced decrease in performance when deviating from it.

Even though the conditions of the test runs are identical, the fatal failure of the PID controller is due to how the reference trajectory is calculated. It works fine up until too large tracking errors result in the car having to attempt tracking a point that's too close or even behind it. This means that the actual tracking stability of the system heavily relies on the trajectory calculation, which is outside of the motion model.

# 5
# Conclusion

**The robust approach for trajectory control of a self-driving R/C car sees high levels of performance for increased velocities. It is not without its twists and turns, but with those sorted out the payoff is clear.**

Collecting data from both simulation and more importantly from *hardware in the loop* tests both tell a similar story; the different metrics calculated from the data and how it relates to robustness seem to indicate that for this case, repeatability is the main metric of robustness. That is, a similar performance over a wider range of uncertain dynamics, where the velocity proved to be the most prominent one. Overlaying the different paths taken by each controller at the different velocities and altered parameters, did illustrate clearly how well the different controllers were able to keep a consistent performance no matter the circumstances. In other words, they are robust. The tracking performance of the robust controllers might not be quite as obviously improved over the PID as one would desire, but with a bit of tuning of the performance aspect of the robust controllers, we are certain they would more clearly outperform the lesser controllers.

Does the robust approach provide a significant gain in performance over the more common control strategies? Well, as with everything, it's a trade-off. The PID controller is quick to implement and will evidently often provide the user with the performance it desires. Even though the PID might outperform the robust controller for a given scenario it's design clearly has inherent flaws (such as running into the wall at high velocities) that just are not found in the robust designs.

The $\mathcal{H}_\infty$ controllers might seem complex, but one must not forget that many of the parameters are often measurable or bear an intuitive physical meaning easing the design of the dynamic weights. That being said, it is far from an easy task and the superiority over less complex design methods might not always be apparent.

In the end of the day however, one may sleep comfortably knowing that in a system riddled by uncertainties, the $\mathcal{H}_\infty$ controller rules supreme.

# Bibliography


[1] "Control System Design," https://www.cds.caltech.edu/~murray/courses/cds101/fa02/caltech/astrom.html, accessed: 2018-05-20.

[2] O. Garpinger, T. Hägglund, and K. J. Åström, "Performance and robustness trade-offs in pid control," *Journal of Process Control*, vol. 24, no. 5, pp. 568 – 577, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0959152414000730

[3] D. A. Allan, M. J. Risbeck, and J. B. Rawlings, "Stability and robustness of model predictive control with discrete actuators," in *2016 American Control Conference (ACC)*, July 2016, pp. 32–37.

[4] K. Zhou and J. Doyle, *Essentials of Robust Control*, ser. Prentice Hall Modular Series for Eng. Prentice Hall, 1998. [Online]. Available: https://books.google.se/books?id=QviHQgAACAAJ

[5] R. Rajamani, *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[6] H. Pacejka, *Tire and vehicle dynamics*. Elsevier, 2005.

[7] D. S. Naidu, *Optimal control systems 1st.Ed*. CRC Press, 2002.

[8] J. Riccati, "Animadversiones in aequationes differentiales secundi gradus," http://www.17centurymaths.com/contents/euler/rictr.pdf, 1724(Translated by Ian Bruce, 2007), accessed: 2018-04-30.

[9] S. Skogestad and I. Postlethwaite, *Multivariable Feedback Control: Analysis and Design*. USA: John Wiley & Sons, Inc., 2005.

[10] S. J. Julier and J. K. Uhlmann, "Unscented filtering and nonlinear estimation," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, 2004.

[11] E. Wan, "Sigma-point filters: An overview with applications to integrated navigation and vision assisted control," in *Nonlinear Statistical Signal Processing Workshop, 2006 IEEE*. IEEE, 2006, pp. 201–202.

[12] S. S. Haykin *et al.*, *Kalman filtering and neural networks*. Wiley Online Library, 2001.

[13] "Kyosho MR-03S," http://www.kyoshoamerica.com/MR-03-Sports_c_1166.html, accessed: 2018-02-06.

[14] "NodeMCU Development Kit," https://www.seeedstudio.com/NodeMCU-v2-Lua-based-ESP8266-development-kit-p-2415.html, accessed: 2018-02-06.

[15] "rosserial_arduino," http://wiki.ros.org/rosserial_arduino, accessed: 2018-02-08.

[16] "ESP8266 Arduino," https://github.com/esp8266/Arduino, accessed: 2018-02-08.

[17] "Flea3 USB3 Vision," https://www.ptgrey.com/flea3-usb3-vision-cameras, accessed: 2018-06-01.
[18] "Arduino Uno," https://store.arduino.cc/arduino-uno-rev3, accessed: 2018-04-09.
[19] "Adafruit PowerBoost 1000," https://www.adafruit.com/product/2465, accessed: 2018-04-27.
[20] "Autodesk fusion 360," https://www.autodesk.com/products/fusion-360/overview, accessed: 2018-04-19.
[21] J. G. Ziegler and N. B. Nichols, "Optimum settings for automatic controllers," *trans. ASME*, vol. 64, no. 11, 1942.
[22] "Scale-model vehicle analysis for the design of a steering controller," https://pdfs.semanticscholar.org/ff27/25b35aa255d5f5bee92b54ee5a046e6bd947.pdf, accessed: 2018-05-20.
[23] "ucover()," https://se.mathworks.com/help/robust/ref/ucover.html, accessed: 2018-05-26.
[24] L. Svensson, "A coordinated turn model," Lecture Notes, Signal Processing Group, Department of Signals and Systems at Chalmers University of Technology, 2015.