UNIVERSITY OF CAMBRIDGE

Department of Engineering

Investigating Aspects of Autonomous Bicycle Control

Author Name: Sia Han Wei

Supervisor: Dr. Richard Roebuck

Date: 30 May 2018

I hereby declare that, except where specifically indicated, the work submitted herein is my own original work.

Signed _____ *date* 30/5/2018

Investigating Aspects of Autonomous Bicycle Control

Sia Han Wei (hws23), Hughes Hall

May 29, 2018

Abstract

The aim of the project is to implement and evaluate control strategies on small scale bicycles travelling at constant speed. The controllers built should be able to stabilize the system and allow smooth turning, while using steering as the only control input. Proportional-Derivative (PD) and the Linear Quadratic Regulator (LQR) control schemes were chosen and implemented on two Lego bicycles.

Lego bicycle 1 was successfully stabilized using proportional-derivative (PD) control applied to the lean angle. The feedback parameters were chosen based on a linearised inverted pendulum model and then experimentally tuned. A feedforward gain was then added to enable turning. A working bicycle that could be remotely steered was achieved but significant deviation from theory was observed, revealing possible model deficiencies.

Lego bicycle 2 was constructed with several improvements made to the frame. An LQR full state feedback controller was designed using the more complicated linearised Whipple model. Although successful in simulations, the controller was not able to stabilize the bicycle in practice.

The bicycle control problem remains incompletely solved. Model identification and validation is very challenging because the system is unstable with dynamics that are dependent on forward speed. This problem has to be addressed properly before further progress can be made. Nevertheless, PD and LQR control have been shown to be viable ways of stabilizing and controlling an autonomous bicycle.

The report is organized as follows. Section 1 gives a short introduction to bicycle design and human rider control, followed by a review of successful controllers designed by others. Section 2 gives an overview of the available Lego hardware used to construct the bicycles. Sections 3 and 4 describe Lego bicycles 1 and 2 respectively. Both these sections will include theoretical modelling, controller design, implementation details, and experimental results. Conclusions and possible future work follow. Derivations of the inverted pendulum model and Whipple model used are shown in appendices A and B.

Contents

1 Introduction

The bicycle has existed for over 200 years, with its design largely evolving via trial and error (Figure 1). The Draisine was the first steerable two-wheeled, human-powered vehicle to be built. These were made of wood and had to be foot-pushed. The subsequent "Boneshaker" and Penny Farthing designs sported pedals on the front wheel hub, with the larger front wheel on the Penny Farthing giving higher speeds. However, trying to steer the front wheel while pedalling was both awkward and dangerous. A major breakthrough came with the invention of the safety bicycle in the late 1880s. Besides having pneumatic tyres and a rear wheel chain drive, it also included a tilted and offset steer axis. This bicycle was safer, easier to learn and more comfortable to ride. With the safety bicycle, cycling became widely adopted as a choice of transport. Since then, bicycle design has remained essentially unchanged.

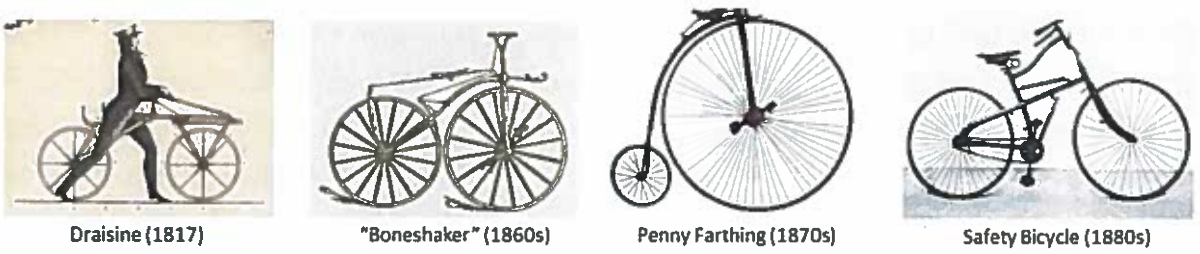


Figure 1: Evolution of bicycle design

Due to its light weight and the low rolling resistance of its two in-line wheels, the bicycle is the most energy efficient form of human transport available. It is also inexpensive and able to access difficult terrain. A self-driving bicycle capable of navigation is thus a possible means of autonomous delivery and transport that should be further explored.

Autonomous bicycles can also solve the problems faced by bike sharing services today. In recent years, dock-less bike sharing services such as Ofo and Mobike have increased in popularity across many cities. While convenient for riders, the dock-less parking feature has resulted in indiscriminately parked bikes causing obstructions. Bike share operators also have to maintain large fleets of bicycles, of which only a small fraction is in use at any one time, further contributing to the above problem. As a result, legislation enforcing fixed parking zones have been passed in several countries. This not only removes the primary attraction of dock-less services, but also imposes additional cost on operators. Smart bikes capable of redistributing themselves across cities will solve these problems.

This project is a step towards a fully autonomous bicycle, in which control techniques for bicycle stabilization and steering were investigated.

1.1 How we ride bicycles

It is first necessary to understand the workings of a bicycle. A stationary bicycle is an inherently unstable system that is very similar to the inverted pendulum. However, an uncontrolled bicycle moving above some minimum speed can be observed to exhibit self-stability. When given a sudden lateral push, the uncontrolled moving bicycle stabilizes itself by automatically steering into the fall.

This coupling of lean to steer can be explained by several properties of the front steering column such as gyroscopic precession of the spinning front wheel and the position of the steering column's centre of mass (which is lower than that of the rear frame, causing it to fall faster and hinge into the fall) [1]. Positive trail, the distance from the front ground contact to the point where the steering axis intersects the ground plane, is also important. When a bicycle with positive trail is leaned, the ground contact forces exert a torque about the steering axis that turns it in the same direction as the lean. The front wheel will also tend to self align behind the steering axis as with a trolley castor [2]. These three factors contribute to different extents depending on the actual bike design and it is generally accepted that no single element alone is a necessary or sufficient condition for self-stability [3]. Nevertheless, it is important to note that the traditional steering column design seen on all bicycles provides some form of feedback from lean to steer that can stabilize the uncontrolled bicycle over a limited range of speeds.

Experience shows that bicycles require more control effort at lower speeds, and become practically impossible to stabilize below a certain threshold speed. Special experimental bicycles that are difficult for a person to ride also tend to not be self-stable [3]. Handling and self-stability are thus very closely related properties and it can be inferred that a human rider makes use of the built-in feedback of the front fork when cycling. This allows for efficiency since control effort is minimized. Figure 2 shows the human rider with 2 degrees of freedom, applying a steer torque to the handlebars and using torso lean to control bicycle balance while tracking some desired path $(x, y)_{\text{desired}}$. When travelling straight, riders usually grip the handlebars lightly and allow the self-stable properties of the front fork to stabilize the bike, thus weaving slightly about the desired straight path. When turning, the rider first counter-steers and leans into the turn, before applying torque on the steering handles in the *opposite* direction to resist the self correcting tendencies of the front fork, achieving a smooth turn radius.

Attempts have been made to identify the human controller shown in Figure 2. This is attractive because it allows for natural human-like bicycle handling. A human can also easily ride many different bicycles after learning on a single bicycle, meaning that the human controller is robust and has some method of rapid parameter tuning. The goal is to design a controller that can replace the human rider and successfully control the bicycle.

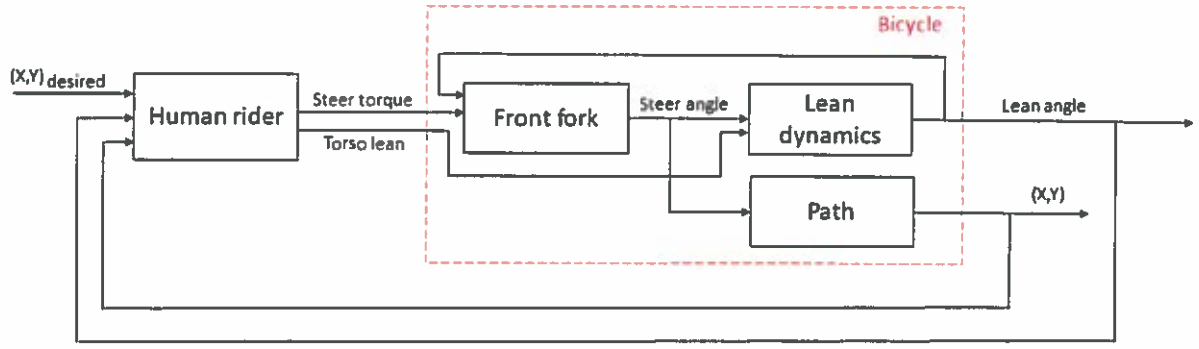


Figure 2: Human controller

1.2 Literature Review

Linearised equations of motion for a bicycle have been presented in a few papers. Astrom *et al.* uses an "inverted pendulum" model that treats the bicycle as a single point mass at its centre of gravity [4]. This model is appealing because it only requires six easily obtained measurements from the physical bicycle. However, it leaves out wheel gyroscopic effects and treats the steering column as being massless, both of which are likely to be non-negligible on a full scale bicycle. A more complicated Whipple bicycle model is derived by Papadopoulos and Meijaard *et al.*, where the bicycle is modelled as four connected rigid bodies (rear wheel, rear frame, front wheel and front fork) [7]. This model has been validated via dynamics simulation software as well as on a physical bicycle [6]. However, it requires 17 parameters, some of which are difficult to accurately obtain from the physical bicycle. Both these models assume no tyre slip and point ground contacts. Through static balance on the steering column in the inverted pendulum model, and dynamic analysis of the steering column in the Whipple model, both models show the innate feedback property described above and explain self-stability in certain speed regimes. Detailed derivations of these models were quite difficult to find and follow, so slightly modified versions of these have been compiled and included in appendices A and B.

Although other methods such as control moment gyroscopes, reaction wheels and weight shifting can be used, only steering control will be considered because of power and size/weight constraints. Trying to meet the dual requirements of stability and path tracking using just 1 degree of freedom will make the problem more challenging.

Either steering torque or angle can be chosen to be the control input. Although using steering torque seems like the natural choice, it is not readily measurable and requires very detailed modelling of the DC steering motor. Controlling steering angle requires a servo motor or a steering controller. By controlling the steer angle directly, the self-stable properties described above will also be forfeited. This represents an important design

choice that has to be made early on.

There have been several attempts to stabilize a bicycle moving at constant velocity via steering control. Classical control techniques have been used successfully on both full scale and model bicycles. Variations of proportional-integral-derivative (PID) control of steer angle or steer torque with feedback of the lean angle are the most common controller designs. He *et al.* controlled steer angle, using PD feedback on lean angle together with a feed-forward portion for turning [8]. Their full sized bicycle showed smooth turning while being stable. Michini *et al.* controlled steering torque instead, using a similar PD controller [9]. Mutsaerts stabilized a Lego bicycle by controlling steering torque and using only derivative control [10].

State space controllers have also been previously applied. The Linear Quadratic Regulator (LQR) method has been used to control Lego bicycles [11],[12] while the Linear Quadratic Integral (LQI) method has been applied on a small scale bicycle [13].

Reinforcement learning has also been used to stabilize bicycles in simulation [14],[15]. The benefit of this method is that no explicit modelling of the bicycle dynamics is required. The agent interacts with the system and learns the correct model and control policy through repeated trials. However, it might take several thousand iterations before a stabilizing controller is found, making it impractical on a physical bicycle.

In general, there are many more successful controllers in simulation than on a physical bicycle. Of these, PID control is the most commonly used method.

1.3 Aims

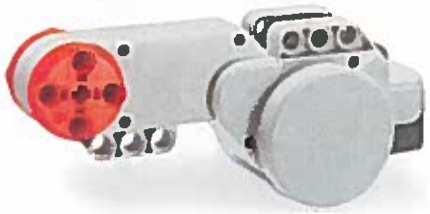
The aim of the project is to implement and evaluate control strategies on small scale bicycles travelling at constant speed. The controllers built should be able to stabilize the system and allow smooth turning, while using steering as the only control input. Proportional-Derivative (PD) and the Linear Quadratic Regulator (LQR) control schemes were chosen and implemented on two Lego bicycles.

2 Hardware

The bicycle frame was constructed using Lego Mindstorms parts. The sensors, motor and micro-controller available are shown below

HiTechnic Gyro sensor[1]
Single axis
Returns tilt rate up to $\pm 360\,\text{degs}^{-1}$
Resolution of $1\,\text{degs}^{-1}$

HiTechnic Acceleration sensor[2]
Three axis
Returns acceleration up to $\pm 2g$
Resolution of 200 counts per g



Lego NXT DC motor
Input integer in range $[-100, 100]$ (PWM duty cycle)
Maximum speed $800\,\text{deg}\,\text{s}^{-1}$
Built in encoder with resolution of 1 deg

EV3 brick
300MHz ARM9 processor
64MB RAM, 16MB Flash
Wireless connectivity with Wi-Fi dongle

The Simulink support package[3] for Lego EV3 was also used. This allowed the control scheme to be designed in Simulink, compiled and then downloaded to the EV3 brick. A Wi-Fi link was then used for real-time data acquisition during trial runs. Results were plotted and analysed in Matlab.

[1] http://www.hitechnic.com/cgi-bin/commerce.cgi?preadd=action&key=NGY1044
[2] http://www.hitechnic.com/cgi-bin/commerce.cgi?preadd=action&key=NAC1040
[3] https://uk.mathworks.com/matlabcentral/fileexchange/45075-simulink-support-package-for-lego-mindstorms-ev3-hardware

3 Lego Bicycle 1

A PD controller for a constant velocity Lego bicycle will be analysed in this section. The bicycle build used is based on the one designed by Mutsaerts [10] but with several modifications (Figure 3). Firstly, a guide rail was added from which the bicycle could be more easily launched, allowing enough time for it to get up to the required speed before the steer controller kicked in. A PID drive controller was also added so that the bicycle would reach the desired speed fast enough and track it well. A Kalman filter was then used as a state observer to give estimates of the lean angle and steer rate which could not be directly measured. Lastly, the PD controller gains were tuned and a feed-forward control path was added to allow for turning.

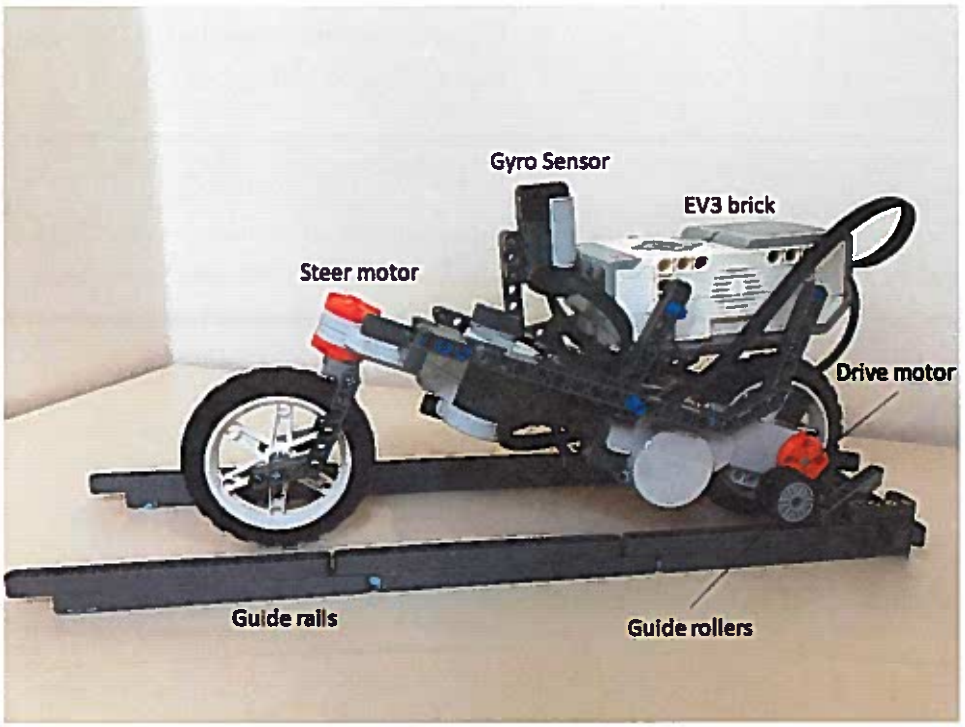


Figure 3: Lego bicycle 1

Mass/kg	m	0.711
Height of CG/m	h	0.09
Distance to CG/m	a	0.075
Wheel base/m	b	0.23
Wheel radius/m	r	0.04
Trail/m	c	0.011
velocity/ms^{-1}	v	0.5
Steering axis angle/deg	λ	65

Table 1: Bicycle parameters of the Lego bicycle
(More clearly defined in Appendix A)

3.1 Dynamics modelling - Inverted pendulum model

The inverted pendulum model of a bicycle derived by Astrom is shown in appendix A. Relevant parameters were measured from Lego Bicycle 1 and are shown in Table 1. Although the model may be overly simplified, it reveals several interesting points. Changing the signs so that the steer angle δ and lean angle φ are both rightward positive gives the bicycle lean dynamics equation. The equation comes from conservation of angular momentum about the lean axis x

$$\frac{dL_x}{dt} = T_{\text{gravity}} + T_{\text{centripetal}}$$

$$mh^2\ddot{\varphi} + \frac{mahv\sin\lambda}{b}\dot{\delta} = mgh\varphi - \frac{m(v^2h - acg)\sin\lambda}{b}\delta$$

The equation shows how steer can be used to influence lean. Steering while travelling causes the bike to start turning. The centripetal acceleration exerts a torque $T_{\text{centripetal}}$ on the bicycle that affects its lean. Steering can thus be used to produce a righting torque that stabilizes the bicycle. In transfer function form,

$$G(s) = \frac{\varphi(s)}{\delta(s)} = -\frac{av\sin\lambda}{bh}\left[\frac{s + \frac{v^2h - acg}{vah}}{s^2 - \frac{g}{h}}\right] \tag{1}$$

The bicycle has 2 poles and 1 zero at

$$z = -\frac{v^2h - acg}{vah}$$

$$p = \pm\sqrt{\frac{g}{h}}$$

It is an unstable system with the open loop pole locations only influenced by h, the height of the CG. If the steer angle can be directly controlled and the zero is in the left half plane (LHP), the system can be stabilized by proportional control alone.

$$\delta = k_p\varphi$$

The Routh-Hurwitz criterion gives the necessary proportional gain k_p.

$$k_p > \frac{bgh}{(hv^2 - acg)\sin\lambda}$$

The size of k_p required decreases with increasing h and v. Intuitively, a shorter inverted pendulum is more difficult to keep upright than a taller one. The height of the centre of gravity of an inverted pendulum determines how fast it falls, and hence how aggressive the controller must be to stabilize the system. Having a large v will give a larger righting torque for the same steer and require less control effort. This is important because a bicycle has maximum steering limits of around ± 50 degrees. At large steering angles, the small angle approximation will also not hold and the model will no longer be accurate. A bicycle design that is controllable in theory but requires very large control gains may be

impossible to achieve.

The zero location depends on v. If the zero crosses into the right half plane (RHP), G becomes non-minimum phase with the RHP zero being to the left of the RHP pole. This makes stabilization with a PID controller impossible. An unstable controller will be necessary if the system is to be stabilized. The step response from steer to lean will also exhibit undershoot which is undesirable. This constraint yields a threshold speed below which steering control will not work.

$$v^2 h > acg$$
$$v > \sqrt{\frac{acg}{h}}$$

This works out to be $v > 0.269$m/s for the Lego bicycle. This control scheme can be used if a sufficiently fast servo motor is available. However, since the angle of the Lego motors cannot be controlled directly, the motors have to be modelled first.

3.2 Steering motor model

The Lego DC motor is used as the steering motor. Figure 4 shows the Simulink blocks used to drive the steer motor. It includes compensation for friction and battery voltage levels. V represents the voltage applied to the DC motor.

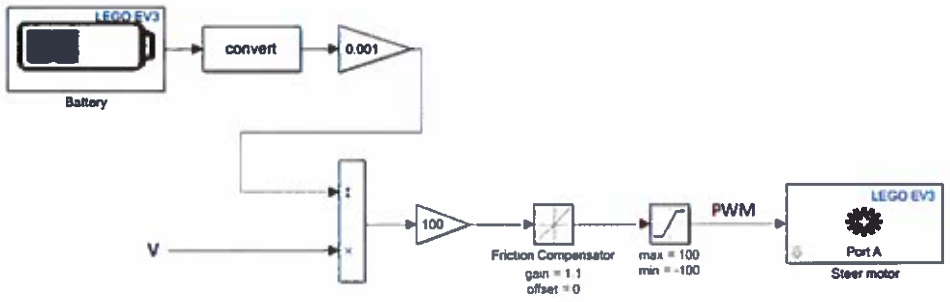


Figure 4: Steer voltage

The physics of a DC motor requires a second order relationship between input voltage and output angle. This relationship was directly estimated through experiments. A step sequence and chirp voltage signal was applied to the motor and the steer angle recorded. Matlab's System Identification toolbox was then used to find the best-fit second order model.

$$C(s) = \frac{\delta(s)}{V(s)} = \frac{1183}{s^2 + 9.253s + 0.01743} \tag{2}$$

This model provides a reasonable fit as shown in the plots below. However, the steer angle showed some error with the chirp input that cannot be explained by the linear model. The reason for this is possibly due to friction and is left as modelling inaccuracy.

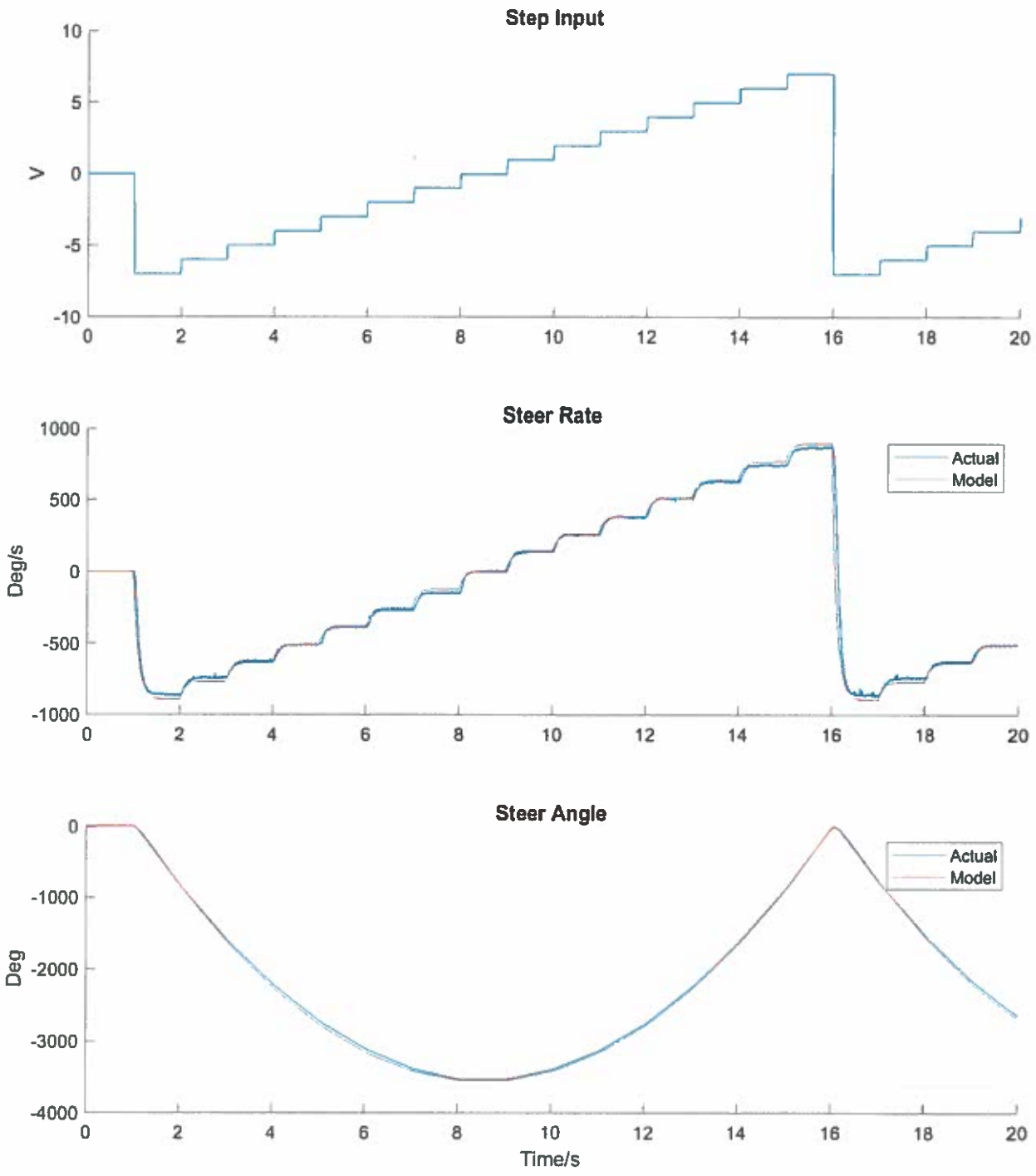


Figure 5: Response to a step stair input

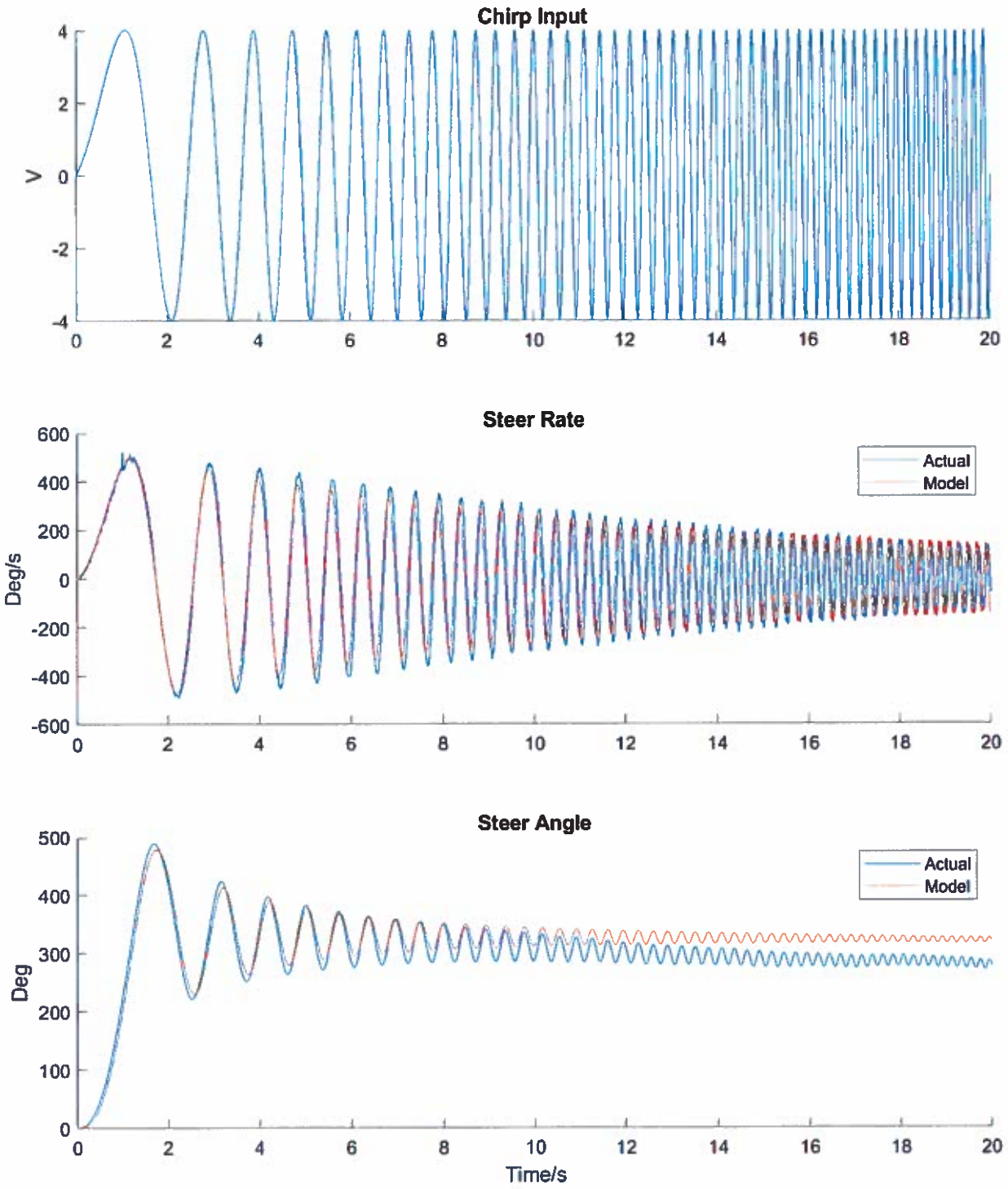


Figure 6: Response to a chirp input

3.3 PD Controller design

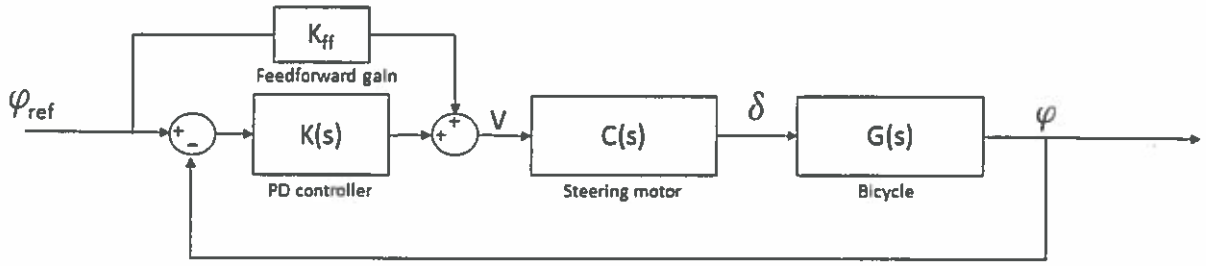


Figure 7: PD controller block diagram

The control method is shown in Figure 7. The PD controller $K(s)$ was first designed to stabilize the bicycle before the feed-forward gain K_{ff} was added to enable turning. Turning is achieved by making the bicycle track a non-zero reference lean angle φ_{ref} in the direction of the desired steer.

The complete system transfer functions for the bicycle $G(s)$ and the steer motor $C(s)$ are given by equations 1 and 2. Looking at the root locus plot of $C(s)G(s)$, the system cannot be stabilized by only proportional or derivative control since a pole always remains in the RHP for all gains (Figure 8).

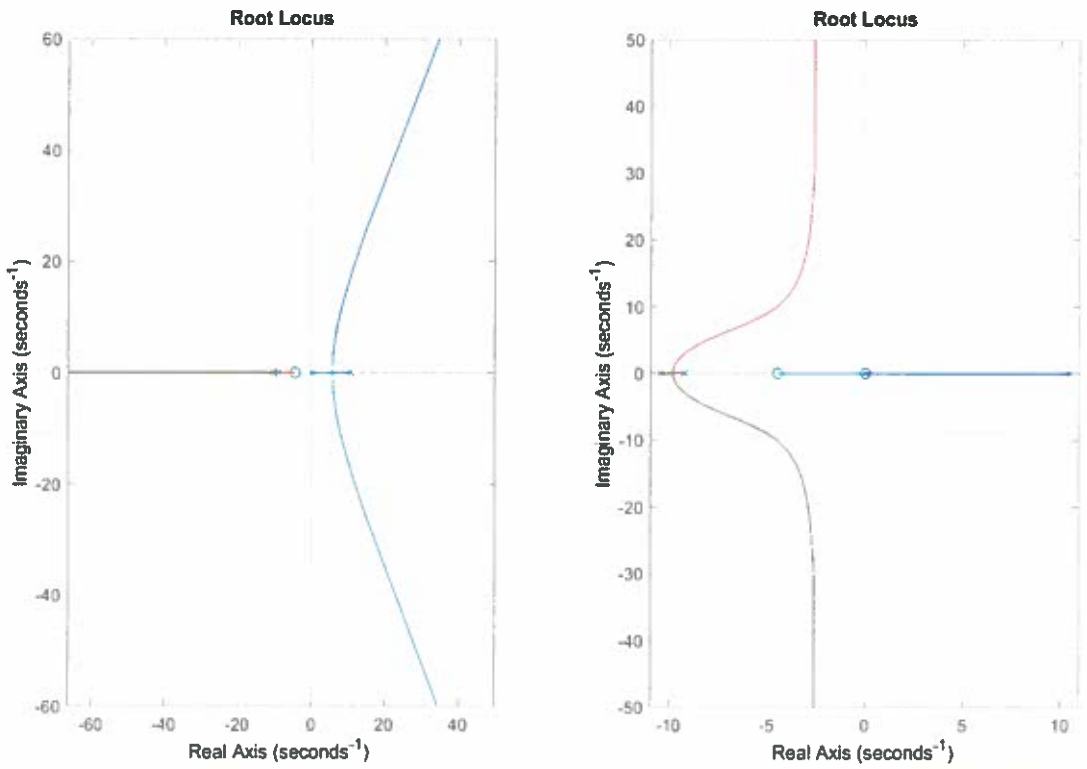


Figure 8: Root locus of proportional control only (left) and derivative control only (right). Unstable for all gains.

However, PD control can be used to pull the root locus into the LHP.

$$V = k_p(\varphi_{\text{ref}} - \varphi) + k_d\dot{\varphi}$$
$$K(s) = \frac{V(s)}{\varphi(s)} = k_p + k_d s$$

$k_p = -0.1$ and $k_d = -9$ were chosen via experimental tuning. With these parameters, the bode plot show that the closed loop poles can be made to all reside in the LHP, achieving closed loop stability (Figure 9). $k_{\text{ff}} = -4$ was chosen to give reliable turning.

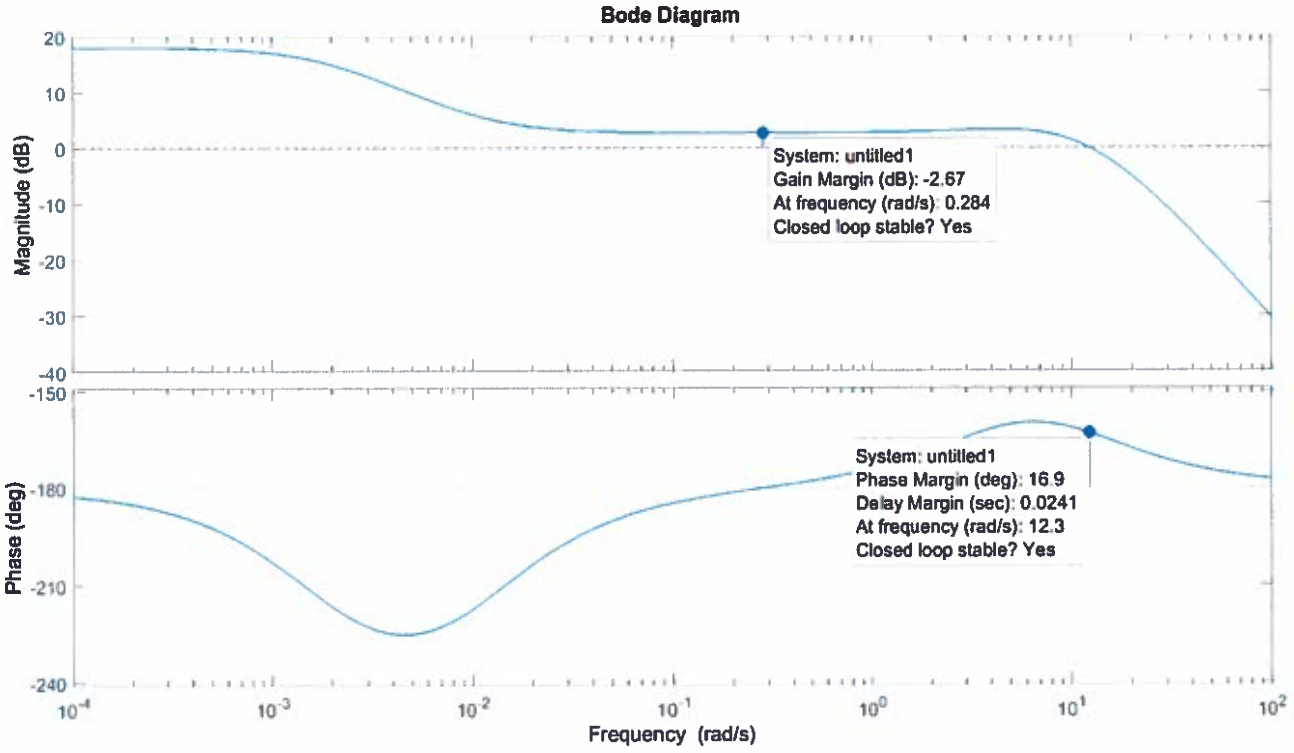


Figure 9: Bode plot for $K(s) = -9s - 0.1$

3.4 Kalman Filter

Both the lean angle φ and lean rate $\dot{\varphi}$ were needed to implement the PD feedback controller designed in section 3.3. While the gyro sensor measures lean rate, the lean angle cannot be read directly. Calculating φ by integrating the gyro readings resulted in unacceptable drift.

Initially, the complementary filter was used to remove drift. Lean angle was derived from 3-axis accelerometer measurements by finding the direction of the gravity vector. Although this gives accurate readings when the sensor is stationary, it is sensitive to centripetal acceleration and changes in speed of the bike. A complementary filter combines lean angle estimates from the gyro and the accelerometer to obtain a more accurate estimate of the lean angle. It does this by taking the low frequency components of accelerometer estimates and the high frequency readings from the gyro estimates. However, this method was not successful because the accelerometer readings were simply too noisy to be of any use.

Instead, a Kalman Filter was implemented as a state observer using only gyro readings. The system model $C(s)G(s)$ was first converted into state space form with state $x = [\varphi, \delta, \dot{\varphi}, \dot{\delta}]$ and observations $z = [\dot{\varphi}, \delta]$ to give the form

$$x_{k+1} = Ax_k + Bu_k + w_k$$
$$z_{k+1} = Cx_{k+1} + n_{k+1}$$

where w_k is the model uncertainty and n_k is the sensor noise. Gyro noise was characterized by reading the gyro sensor when it was stationary. The variance was measured to be around 7.8640×10^{-4} $\text{rad}^2\,\text{s}^{-2}$ about a mean of -0.0142$\text{rad}\,\text{s}^{-1}$. Compared to the expected size of lean rate measurements on the Lego bicycle, gyro noise is practically negligible. The encoder quantization error variance was also calculated to be 6.3452×10^{-6} rad^2. Therefore, n_k was taken to be 0 mean with covariance matrix $R = \texttt{diag}([7.8640 \times 10^{-4}, 6.3452 \times 10^{-6}])$. w_k was taken to be 0 mean with covariance matrix $Q = \texttt{diag}([10^{-8}, 10^{-5}, 10^{-2}, 10^{-7}])$.

The Kalman Filter uses the provided model, control input and observed sensor measurements to produce an estimate of the system's true state. This is better than using any one method alone. It is an iterative algorithm comprising of a predict and update stage. The predict stage uses the state estimate from the previous time-step to produce an estimate of the state at the current time-step. The update stage then combines this prediction with the current sensor readings, with the weights determined by the Kalman gain K. At each time-step, the optimal Kalman gain K has to be calculated.

Implementation of the Kalman Filter is shown below.

```
%Predict
x_kkprev= A*x_kprevkprev + B*u_kprev; %Predict current state x[k|k-1]
P_kkprev= A*P_kprevkprev*A'+ Q;  %Covariance of prediction P[k|k-1]

%Update
error_k= z_k - C*x_kkprev;  %error between sensor and estimation
cov_error_k= C*P_kkprev*C' + R; %Covariance of observation error
K= P_kkprev*C'*inv(cov_error_k); %Kalman gain
x_kk=x_kkprev+K*error_k; %observer equation to get x[k|k]
P_kk=P_kkprev-K*C*P_kkprev; %covariance of updated state P[k|k]

%update variables for next iteration
x_kprevkprev=x_kk;
P_kprevkprev=P_kk;
```

The Kalman Filter is only started after 0.6s when the bicycle leaves the guide rails. Results from a typical run is shown in Figure 10. It successfully removes drift from gyro integration as well as noise from steer angle differentiation.

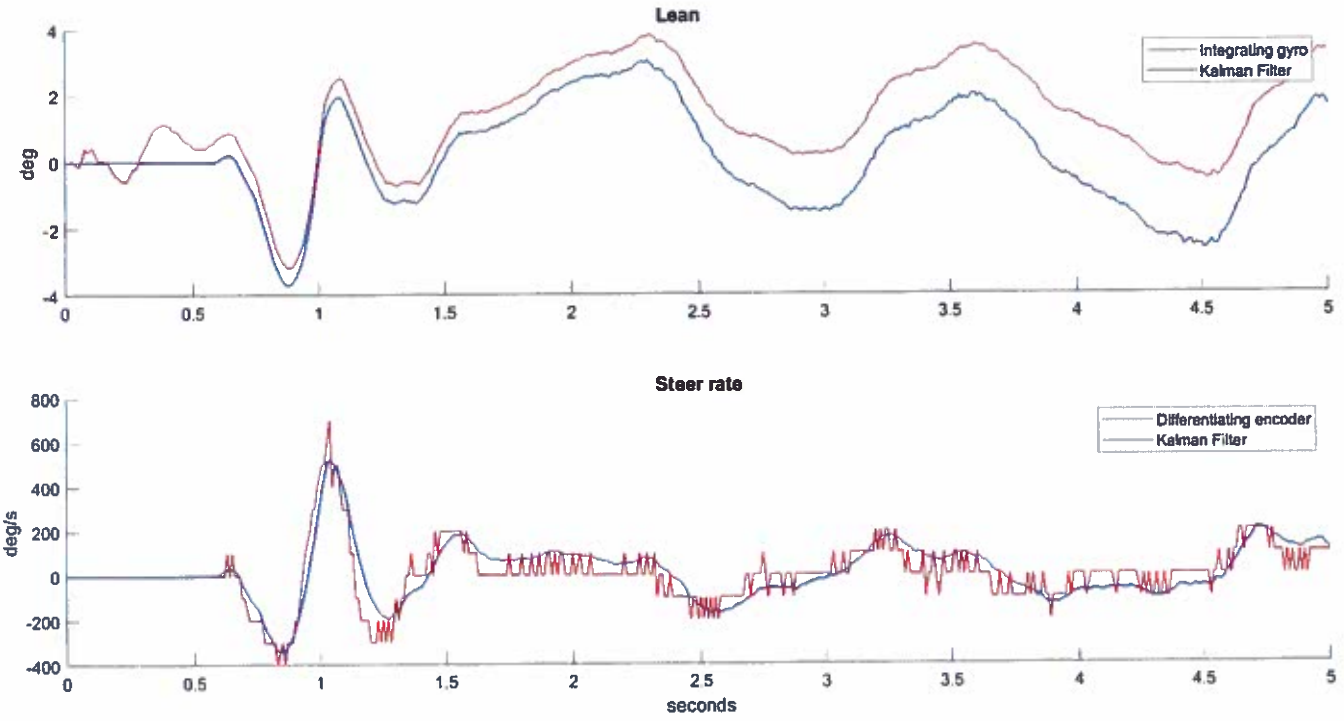


Figure 10: Kalman filter removes drift and noise

3.5 Drive controller

Design of the PID drive controller is not as critical. It only has to get the bicycle up to the desired reference speed and track it. The PID gains of $[0.08, 0.06, 0.01]$ were chosen experimentally. Voltage and friction compensation were used as with the steer motor, but they are not essential. A forward speed of $0.5\,\mathrm{m\,s^{-1}}$ translates to $715\,\mathrm{deg\,s^{-1}}$. The bicycle is observed to reach this speed within 0.6 s which is fast enough and before it exits the guide rails (Figure 12) .

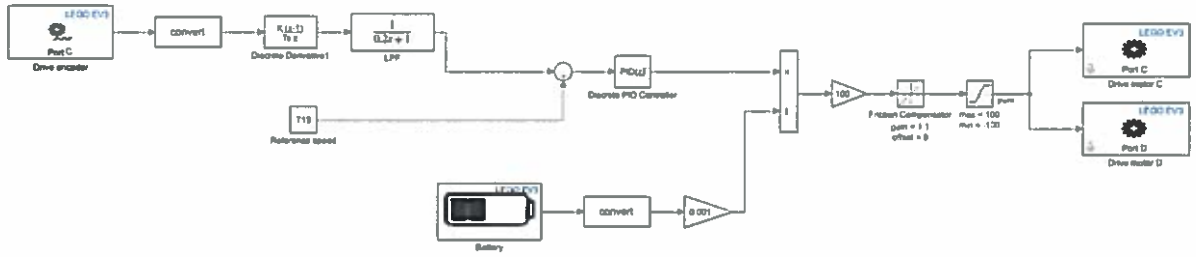


Figure 11: PID controller for drive motors

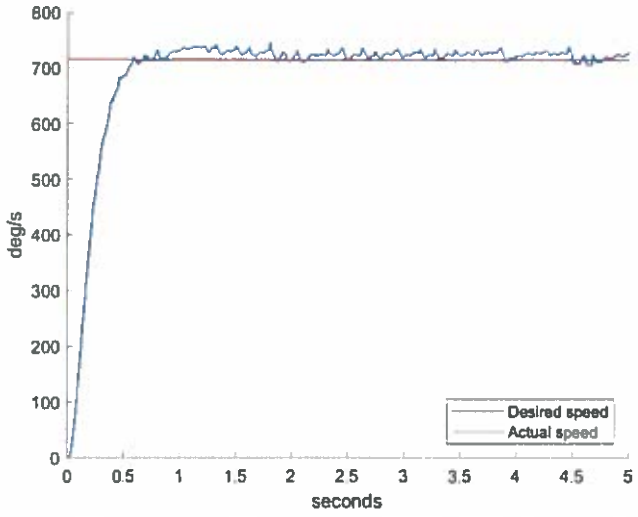


Figure 12: Tracking of desired reference drive speed

3.6 Experimental results

The full controller is shown in Figure 13. In addition to the components described above, it also has a timer that enables the controller and Kalman filter after 0.6 s. The program is terminated if the absolute steering angle exceeds 90 deg and the lean angle exceeds 60 deg. Results for a straight run ($\varphi_{\mathrm{ref}} = 0\,\mathrm{deg}$), right turn ($\varphi_{\mathrm{ref}} = 2\,\mathrm{deg}$) and left turn ($\varphi_{\mathrm{ref}} = -2\,\mathrm{deg}$) are shown. The bicycle trajectories were calculated from the steer angle and speed using Equation 9 in Appendix A.

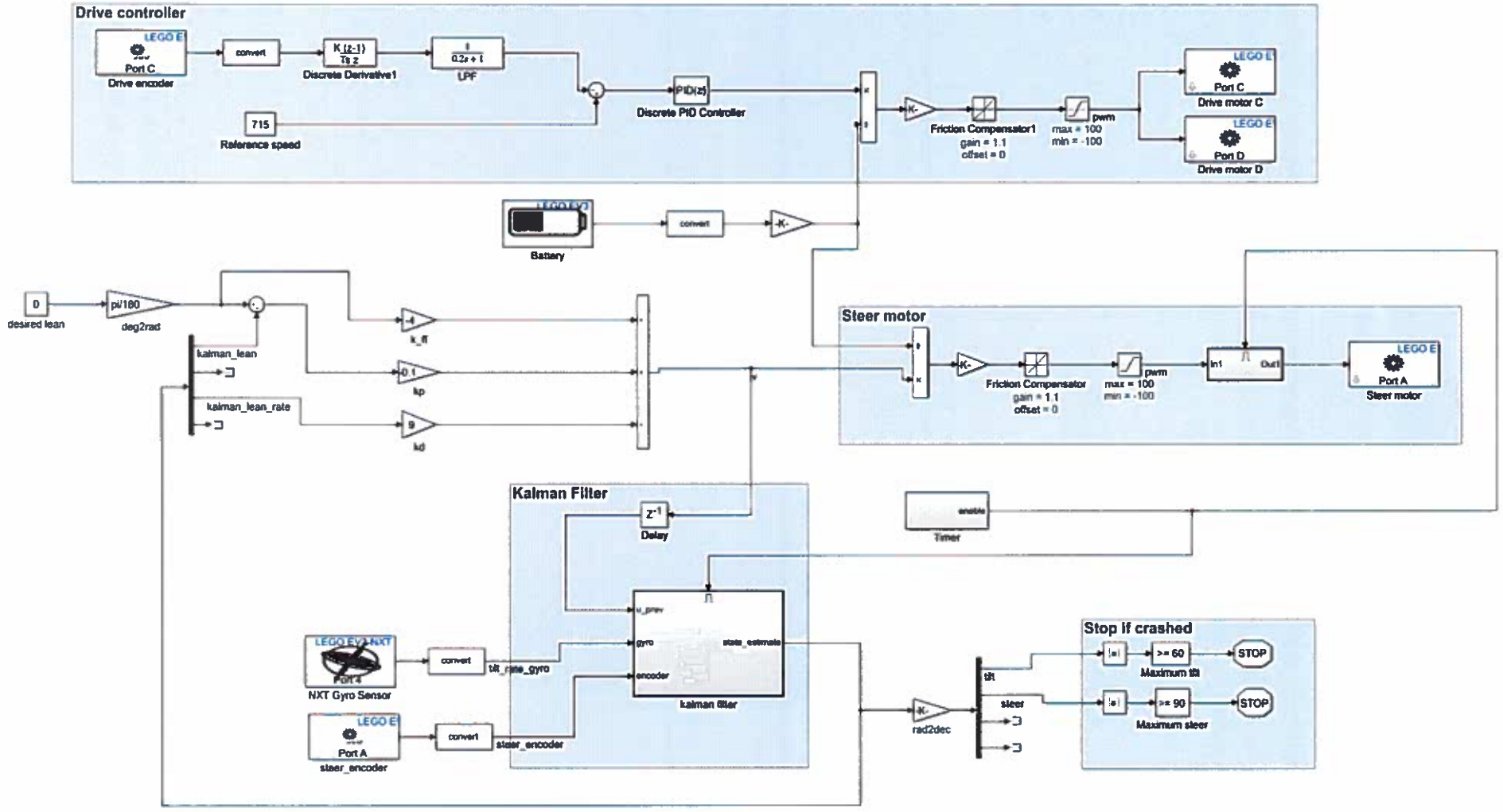


Figure 13: Full PD controller implemented in Simulink

3.6.1 Straight ahead

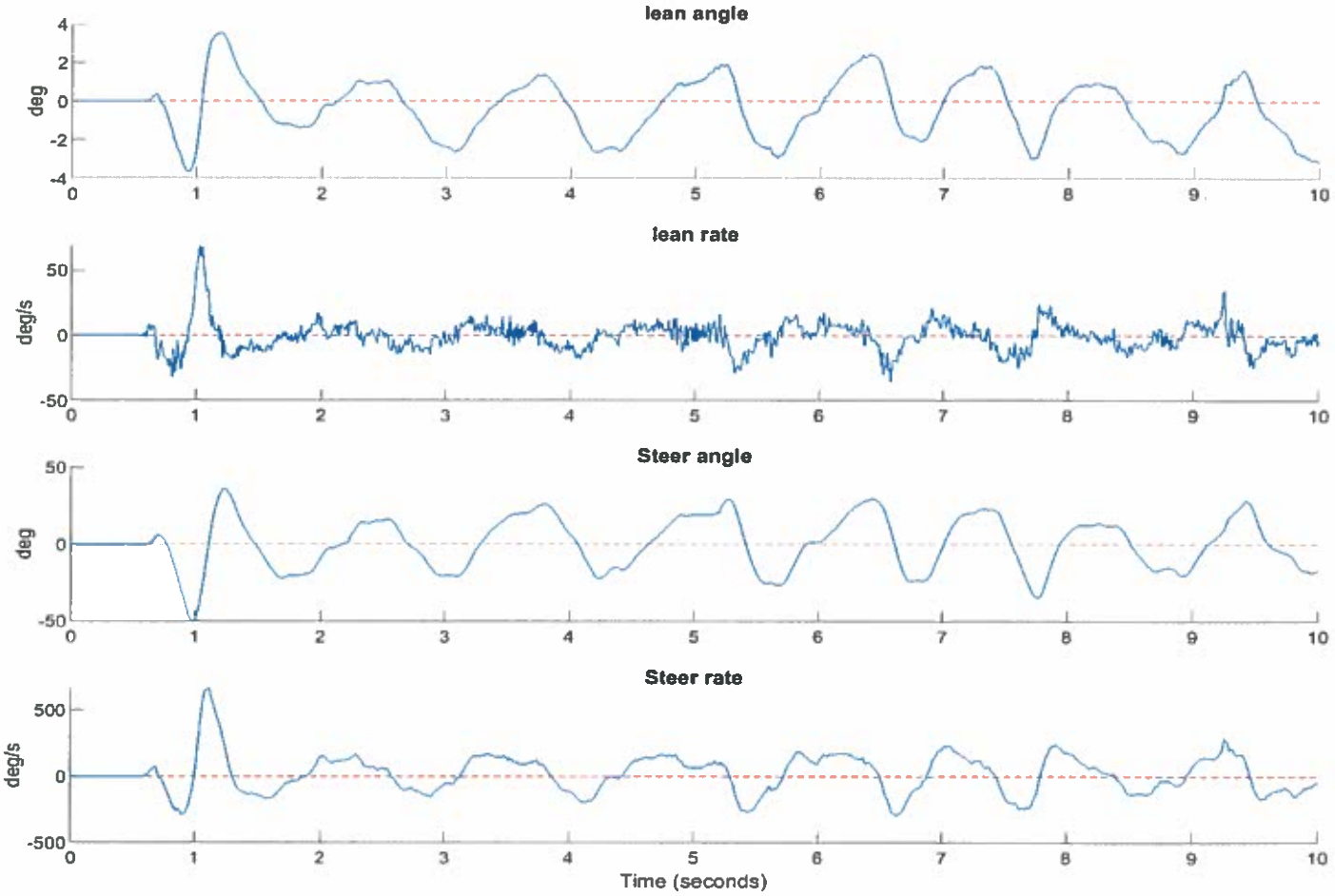


Figure 14: States during straight run

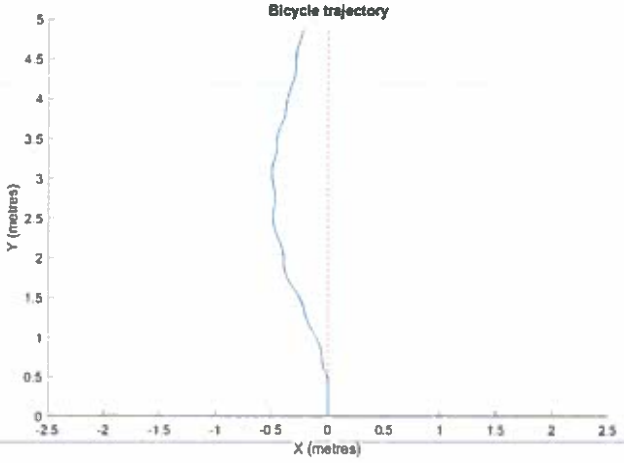


Figure 15: Path taken by bicycle during straight run

3.6.2 Right turn

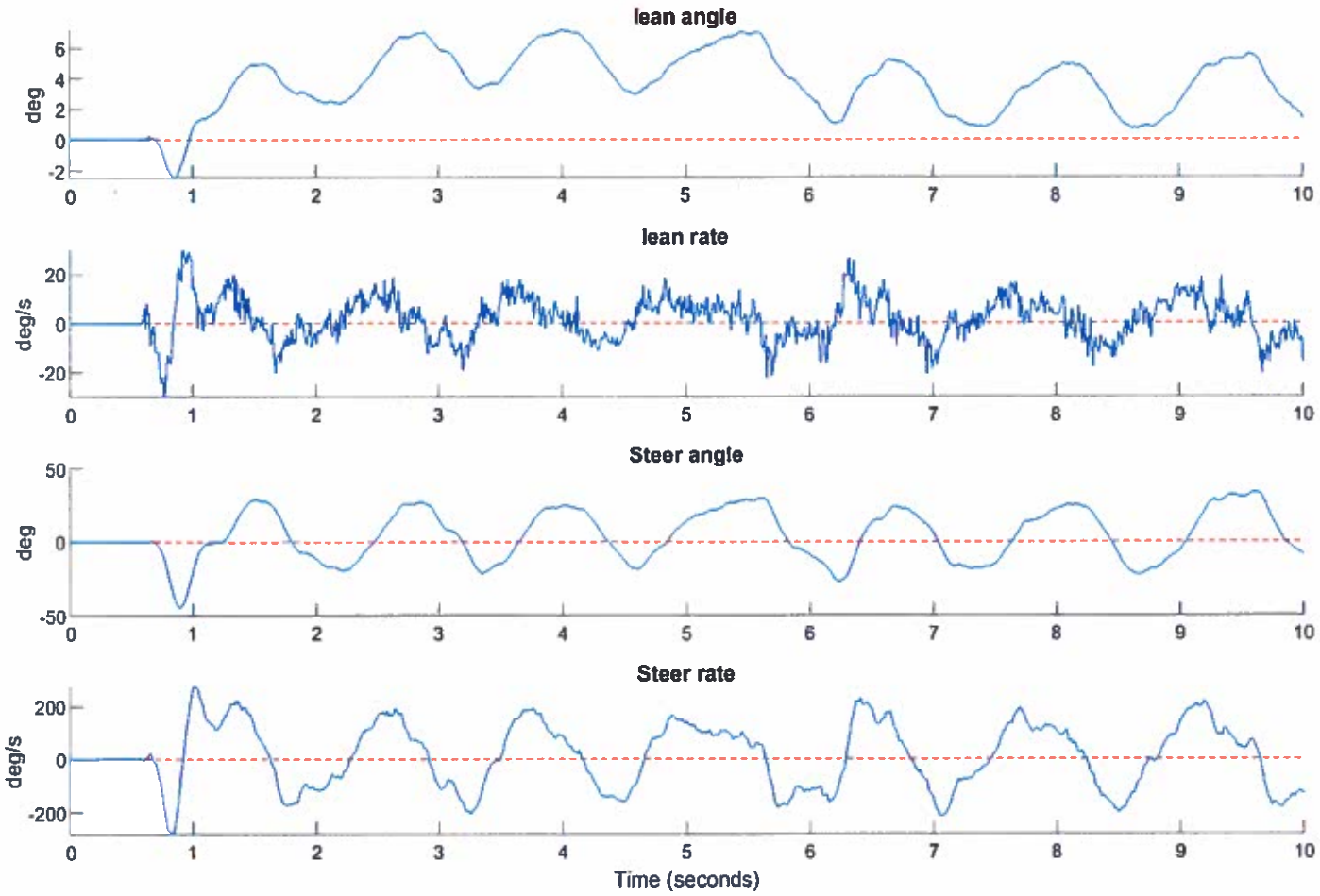


Figure 16: States during right turn

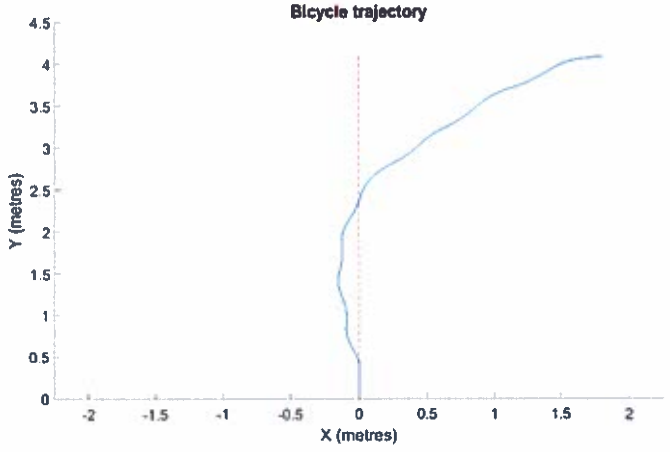


Figure 17: Path taken by bicycle during right turn

3.6.3 Left turn

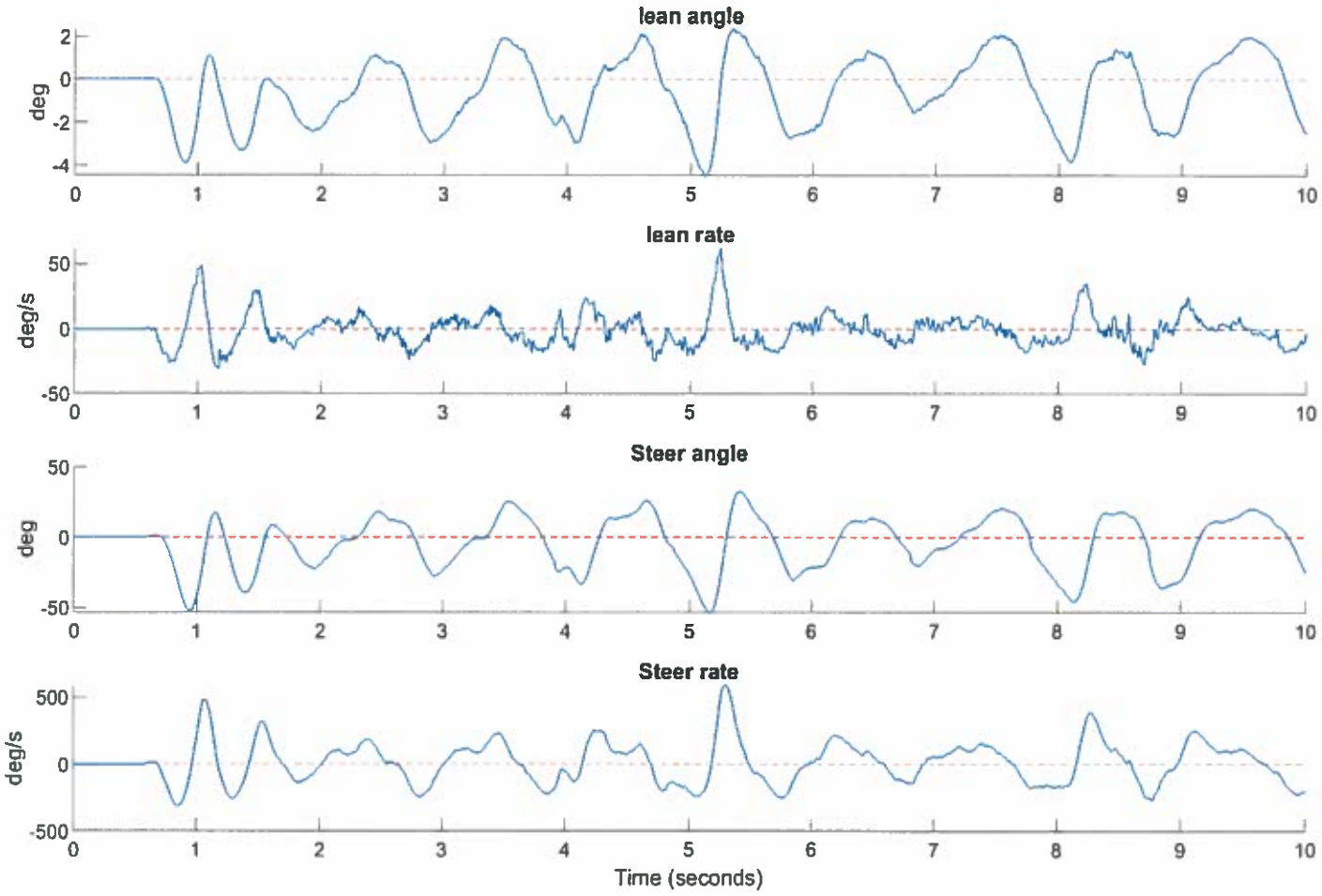


Figure 18: States during left turn

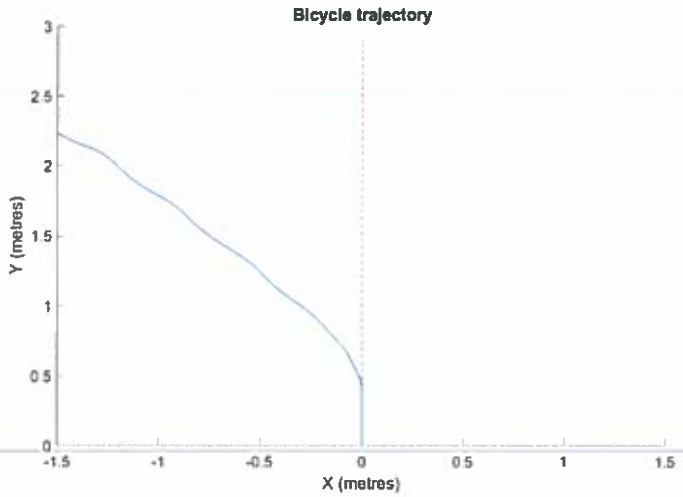


Figure 19: Path taken by bicycle during left turn

A working bicycle has been achieved. When no turn is demanded (Figures 14), the bicycle manages to stay upright with lean angle oscillating between ± 3 deg and steer angle between ± 25 deg. When a turn was demanded (Figures 16, 18), the bicycle tracked a non-zero average lean angle and turned in the correct direction. The largest lean rates and steer angles were observed when the bicycle just left the guide rails. This was where the bicycle was most likely to fail. A remote control was also built, enabling the trajectory of the bicycle to be controlled from a laptop.

3.7 Discussion

Playing with the feedback gains reveals deviation from theory. Derivative-only action over the range $k_d = 6.5$ to 12 was observed to stabilize the system even though this has been shown to be theoretically impossible in section 3.3. The designed controller also has very poor stability margins as seen in Figure 9 and was not expected to have worked in practice. This suggests modelling deficiencies.

Several aspects of the bicycle build could have contributed to modelling inaccuracies. The weight distribution of this Lego bicycle is 7g for the steering column, 646g for the frame and EV3, and 29g for each wheel. For a bike of this scale, most of the weight comes from the rear frame and the inverted pendulum model was expected to be sufficiently accurate. However, even when no turn was demanded, the bicycle showed a leftward drifting path because of it's asymmetric left-heavy build. The front steering column was also observed to flex a lot. More significantly, the rubber wheels on the Lego bicycle are actually quite thick with square edges. It is thus possible that the model used leaves out some important dynamics.

The limitations of classical PID for the bicycle problem were also observed. The small angle approximation will not hold for large steering angles. Therefore, although large inputs may be able to stabilize the bicycle based on the linearised model, it will tend to destabilize the system or hit the steering limits in reality. It is therefore very important to minimize the control effort while still trying to maximize the stability margins. There is no neat way to do this except through experimental tuning.

The Linear Quadratic Regulator (LQR) method will be a better solution for the bicycle stability problem. Given an accurate enough model of the physical system, it yields an optimal controller that minimizes a quadratic cost function which includes the control effort, while also having guaranteed phase and gain margins.

4 Lego Bicycle 2

The LQR controller will be explored in this section. Several changes were made to the previous bicycle build to address problems with Lego Bicycle 1. Firstly, supporting members were added to the steering axis to reduce the unwanted flex. The frame was also made more symmetric so that the weight distribution of the bicycle is more balanced. These changes will reduce the difference between the model and the physical bicycle.

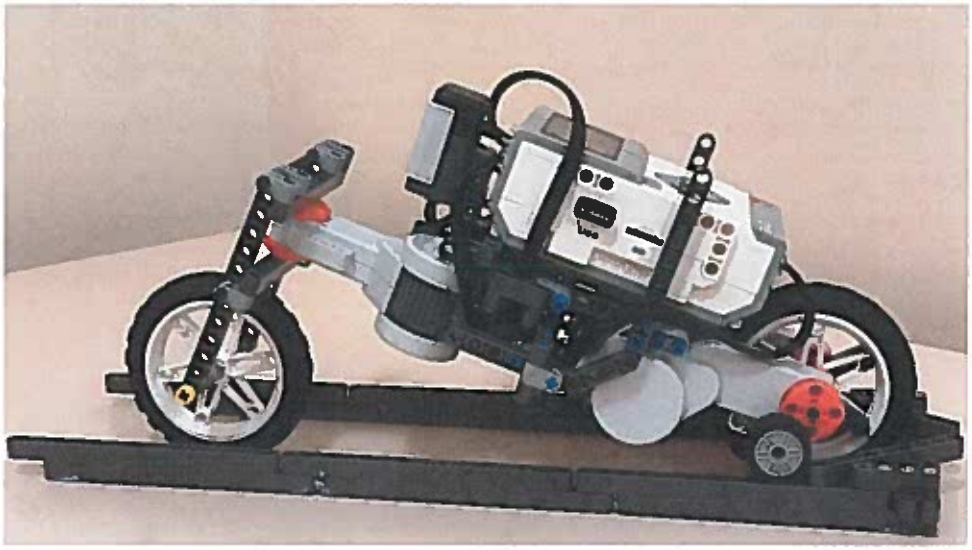
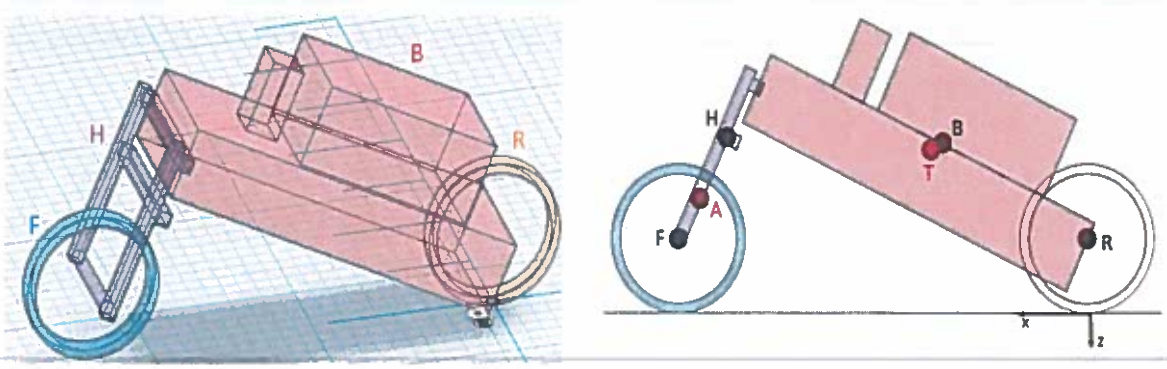


Figure 20: Lego bicycle 2

4.1 Dynamics modelling - Whipple model

A more complicated bicycle model was used. The Whipple bicycle model derived by Schwab *et al.* is shown in Appendix B, in which the bicycle is treated as 4 connected rigid bodies (Handlebar assembly H, main body frame B, front wheel F and rear wheel R). The moments of inertia and centre of mass of each component were found by further decomposing the bicycle into simple shapes (Figure 21a) and using the parallel axis theorem. The rear wheel ground contact point is taken as the origin.



(a) Combination of simple shapes (b) Component CG positions

Figure 21: Simplified model of bicycle used to calculate parameters

The position of the centre of gravity of these elements, as well as the combined steering column A=H+F and the combined bicycle T=H+F+B+R are shown in Figure 21b. Calculated parameters are shown in the tables below.

B (EV3,Gyro sensor,frame)		H (Handlebar assembly)		R/F (Wheels)	
m_B	0.648	m_H	0.0230	m_R/m_F	0.029
x_B	0.0868	x_H	0.2170	$r1$ (inner radius)	0.03
z_B	-0.0915	z_H	-0.0891	$r2$ (outer radius)	0.04
I_{Bxx}	8.1867e-04	I_{Hxx}	3.2714e-05	t (thickness)	0.01
I_{Bxz}	4.3943e-04	I_{Hxz}	-9.2895e-06	I_{Fxx}/I_{Rxx}	1.8367e-05
I_{Bzz}	0.0018	I_{Hzz}	2.1425e-05	I_{Fzz}/I_{Rzz}	1.8367e-05
I_{Byy}	0.0021	I_{Hyy}	2.2293e-05	I_{Fyy}/I_{Ryy}	3.6250e-05

T (Combined bicycle)		A (Combined steering axis)		General parameters	
m_T	0.729	m_A	0.0520	Wheel base w	0.245
x_T	0.0938	x_A	0.2326	Trail c	0.016
z_T	-0.0873	z_A	-0.0617	Steering axis angle λ	25°
I_{Txx}	0.0066	I_{Axx}	8.2063e-05	Speed v	0.5
I_{Txz}	0.0063	I_{Axz}	-2.6907e-05		
I_{Tzz}	0.0096	I_{Azz}	4.9813e-05		

Table 2: Bicycle parameters
(More clearly defined in Appendix B)

With steer angle δ and lean angle φ both rightward positive, substituting in the parameters above gives the bicycle equations of motion in matrix form

$$\begin{bmatrix} 0.0066 & 3.8490\text{e-}04 \\ 3.8490\text{e-}04 & 7.3093\text{e-}05 \end{bmatrix} \begin{bmatrix} \ddot{\varphi} \\ \ddot{\delta} \end{bmatrix} + \begin{bmatrix} 0 & 0.0140 \\ -4.6431\text{e-}04 & 0.0012 \end{bmatrix} \begin{bmatrix} \dot{\varphi} \\ \dot{\delta} \end{bmatrix} + \begin{bmatrix} -0.6246 & 0.0207 \\ -0.0399 & -0.0127 \end{bmatrix} \begin{bmatrix} \varphi \\ \delta \end{bmatrix} = \begin{bmatrix} T_\varphi \\ T_\delta \end{bmatrix}$$

$$\boldsymbol{M\ddot{q} + C\dot{q} + Kq = f}$$

where T_φ is some disturbance lean torque on the bicycle and T_δ is some disturbance steer torque on the handlebars. These matrix values are very similar to those obtained in [12] and should be reasonably accurate.

The above matrix equation consists of the lean (top) and steer (bottom) equations. They are derived by considering conservation of angular momentum about the lean and steer axis respectively. Because a steer angle PID controller will be designed to directly control δ (Section 4.2), the steer equation above can be ignored. The remaining lean equation is given by

$$M_{11}\ddot{\varphi} + M_{12}\ddot{\delta} + C_{11}\dot{\varphi} + C_{12}\dot{\delta} + K_{11}\varphi + K_{12}\delta = T_\varphi \tag{3}$$

4.2 Steering controller model

Modelling of the DC motor in Section 3.2 showed unexplained errors with the chirp signal. A more precise modelling of the steering can be achieved by first designing a PID controller around the DC motor that tracks a desired reference steer angle δ_{ref}, which is then directly controlled. Feedback gains $k_p = 0.45$, $k_d = 0.01$, $k_i = 0.1$ were used to give a rise time of around 0.1 s with no oscillations and negligible steady state error.

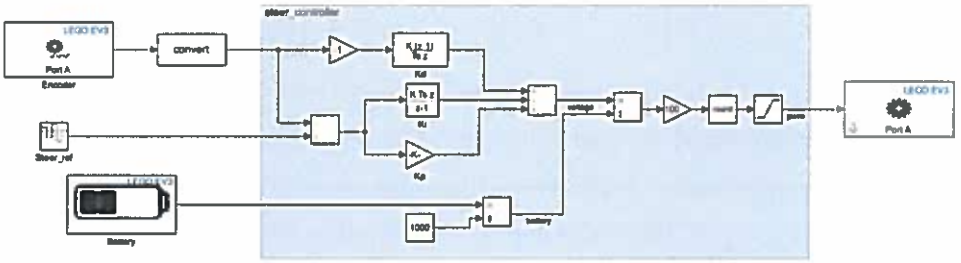


Figure 22: PID steer controller to track reference steer angle

A second order model was then fitted to the step response. This model was checked by putting in a periodic step signal and a chirp signal. It fits the actual observed steer angle well.

$$\ddot{\delta} + P_1\dot{\delta} + P_2\delta = k\delta_{\text{ref}} \tag{4}$$

where $P_1 = 40.02$, $P_2 = 621.5$, $k = 621.5$.

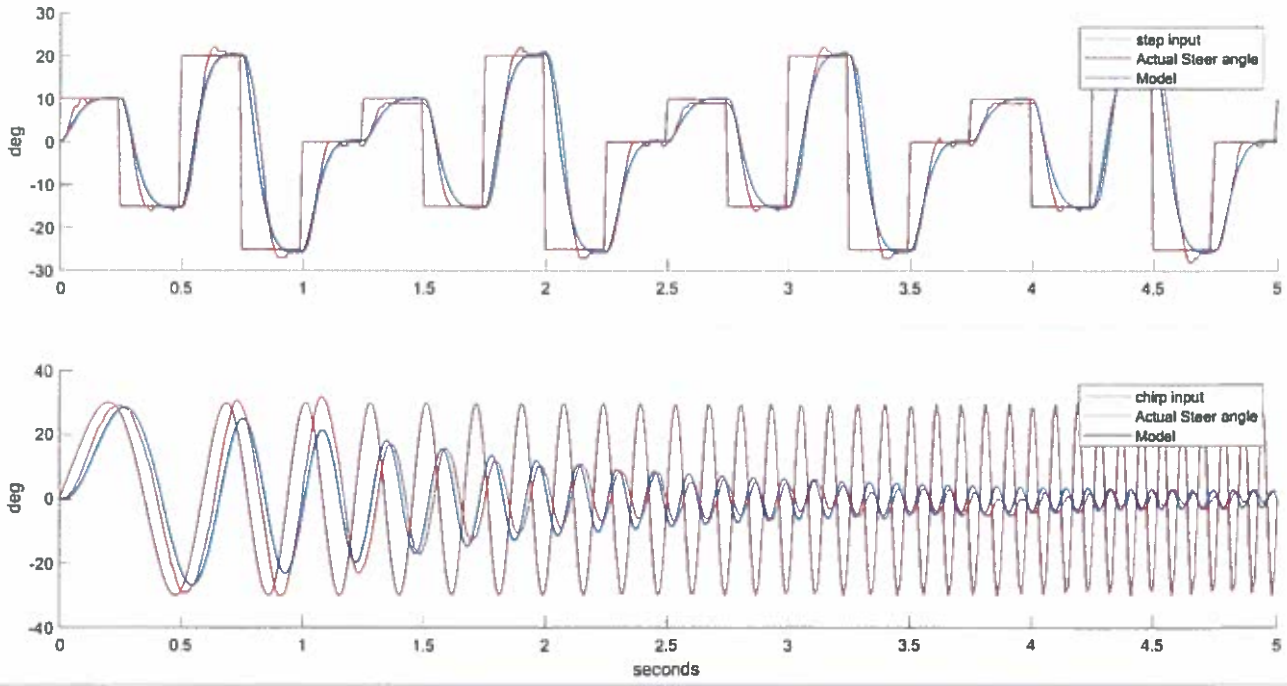


Figure 23: Actual and model response to step input (top) and chirp input (bottom)

4.3 Full bicycle dynamics

Equations 3 and 4 represent the full bicycle dynamics. In matrix form,

$$\begin{bmatrix} M_{11} & M_{12} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \ddot{\varphi} \\ \ddot{\delta} \end{bmatrix} + \begin{bmatrix} C_{11} & C_{12} \\ 0 & P1 \end{bmatrix} \begin{bmatrix} \dot{\varphi} \\ \dot{\delta} \end{bmatrix} + \begin{bmatrix} K_{11} & K_{12} \\ 0 & P2 \end{bmatrix} \begin{bmatrix} \varphi \\ \delta \end{bmatrix} = \begin{bmatrix} T_\varphi \\ k\delta_{\text{ref}} \end{bmatrix}$$

$$\boldsymbol{M} \begin{bmatrix} \ddot{\varphi} \\ \ddot{\delta} \end{bmatrix} + \boldsymbol{C} \begin{bmatrix} \dot{\varphi} \\ \dot{\delta} \end{bmatrix} + \boldsymbol{K} \begin{bmatrix} \varphi \\ \delta \end{bmatrix} = \begin{bmatrix} T_\varphi \\ k\delta_{\text{ref}} \end{bmatrix} \tag{5}$$

This equation can be expressed in state space form. As with Lego Bicycle 1, a gyro sensor and a steer encoder are used to read the tilt rate and steer angle of the bicycle. Both were modelled as having some Additive White Gaussian Noise (AWGN) represented by variables n_{gyro} and n_{encoder}. The variance of these noise are as calculated in section 3.4. Taking the state $x = [\varphi; \delta; \dot{\varphi}; \dot{\delta}]$, observations $y = [\dot{\varphi}; \delta]$, input $u = \delta_{ref}$ and disturbance $v = T_\varphi$, the continuous state space form of the system dynamics is given by

$$\begin{bmatrix} \dot{\varphi} \\ \dot{\delta} \\ \ddot{\varphi} \\ \ddot{\delta} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\boldsymbol{M}^{-1}\boldsymbol{K} & & -\boldsymbol{M}^{-1}\boldsymbol{C} & \end{bmatrix} \begin{bmatrix} \varphi \\ \delta \\ \dot{\varphi} \\ \dot{\delta} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \boldsymbol{M}^{-1} \begin{bmatrix} 0 \\ k \end{bmatrix} \end{bmatrix} \delta_{\text{ref}} + \begin{bmatrix} 0 \\ 0 \\ \boldsymbol{M}^{-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \end{bmatrix} T_\varphi$$

$$\begin{bmatrix} \dot{\varphi} \\ \delta \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \varphi \\ \delta \\ \dot{\varphi} \\ \dot{\delta} \end{bmatrix} + \begin{bmatrix} n_{\text{gyro}} \\ n_{\text{encoder}} \end{bmatrix}$$

$$\dot{x} = \tilde{A}x + \tilde{B}u + \tilde{G}v$$
$$y = \tilde{C}x + n$$

The discrete state space form was then obtained by using the sample time of $T_s = 0.01s$ and using zero-order hold. This was done easily using the c2d command in Matlab. The obtained matrices are shown below

$$x_{k+1} = Ax_k + Bu_k + Gv_k$$
$$y_k = Cx_k + n_k$$

$$A = \begin{bmatrix} 1.0047 & 0.0016 & 0.0100 & 1.4217e-05 \\ 0 & 0.9728 & 0 & 0.0082 \\ 0.9493 & 0.3233 & 1.0047 & 0.0032 \\ 0 & -5.0692 & 0 & 0.6464 \end{bmatrix}, \quad B = \begin{bmatrix} -0.0018 \\ 0.0272 \\ -0.3547 \\ 5.0692 \end{bmatrix}, \quad G = \begin{bmatrix} 0.0076 \\ 0 \\ 1.5199 \\ 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

4.4 Model validation

As the system is open loop unstable, model validation for the bicycle is challenging. With no feedback, the bicycle falls over very quickly once it leaves the guide rails. This gives only a small window of time where a step input to δ_{ref} can be applied and its effect on the system's states recorded. The actual and model behaviour were then compared. Results are shown in Figure 24. The bicycle was allowed 0.6 s to get up to speed along the guide rails, after which a ± 10 deg step steer angle was demanded.

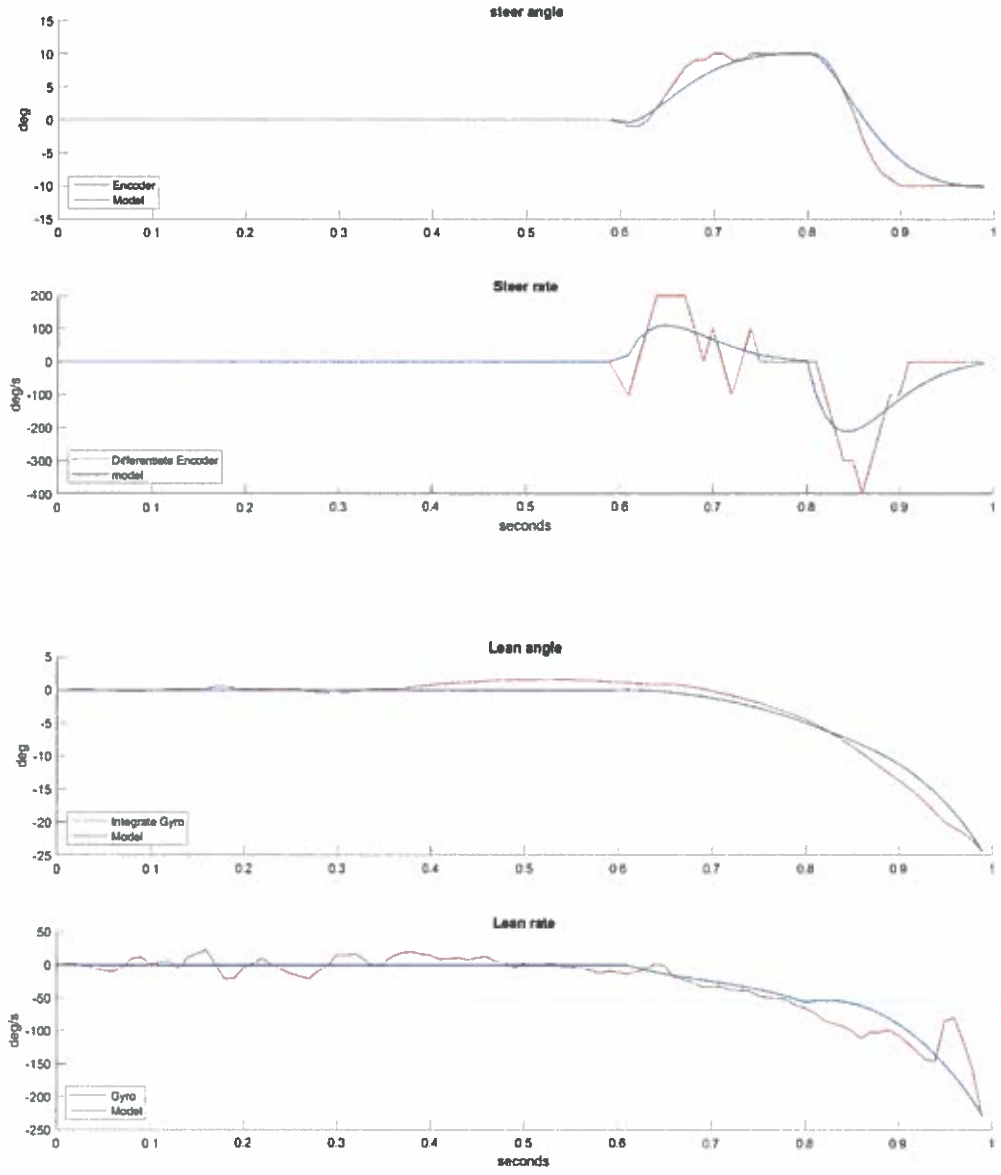


Figure 24: Comparison between model and recorded states

The bicycle falls over within 0.4 s of leaving the rails and the model's predictions matches the recorded data reasonably well. The spike in tilt rate at around 0.96 s was due to the the guide rollers hitting the ground and should be ignored. Based on this simple test, it would appear that the model used is good enough.

4.5 LQR controller design

The discrete time state space form of the bicycle system was derived in section 4.3. It was checked to be both controllable and observable. A stabilizing full state negative feedback controller that minimises control effort is desired. This is equivalent to minimising the cost function J that has quadratic penalty on both control effort and state deviation from the 0 state.

$$u_k = -Kx_k$$

$$J = \sum_{k=0}^{\infty} x_k^T Q x_k + u_k^T R u_k$$

Q and R are the weight matrices that trades off regulation performance and control effort. The optimal feedback gain K that minimizes this cost function is given by

$$K = (R + B^T P B)^{-1}(B^T P A)$$

where P is the unique solution to the Discrete-time Algebraic Riccati Equation (DARE).

$$A^T P A - P - A^T P B(B^T P B + R)^{-1} B^T P A + Q = 0$$

Obtaining this solution was done easily using Matlab's `dlqr` command. The LQR controller obtained in this fashion will be robust with guaranteed stability margins. As with Lego bicycle 1, the Kalman filter designed in Section 3.4 was used to estimate the states for feedback. However, this turns it into a Linear Quadratic Gaussian (LQG) regulator, for which there are no stability guarantees. The control scheme is shown in Figure 25.

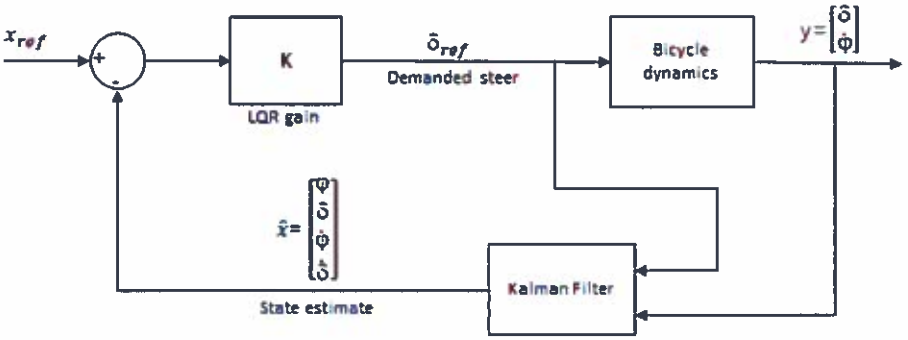


Figure 25: LQG control

The task of controller design now becomes that of tuning the weight matrices Q and R of both the LQR controller and the Kalman Filter $Q_{lqr}, R_{lqr}, Q_{KF}, R_{KF}$. Since a lean angle of ± 3 deg, lean rate of ± 30 deg/s, steer angle of ± 30 deg and steer rate of ± 200deg/s are considered normal, the inverse square of these values are used to give $Q_{lqr} = \texttt{diag}([364.76, 3.6476, 3.6476, 0.0821])$ and $R_{lqr} = 100$. The feedback gain matrix was then calculated to be $K = [-11.4201, -1.5268, -1.1724, -0.0325]$.

4.6 Simulation results

Simulations were run in Simulink to check that the system is closed loop stable. To simulate the unstable system in real life, torque disturbances and sensor noise were added in the form of white noise.

4.6.1 Straight ahead

With the desired state $x_{\text{ref}} = [0; 0; 0; 0]$, Larger torque impulses were added at $t = 2\,\text{s}$ (leftward) and $4\,\text{s}$(rightward) to simulate a push on the bicycle. The controller was able to recover quickly and remain upright. The bicycle trajectory shows the bicycle steering sharply in the direction of the applied impulse. The system also remained stable when the magnitude of the white noise disturbances were increased by $10\times$, showing robustness.

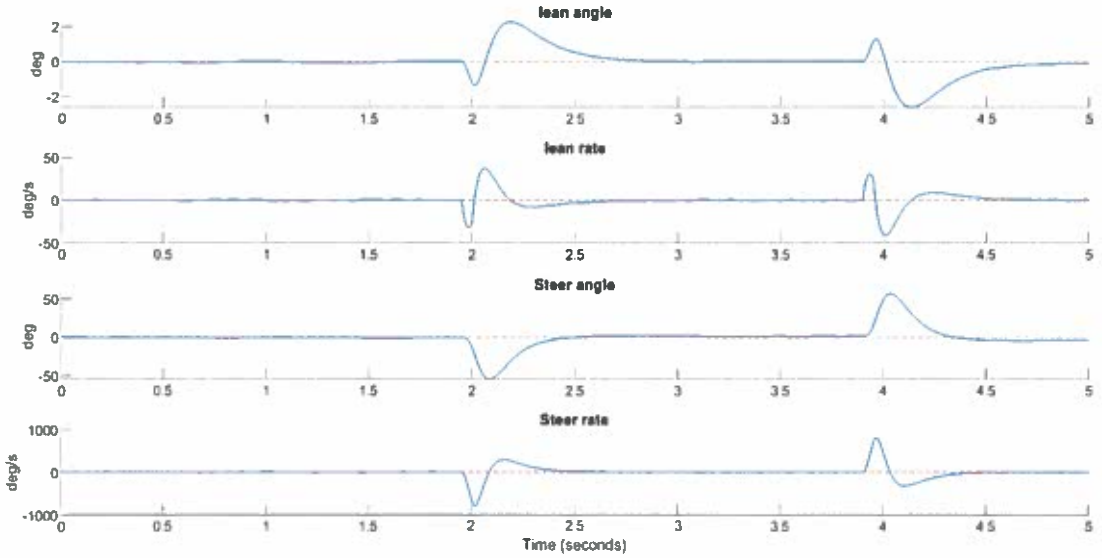


Figure 26: System states

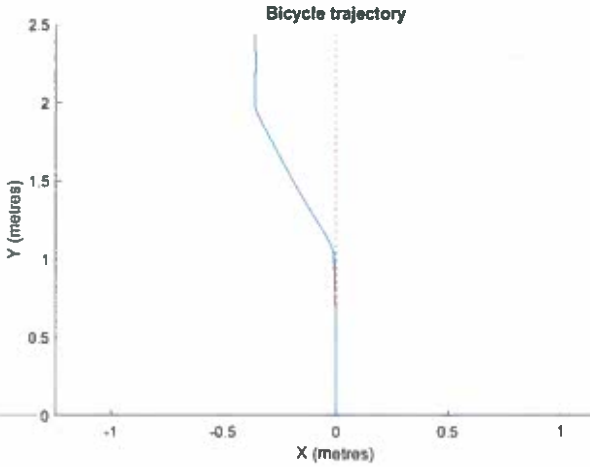


Figure 27: Bicycle trajectory

4.6.2 Turning

The bicycle can be made to turn left by using $x_{\text{ref}} = [0; -2; 0; 0]$. The controller initiates the turn by first counter-steering to the right, causing the bicycle to tilt to the left (Figure 28). This non-minimum phase steering behaviour is expected.

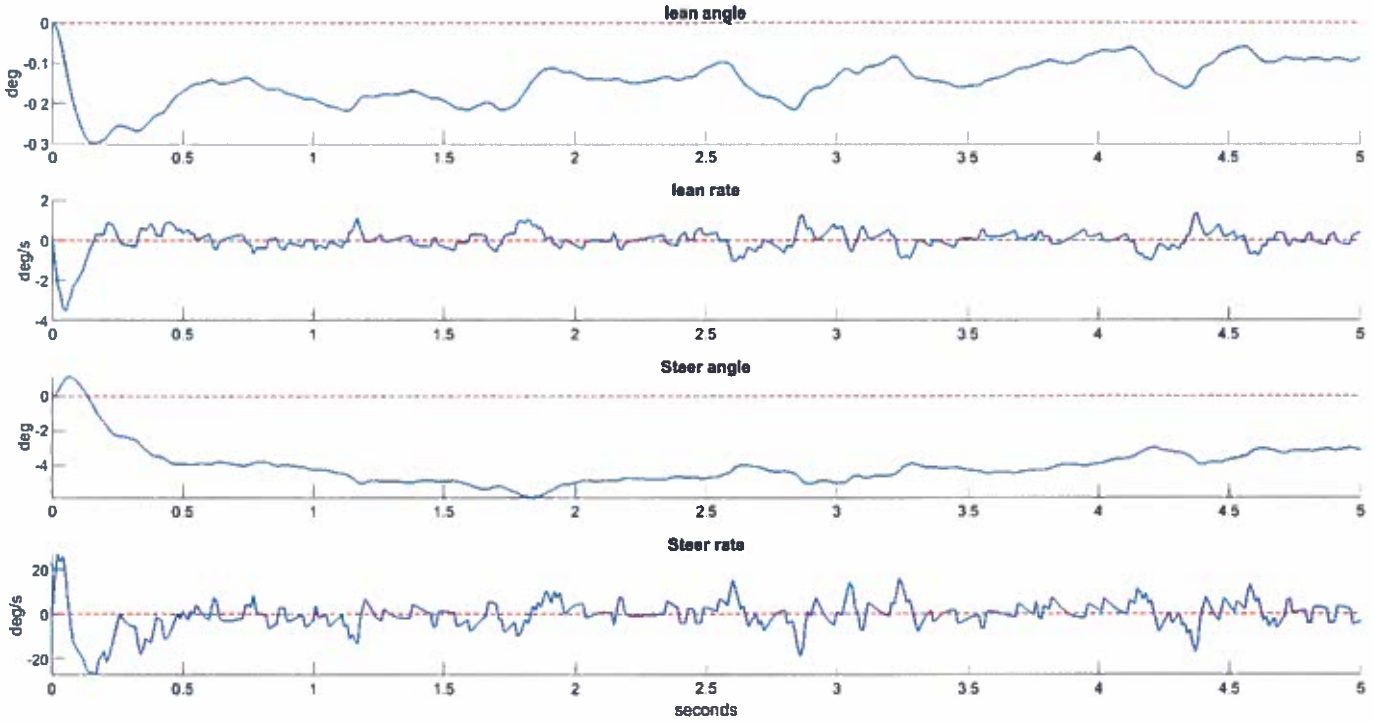


Figure 28: System states

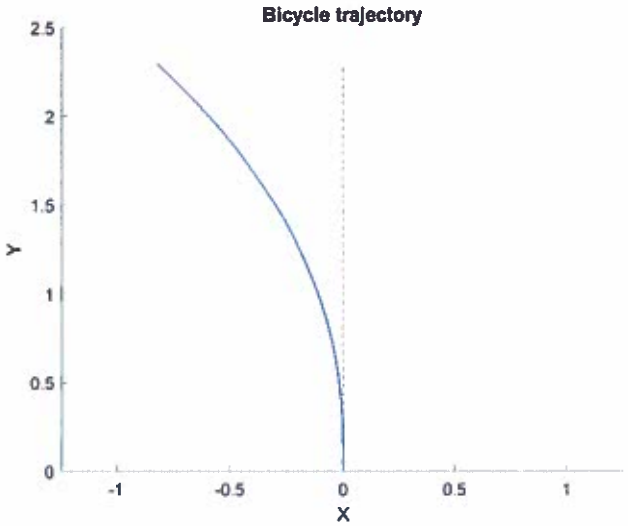


Figure 29: Bicycle trajectory

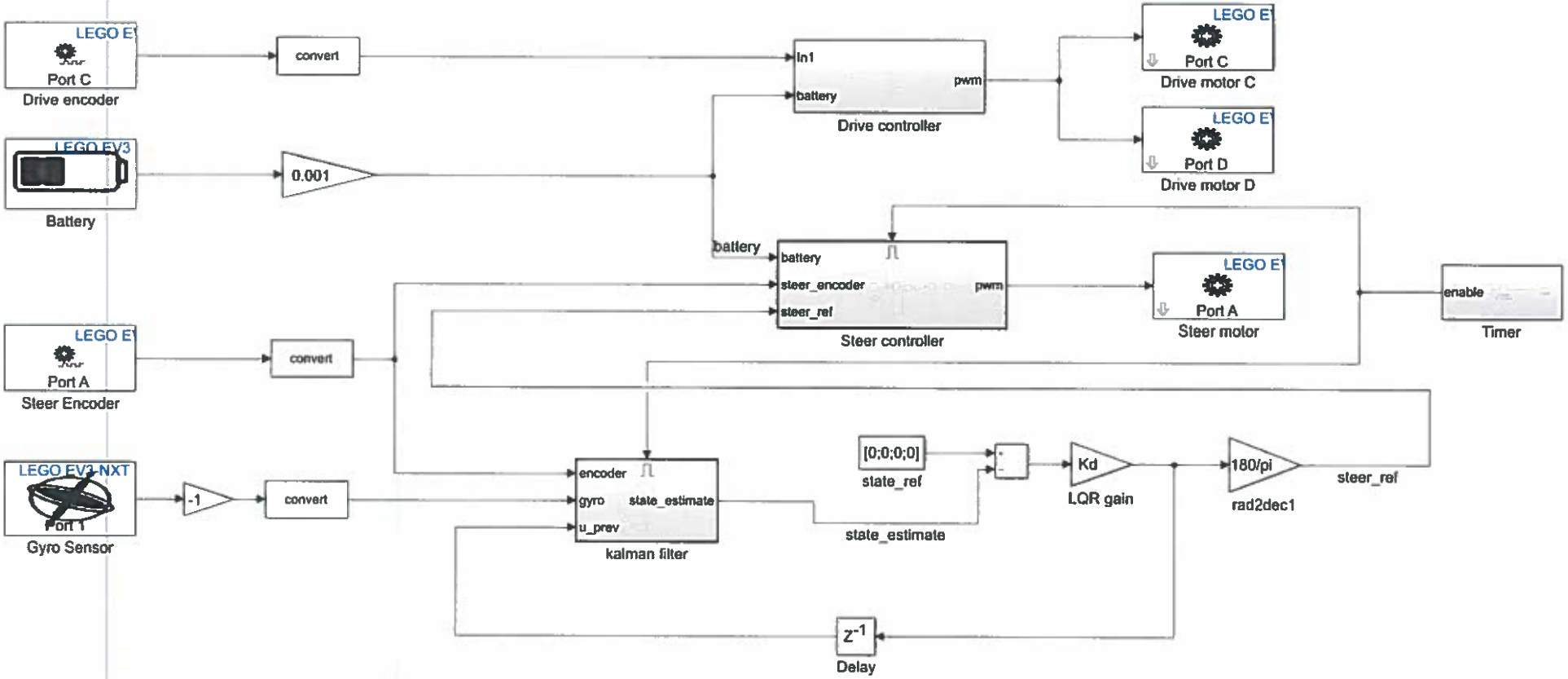


Figure 30: Full LQR controller implemented in Simulink

4.7 Discussion

The full LQR controller used (Figure 30) was not successful in practice. When implemented on the Lego bicycle, it often demanded very large steer angles that caused the bicycle to fall over. Changing the Q_{lqr} and R_{lqr} matrices did not improve performance.

Incorrect modelling could be a reason for the failure of this method. Ignoring the steer equation in Section 4.1 may be wrong because the DC motor actually exerts a torque on the steering axis that is related to the lean. Ignoring the steer equation and replacing it with Equation 4 only models the effect of the motor on steer angle. Instead, modelling the DC motor's torque output and taking T_δ as the control input will fit nicely with the Whipple model. Controlling steer torque rather than steer angle thus appears to be the better solution.

The model validation step in Section 4.4 was not sufficient. The small time window of around 0.5 s was too short to properly evaluate the correctness of the model. A possible solution would be to first design a stabilizing PD controller by experimental tuning as in Section 3. System identification can then be carried out on the measured data.

5 Conclusion

Two different methods of controlling a bicycle were considered in this project. A working Lego bicycle capable of self-stabilization and turning on demand was achieved using PD control (Figure 31). The LQR method worked in simulation but not on the physical bicycle. Nevertheless, it is a promising method and should be further pursued.

Figure 31: Lego Bicycle 1 stabilizing itself after leaving guide rails

5.1 Future Work

A more detailed model identification and validation step must be carried out if the control problem is to be properly solved. This was the most difficult part of the project because the system is unstable with dynamics dependent on forward speed. Multi-body simulation software can be used to achieve this. Alternatively, a human rider could first stabilize a full-scale bicycle without using body lean. The recorded system states could then be used to fit a model. Observing cyclists show that lean and steer angles are often very small even during turns, meaning that linearised models are likely to be sufficient.

Once a stabilizing controller capable of smooth turning at constant speed v is achieved, gain scheduling can be used to ensure that it remains stable across a range of speeds. An outer control loop for path-tracking can also be designed. As discussed in Section 3.1, there will be a threshold speed below which stabilization by steering is no longer possible. Track-standing, where the steer angle is fixed and the bicycle moves forward and backward, can be explored as a method of stabilizing a stationary bicycle.

Other methods of control could also be considered. One possible method is the Probabilistic Inference for Learning and Control (PILCO) framework. It first infers a probabilistic model of the system dynamics by applying inputs and observing the system behaviour. It then searches for a control policy that stabilizes the system. These steps of learning the dynamics, updating the controller parameters and testing it are repeated over a number of trials. It is data efficient, requiring only a few trials to learn a policy. It has been successfully used to swing up and balance an inverted pendulum, requiring less than 20 seconds of interaction with the system to learn the policy.

6 Risk assessment

No significant safety risks were encountered. In order to mitigate the risks of prolonged computer use, I took frequent breaks from work to rest my eyes. I also made sure to use cordoned off, secluded areas when testing my Lego bicycles to avoid any potential accidents.

7 Acknowledgements

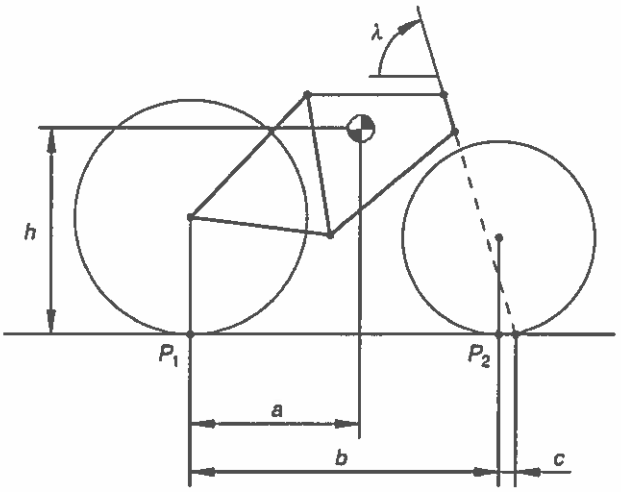
I would like to thank my supervisor, Dr. Richard Roebuck, for his invaluable support and guidance over the past year. I encountered many obstacles along the way and I would not have been able to complete the project without his help.

I would like to thank Dr. Fulvio Forni for giving me advice on control theory and lending me his Lego EV3 brick which made controller design and data acquisition much easier.

I would like to thank my friends (in particular Andrea, Joshua, Joel, Ruthanne, Cheng wen and William) for their encouragement and for making my final year a fun-filled one.

Appendix A Inverted pendulum model

These equations are based on a paper by Astrom [4]. An "inverted pendulum" model is used, where the bicycle is modelled as a point mass located at its centre of gravity. It is also assumed to be symmetrical and fully described by 6 constants as shown in Figure 32. P1 and P2 are assumed to be point contacts.



	Symbol
Distance to CG	a
Wheel Base	b
Trail	c
Steering axis angle	λ
Height of CG	h
Mass of bicycle	m

Figure 32: Bicycle parameters

Frame Lean equation

The bicycle is assumed to be travelling at a constant speed v, with a rightward lean φ and leftward steer δ.

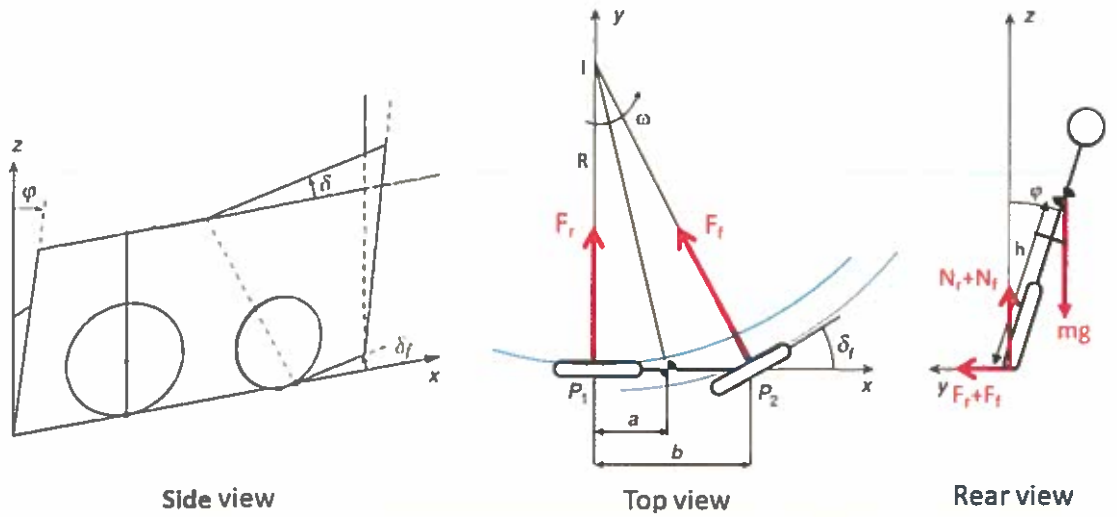


Figure 33: Different views of a moving bicycle

Because of the non vertical steering axis, the steering angle δ is related to the projected steering angle δ_f (derived in Cossalter, 2006, p.39).

$$\tan \delta_f = \tan \delta \sin \lambda \tag{6}$$

The bicycle enters into a left turn with radius of curvature R. The front and back wheel map out concentric circles around the instantaneous centre of curvature I.

$$R = \frac{b}{\tan \delta_f}$$
$$\omega = \frac{v}{R} = \frac{v \tan \delta_f}{b}$$

Angular momentum about the X axis can be expressed as

$$\begin{aligned} L_x &= I_{xx}\dot{\varphi} - I_{xz}\omega \\ &\approx mh^2\dot{\varphi} - \frac{mahv \tan \delta_f}{b} \end{aligned}$$

The normal reaction force from the wheel contacts N_r, N_f must balance the weight of the bicycle, while the lateral reaction F_r, F_f provides the net perpendicular force for centripetal acceleration. Taking moments about the X axis,

$$\begin{aligned} T_x &= mgh \sin \varphi + \frac{mv^2}{R} h \cos \varphi \\ &= mgh \sin \varphi + \frac{mv^2 h \tan \delta_f \cos \varphi}{b} \end{aligned}$$

Because of the geometry of the steering axis, a leftward steer causes the front ground contact to shift from P_2 to P_2' by perpendicular distance Δy . The X axis is consequently rotated slightly to X'.

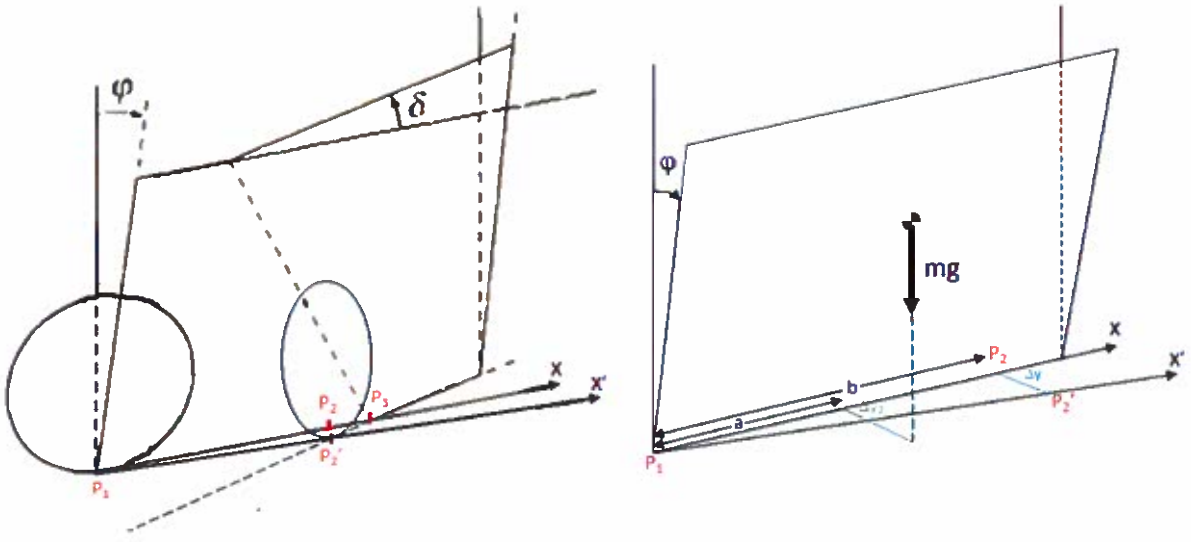


Figure 34: Left steer shifts front contact slightly to the right from P2 to P2'

The distance Δy is derived in [5].

$$\Delta y \approx \frac{c \sin \lambda}{\cos \varphi} \delta$$

For the same lean angle φ, the moment arm of the mass is now reduced by Δy_2

$$\Delta y_2 = \frac{ac \sin \lambda}{b \cos \varphi} \delta$$

The net torque about the wheel base X' now becomes

$$T_x = mg(h \sin \varphi - \frac{ac \sin \lambda}{b \cos \varphi} \delta) + \frac{mv^2 h \tan \delta_f \cos \varphi}{b}$$

Applying conservation of angular momentum,

$$\frac{dL_x}{dt} = T_x$$

$$mh^2 \ddot{\varphi} - \frac{mahv \sec^2 \delta_f}{b} \dot{\delta}_f = mg(h \sin \varphi - \frac{ac \sin \lambda}{b \cos \varphi} \delta) + \frac{mv^2 h \tan \delta_f \cos \varphi}{b} \tag{7}$$

Combining equations 6 and 7, and assuming small angles gives the linear frame lean equation

$$\boxed{mh^2 \ddot{\varphi} - \frac{mahv \sin \lambda}{b} \dot{\delta} = mgh\varphi + \frac{m(v^2 h - acg) \sin \lambda}{b} \delta} \tag{8}$$

Path Equation

Initially, the bicycle's coordinate axes $[X, Y, Z]$ are aligned with the fixed global axes $[\epsilon, \eta, \zeta]$ at the origin O. As the bicycle travels at constant speed v with some changing steer angle δ, its rear wheel contact P_1 maps out a path in the $\epsilon\eta$ plane.

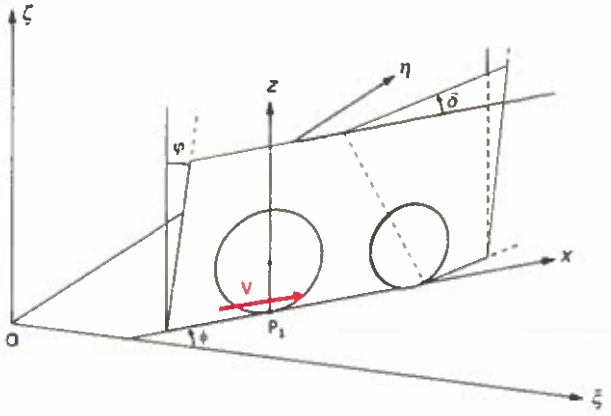


Figure 35: Location of bicycle relative to the global origin O

Equations for the position of the bicycle can be easily worked out

$$\boxed{\begin{aligned} \dot{\eta} &= v \sin \phi \approx v\phi \\ \dot{\epsilon} &= v \cos \phi \approx v \\ \dot{\phi} = \omega &= \frac{v \tan \delta_f}{b} \approx \frac{v\delta \sin \lambda}{b} \end{aligned}} \tag{9}$$

Appendix B Whipple model

The Whipple model is used, where the bicycle is modelled as 4 connected rigid bodies (Handlebar assembly H, main body frame B, front wheel F and rear wheel R). P1 and P2 are assumed to be point contacts.
For each of these bodies, the mass m, CG location (x, z) and inertia matrix (specifically I_{xx}, I_{xz}, I_{zz}) have to be specified. Since the bicycle is assumed to be symmetric about the XZ plane, all products of inertial terms involving Y will be 0. Wheels are also symmetric about the XY plane, hence cross products with z will be 0 and $I_{xx} = I_{zz}$. The origin is defined to be at the rear wheel contact P1, and the axes are oriented as shown in Figure 36.

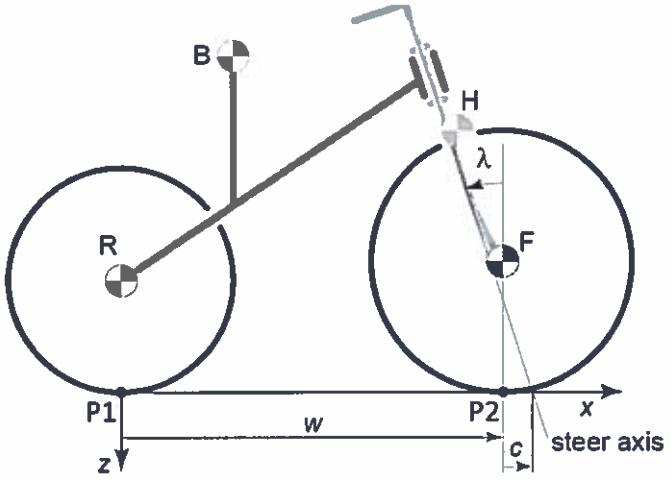


Figure 36: Whipple bicycle model

Treating the bike as a combined rigid body (T=R+B+H+F), properties of the total bike T can be defined as follows.

$$
\begin{aligned}
m_T &= m_R + m_F + m_B + m_H \\
x_T &= (x_B m_B + x_F m_F + x_H m_H)/m_T \\
z_T &= (z_B m_B + z_F m_F + z_H m_H + z_R m_R)/m_T
\end{aligned}
$$

The inertia terms for T defined at P1 and along the global axes are shown below. The parallel axis theorem is used.

$$
\begin{aligned}
I_{Txx} &= I_{Rxx} + m_R z_R^2 + I_{Hxx} + m_H z_H^2 + I_{Fxx} + m_F z_F^2 + I_{Bxx} + m_B z_B^2 \\
I_{Tzz} &= I_{Rzz} + I_{Hzz} + m_H x_H^2 + I_{Fzz} + m_F x_F^2 + I_{Bzz} + m_B x_B^2 \\
I_{Txz} &= -m_F x_F z_F + I_{Hxz} - m_H x_H z_H + I_{Bxz} - m_B x_B z_B
\end{aligned}
$$

The steering column can also be treated as one body (A=H+F). Properties can similarly be defined. The inertia values are defined at the location of the combined CG (x_A,z_A)

$$
\begin{aligned}
m_A &= m_F + m_H \\
x_A &= (x_F m_F + x_H m_H)/m_A \\
z_A &= (z_F m_F + z_H m_H)/m_A \\
I_{Axx} &= I_{Hxx} + m_H(z_H - z_A)^2 + I_{Fxx} + m_F(z_F - z_A)^2 \\
I_{Azz} &= I_{Hzz} + m_H(x_H - x_A)^2 + I_{Fzz} + m_F(x_F - x_A)^2 \\
I_{Axz} &= I_{Hxz} - m_H(x_H - x_A)(z_H - z_A) - m_F(x_F - x_A)(z_F - z_A)
\end{aligned}
$$

Some geometrical parameters of the front steering column can be worked out. μ_C (perpendicular distance of the front wheel contact behind the steering axis) and μ_A (perpendicular distance of front column CG from steering axis) are derived referring to Figure 37.

$$
\begin{aligned}
\mu_C &= c\cos\lambda \\
\mu_A &= -z_A \sin\lambda - (w + c - x_A)\cos\lambda
\end{aligned}
$$

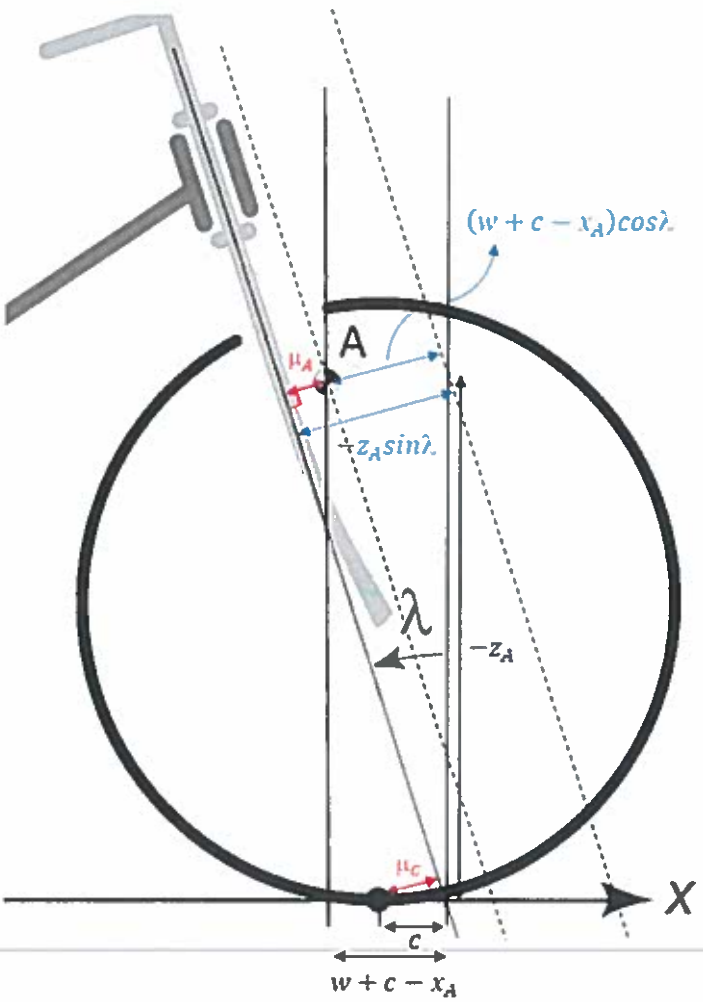


Figure 37: Geometry of steering column A

The inertia matrix for the steering column A can be defined at several locations for ease of use with the equations later. Referring to Figure 38, the inertia matrix for the steering column A can be defined at the point A2 (where the steering axis intersects the ground plane)

$$I' = \begin{bmatrix} I_{Axx} + m_A z_0^2 & 0 & I_{Axz} - m_A x_0 z_0 \\ 0 & I_{Ayy} + m_A(z_0^2 + x_0^2) & 0 \\ I_{Axz} - m_A z_0 x_0 & 0 & I_{Azz} + m_A x_0^2 \end{bmatrix}$$

$$z_0 = z_A$$
$$x_0 = -(w + c - x_A)$$

It can also be defined at the point A1 (where the steering axis intersects the z axis)

$$I'' = \begin{bmatrix} I_{Axx} + m_A z_1^2 & 0 & I_{Axz} - m_A x_1 z_1 \\ 0 & I_{Ayy} + m_A(z_1^2 + x_1^2) & 0 \\ I_{Axz} - m_A z_1 x_1 & 0 & I_{Azz} + m_A x_1^2 \end{bmatrix}$$

$$z_1 = \frac{(w + c)}{\tan \lambda} + z_A$$
$$x_1 = x_A$$

The inertia matrices I', I'' can also be rotated anticlockwise by λ such that the z-axis aligns with the steer axis. The superscript R denotes a rotated inertia matrix

$$I^R = RIR^T$$

$$R = \begin{bmatrix} \cos\lambda & 0 & -\sin\lambda \\ 0 & 1 & 0 \\ \sin\lambda & 0 & \cos\lambda \end{bmatrix}$$

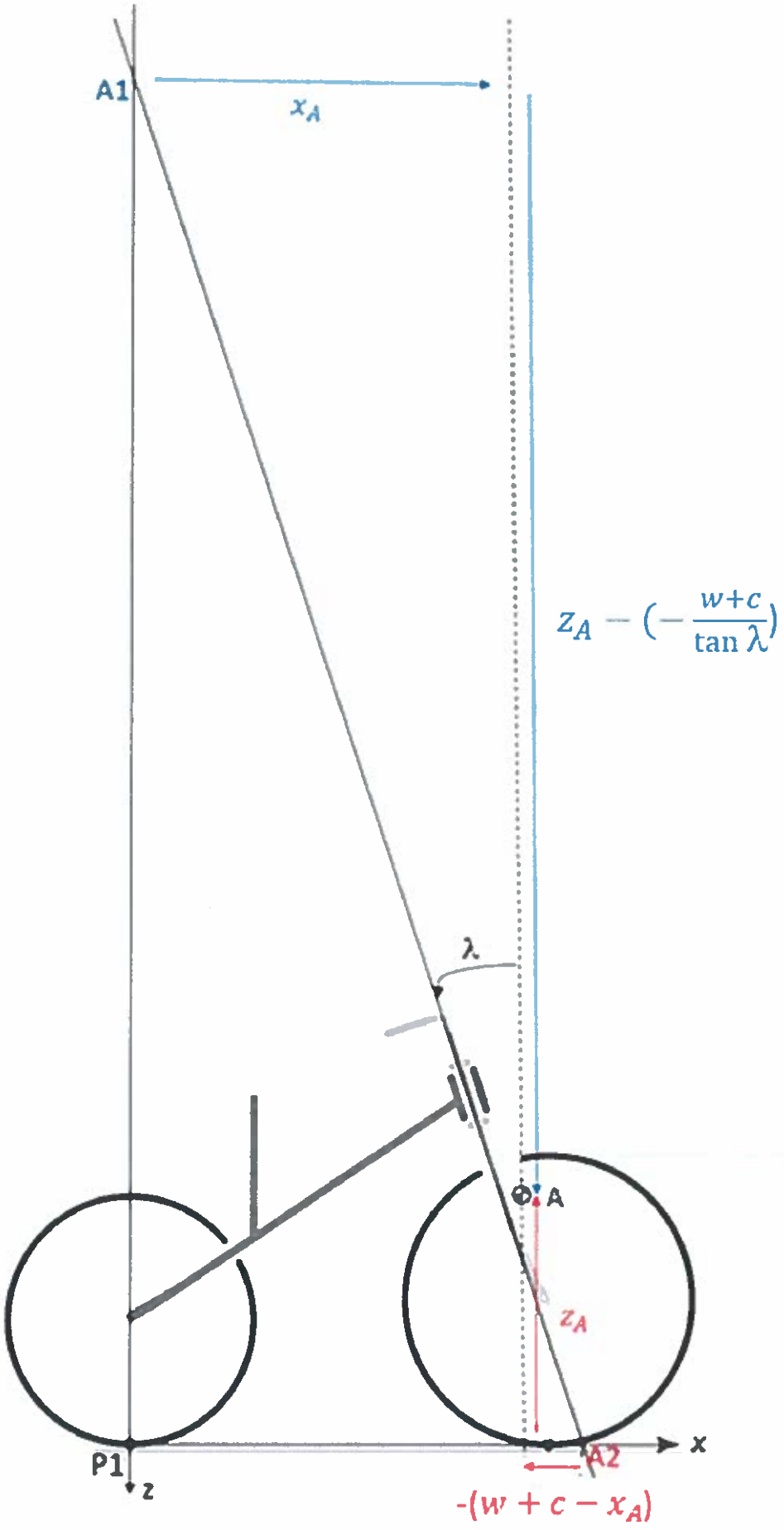


Figure 38: Inertia defined at different points A1,A2

Conservation of angular momentum

The bicycle is assumed to be travelling at a constant speed v, with wheel rotations θ_F and θ_R, lean φ, steer δ and yaw ϕ. The rear contact point has lateral acceleration $\ddot{y}_R$. An external steer torque T_δ and lean torque T_φ is applied to the bicycle, as well as ground contact forces F_R, F_F. Equations of motion can be derived by applying conservation of angular momentum about the different axes of rotation. All angles are assumed small.

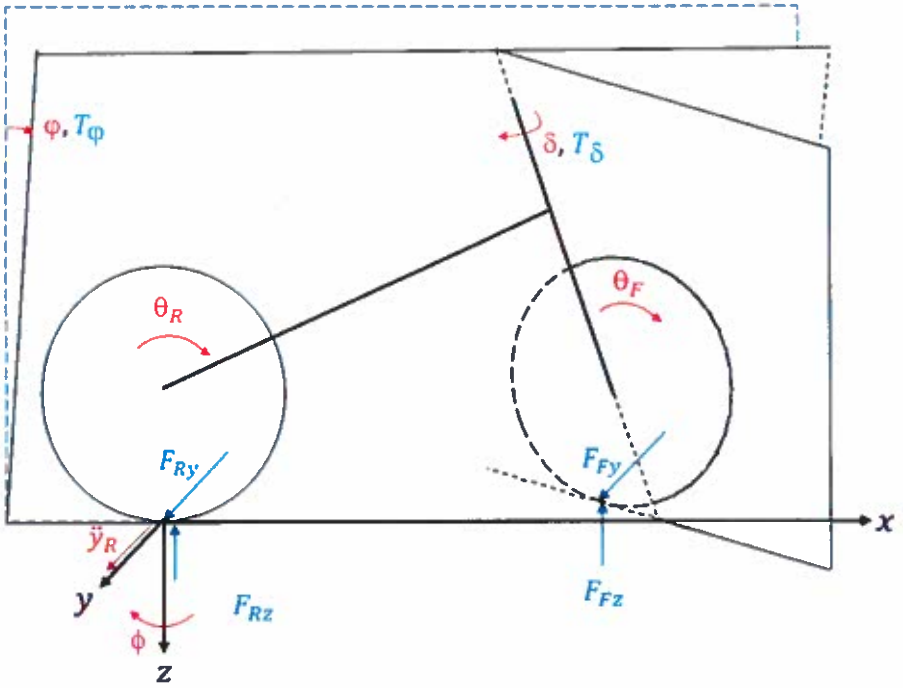


Figure 39: External forces acting on bicycle (blue) and direction of rotation and acceleration of bicycle (red)

About tilt axis

Applying conservation of angular momentum about the X axis.

$$\begin{aligned} T_\varphi - m_T g z_T \varphi + F_{Fz}\delta\mu_C + m_A g \delta \mu_A = &- m_T \ddot{y}_R z_T \\ &+ I_{Txx}\ddot{\varphi} + I_{Txz}\ddot{\phi} \\ &+ I'_{11}(\ddot{\delta}\sin\lambda) + I'_{13}\ddot{\delta}\cos\lambda \\ &+ (I_{Fyy}\dot{\theta}_F + I_{Ryy}\dot{\theta}_R)\dot{\phi} + I_{Fyy}\dot{\theta}_F\dot{\delta}\cos\lambda \end{aligned} \tag{10}$$

LHS:

- Externally applied lean torque
- Torque from gravity due to lean of bicycle
- Torque from lateral displacement of the front vertical reaction force (Figure 40)
- Torque from shifted CG of steering column A due to steer (Figure 40)

RHS:

- x moment required for y acceleration of whole bicycle
- Rate of change of angular momentum of whole bicycle in X direction due to lean and yaw
- Rate of change of angular momentum of steering column A in X direction due to steer
- Gyroscopic torques required for precession

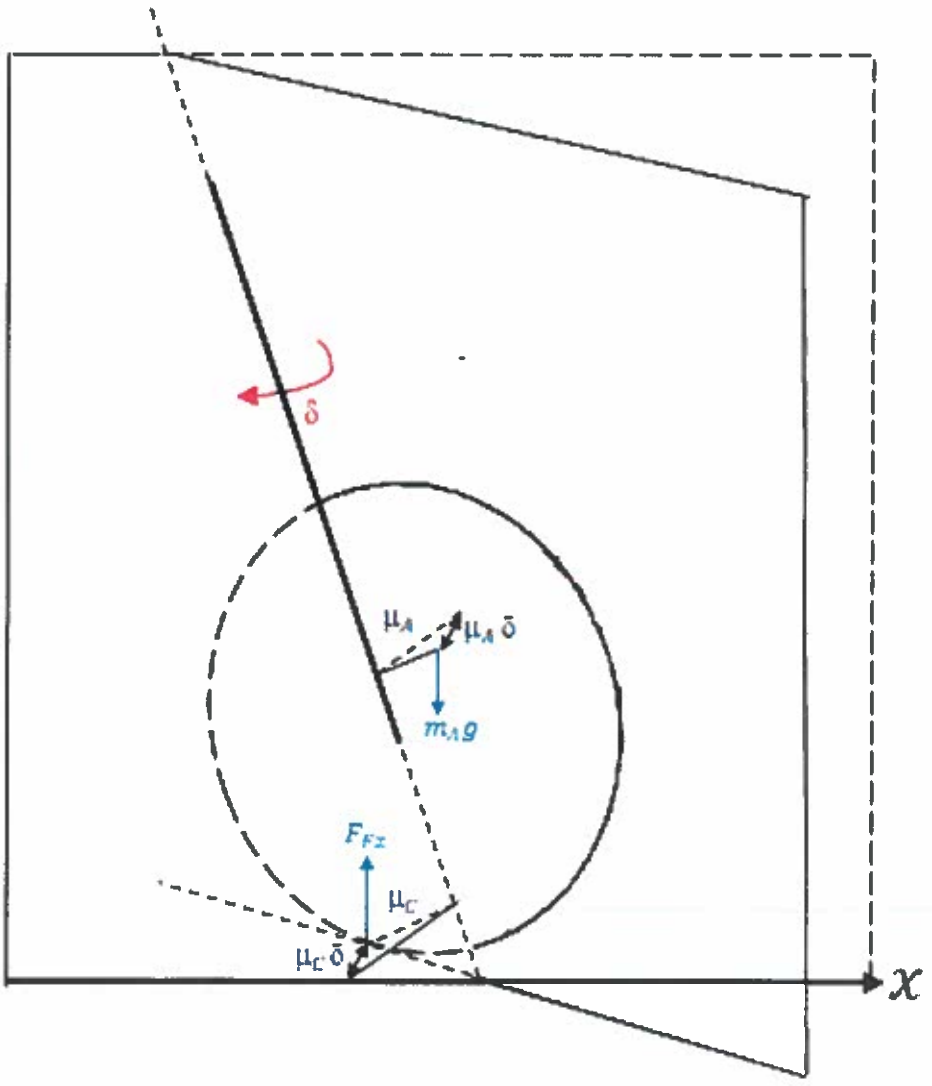


Figure 40: When the bicycle is only steered but not leaned, the vertical front contact force F_{Fz} and CG of steering column A are displaced from the body plane, and produces a torque about the x axis.

About yaw axis

Applying conservation of angular momentum about the z axis,

$$\begin{aligned} F_{Fy}w =& m_T\ddot{y}_R x_T \\ &+ I_{Txz}\ddot{\varphi} + I_{Tzz}\ddot{\phi} \\ &+ I''_{31}(\ddot{\delta}\sin\lambda) + I''_{33}\ddot{\delta}\cos\lambda \\ &- (I_{Fyy}\dot{\theta}_F + I_{Ryy}\dot{\theta}_R)\dot{\varphi} - I_{Fyy}\dot{\theta}_F(\dot{\delta}\sin\lambda) \end{aligned} \tag{11}$$

LHS:

- Torque about z axis from front lateral reaction force

RHS:

- z moment required for y acceleration of whole bicycle
- Rate of change of angular momentum of whole bicycle in z direction due to lean and yaw
- Rate of change of angular momentum of steering column A in z direction due to steer
- Gyroscopic torques required for precession

About steer axis

Applying conservation of angular momentum for A about the steer axis,

$$\begin{aligned} &T_\delta + \left[m_A g\varphi\mu_A + F_{Fz}\varphi\mu_C\right] + \left[m_A g(\sin\lambda)\mu_A\delta + F_{Fz}(\sin\lambda)\mu_C\delta\right] - F_{Fy}\mu_C = \\ &m_A\ddot{y}_R\mu_A + (-I''^R_{31}\ddot{\phi}\sin\lambda + I''^R_{33}\ddot{\phi}\cos\lambda) + (I''^R_{31}\ddot{\varphi}\cos\lambda + I'^R_{33}\ddot{\varphi}\sin\lambda) + I'^R_{33}\ddot{\delta} + I_{Fyy}\dot{\theta}_F(-\dot{\varphi}\cos\lambda + \dot{\phi}\sin\lambda) \end{aligned} \tag{12}$$

LHS:

- Externally applied steer torque
- When bicycle leaned only, CG of front column and vertical ground reaction force exert torque about steering axis (Figure 41)
- When bicycle steered only, CG of front column and vertical ground reaction force exert torque about steering axis (Figure 42)
- Torque about steering axis from front lateral reaction force

RHS:

- Steering moment required for y acceleration of front column
- Rate of change of angular momentum of steering column due to yaw
- Rate of change of angular momentum of steering column due to lean
- Rate of change of angular momentum of steering column due to steer
- Gyroscopic torques required for precession of front wheel

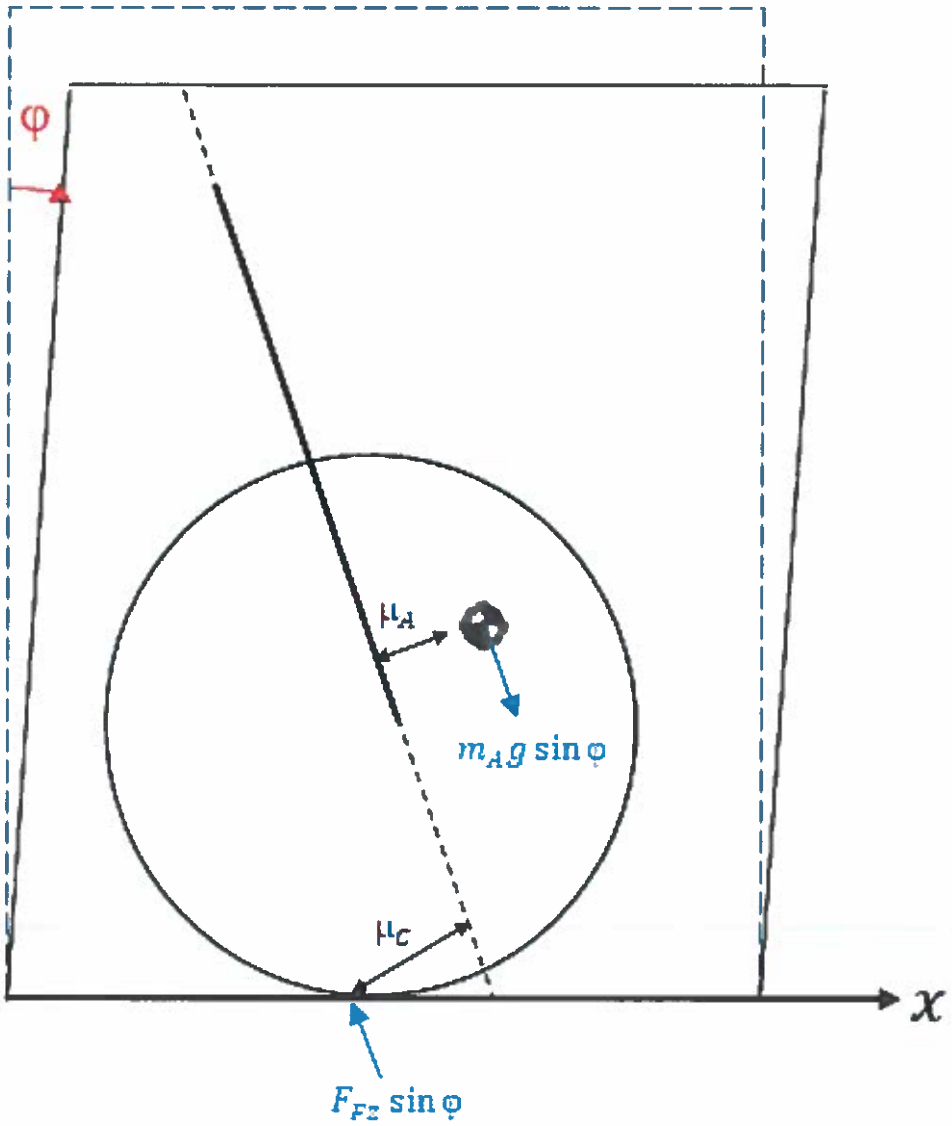


Figure 41: When the bicycle is only leaned but not steered, vertical contact force and mass A have components perpendicular to plane of steering wheel that gives torque.

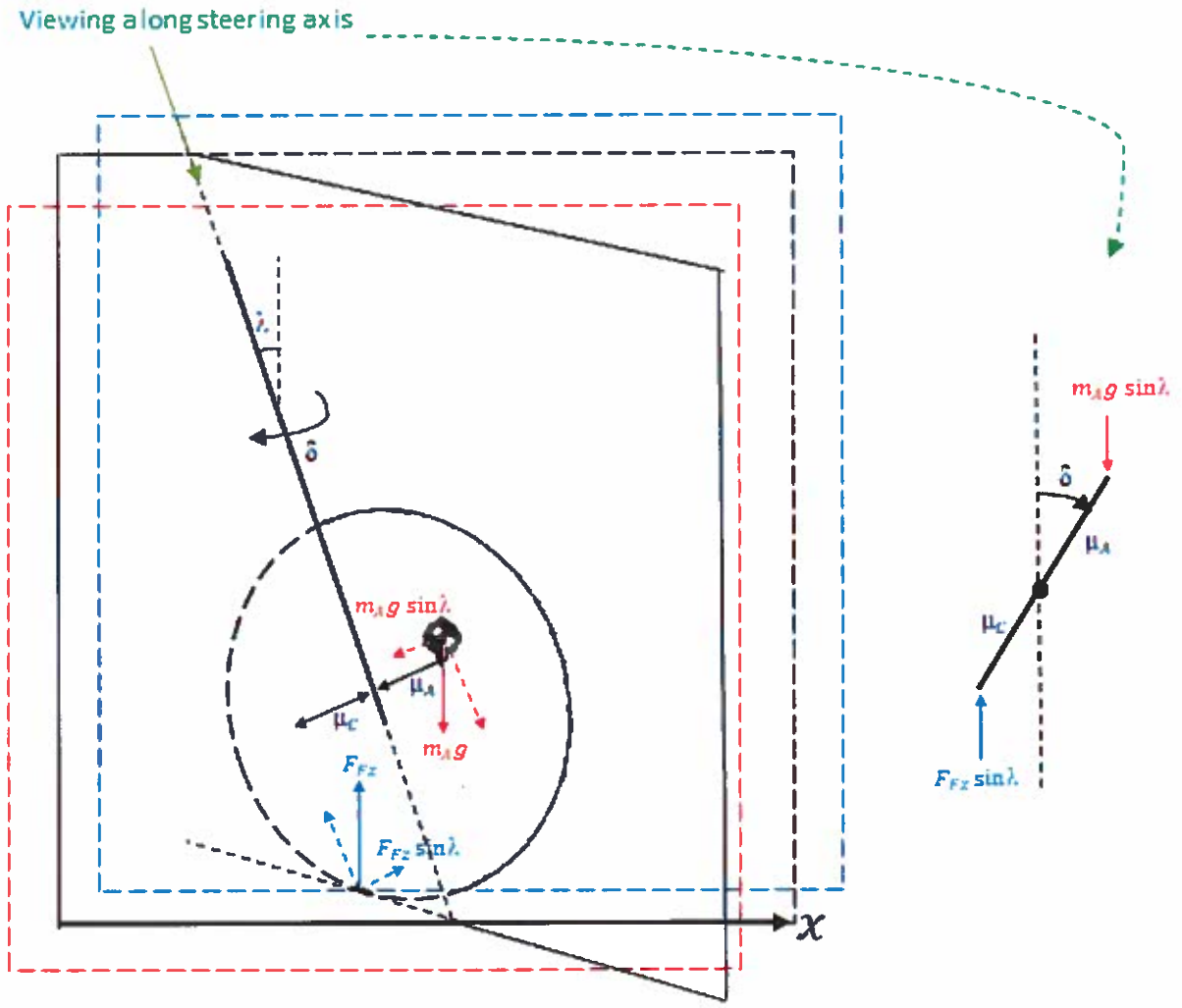


Figure 42: When the bicycle is only steered but not leaned, the vertical front contact force F_{Fx} and mass A have components perpendicular to the steering axis that gives torque. Similarly coloured lines are in the same vertical plane.

Constraints and geometrical relationships

Only equations that express the lateral dynamics of the bicycle are of interest. Several constraints and relationships can be included to reduce the number of variables.
The angular velocity of the wheels are dependent on the velocity v

$$\begin{aligned} \dot{\theta}_F &= \frac{v}{r_F} \\ \dot{\theta}_R &= \frac{v}{r_R} \end{aligned} \tag{13}$$

The vertical ground reaction forces must balance the weight of the bicycle

$$\begin{aligned} F_{Fz} &= m_T g \frac{x_T}{w} \\ F_{Rz} &= m_T g \frac{w - x_T}{w} \end{aligned} \tag{14}$$

Referring to Figure 43, relationships between the variables can be derived. Assuming no wheel slip,

$$\dot{y}_R = v\phi \tag{15}$$

$$y_F \approx y_R + w\phi - c(\delta \cos\lambda)$$

$$\dot{y}_F = \frac{d(y_R + w\phi - c(\delta \cos\lambda)}{dt} = v(\phi + \delta \cos\lambda) \tag{16}$$

Subtracting equation 5 from 6 and taking the derivative,

$$\dot{\phi} = (\frac{c\dot{\delta} + v\delta}{w}) \cos\lambda \tag{17}$$

$$\ddot{\phi} = (\frac{v\dot{\delta} + c\ddot{\delta}}{w}) \cos\lambda \tag{18}$$

Finally, equation 5 can be differentiated to give

$$\ddot{y}_R = v\ddot{\phi} = (\frac{v^2\delta + vc\dot{\delta}}{w}) \cos\lambda \tag{19}$$

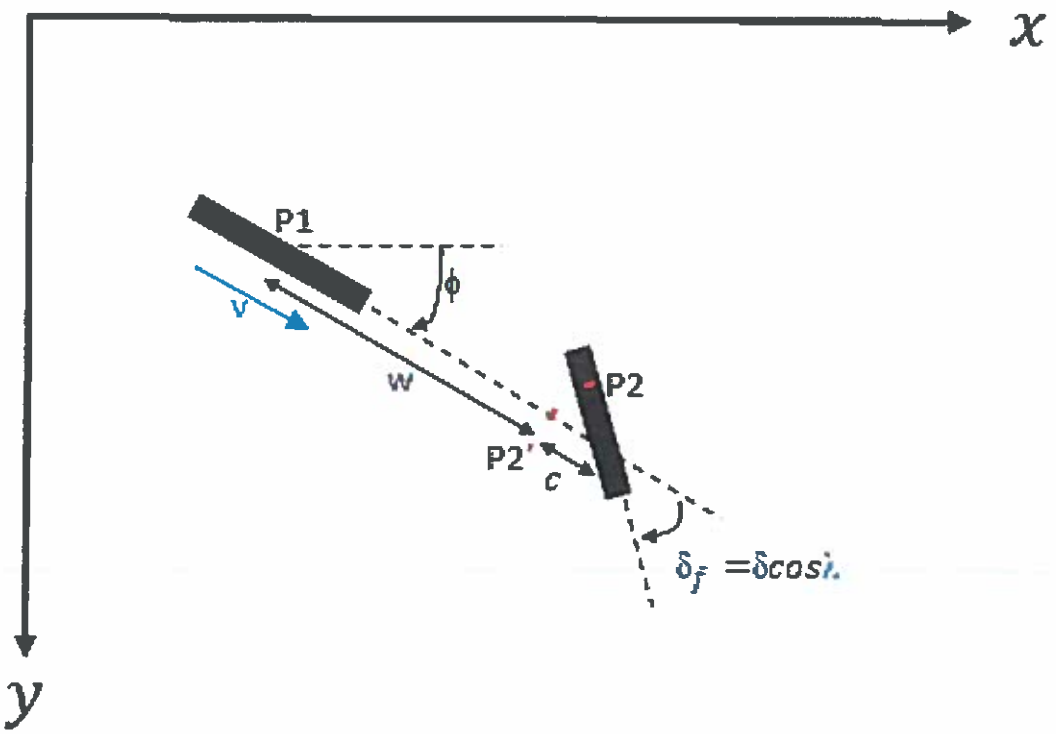


Figure 43: Motion of bicycle relative to global origin. Front ground contact is shifted from P2' to P2 when steering.

Substituting these relationships into equations 10, 11 and 12 will give the equations of motion. Doing this will yield the exact same equations as the form used below (checked using Matlab symbolic math toolbox).

Equations of motion of bicycle

Some quantities can be defined to be consistent with the paper and to shorten notation.

$$
\begin{aligned}
I_{A\lambda\lambda} &= m_A\mu_A^2 + I_{Axx}\sin^2\lambda + 2I_{Axz}\sin\lambda\cos\lambda + I_{Azz}\cos^2\lambda \\
I_{A\lambda x} &= -m_A\mu_A z_A + I_{Axx}\sin\lambda + I_{Axz}\cos\lambda \\
I_{A\lambda z} &= -m_A\mu_A x_A + I_{Axz}\sin\lambda + I_{Azz}\cos\lambda \\
S_R &= \frac{I_{Ryy}}{r_R} \\
S_F &= \frac{I_{Fyy}}{r_F} \\
S_T &= S_R + S_F \\
\mu &= \frac{\mu_C}{w} \\
S_A &= m_A\mu_A + \mu m_T x_T
\end{aligned}
$$

Lean equation

Substituting the above constraints and the relevant inertia matrix elements into equation 10 gives the lean equation. In matrix form,

$$
\begin{bmatrix} M_{11} & M_{12} \end{bmatrix} \begin{bmatrix} \ddot{\varphi} \\ \ddot{\delta} \end{bmatrix} + v \begin{bmatrix} C1_{11} & C1_{12} \end{bmatrix} \begin{bmatrix} \dot{\varphi} \\ \dot{\delta} \end{bmatrix} + \left(g \begin{bmatrix} k0_{11} & k0_{12} \end{bmatrix} + v^2 \begin{bmatrix} k2_{11} & k2_{12} \end{bmatrix} \right) \begin{bmatrix} \varphi \\ \delta \end{bmatrix} = T_\varphi
$$

$$
\begin{aligned}
M_{11} &= I_{Txx} \\
M_{12} &= I_{A\lambda x} + I_{Txz}\mu \\
C1_{11} &= 0 \\
C1_{12} &= S_T\mu + S_F\cos\lambda + \frac{I_{Txz}\cos\lambda}{w} - m_T z_T \mu \\
k0_{11} &= m_T z_T \\
k0_{12} &= -s_A \\
k2_{11} &= 0 \\
k2_{12} &= (-m_T z_T + S_T)\frac{\cos\lambda}{w}
\end{aligned}
$$

Steer equation

First, equations 11 and 12 can be combined to eliminate F_{Fy} the front lateral contact force. Next, similarly substituting in the constraints and the relevant inertia matrix elements gives the steer equation. In matrix form,

$$\begin{bmatrix} M_{21} & M_{22} \end{bmatrix} \begin{bmatrix} \ddot{\varphi} \\ \ddot{\delta} \end{bmatrix} + v \begin{bmatrix} C1_{21} & C1_{22} \end{bmatrix} \begin{bmatrix} \dot{\varphi} \\ \dot{\delta} \end{bmatrix} + \left(g \begin{bmatrix} k0_{21} & k0_{22} \end{bmatrix} + v^2 \begin{bmatrix} k2_{21} & k2_{22} \end{bmatrix} \right) \begin{bmatrix} \varphi \\ \delta \end{bmatrix} = T_\delta$$

$$\begin{aligned}
M_{21} &= I_{A\lambda x} + I_{Txz}\mu \\
M_{22} &= I_{A\lambda\lambda} + 2I_{A\lambda z}\mu + I_{TZZ}\mu^2 \\
C1_{21} &= -S_T\mu - S_F \cos\lambda \\
C1_{22} &= S_A\mu + (I_{Tzz}\mu + I_{A\lambda z})\frac{\cos\lambda}{w} \\
k0_{21} &= -S_A \\
k0_{22} &= -S_A \sin\lambda \\
k2_{21} &= 0 \\
k2_{22} &= \left[S_A + S_F \sin\lambda\right]\frac{\cos\lambda}{w}
\end{aligned}$$

Overall equation

The lean and steer equation in the previous section can be combined to give the overall linearised equations for a bicycle. This is the form of the equation presented in the paper.

$$\begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \begin{bmatrix} \ddot{\varphi} \\ \ddot{\delta} \end{bmatrix} + v \begin{bmatrix} C1_{11} & C1_{12} \\ C1_{21} & C1_{22} \end{bmatrix} \begin{bmatrix} \dot{\varphi} \\ \dot{\delta} \end{bmatrix} + \left(g \begin{bmatrix} k0_{11} & k0_{12} \\ k0_{21} & k0_{22} \end{bmatrix} + v^2 \begin{bmatrix} k2_{11} & k2_{12} \\ k2_{21} & k2_{22} \end{bmatrix} \right) \begin{bmatrix} \varphi \\ \delta \end{bmatrix} = \begin{bmatrix} T_\varphi \\ T_\delta \end{bmatrix}$$

$$\boldsymbol{M\ddot{q} + C\dot{q} + Kq = f}$$

References

[1] Jones, D. E. (1970). The stability of the bicycle. Physics today, 23(4), 34-40.

[2] Meijaard, J. P., Papadopoulos, J. M., Ruina, A., & Schwab, A. L. (2011). History of thoughts about bicycle self-stability.

[3] Kooijman, J. D. G., Meijaard, J. P., Papadopoulos, J. M., Ruina, A., & Schwab, A. L. (2011). A bicycle can be self-stable without gyroscopic or caster effects. Science, 332(6027), 339-342.

[4] Astrom, K. J., Klein, R. E., & Lennartsson, A. (2005). Bicycle dynamics and control: adapted bicycles for education and research. IEEE Control Systems, 25(4), 26-47.

[5] Cossalter, V. (2010). Motorcycle dynamics (2nd ed.). Lexington, KY: Lulu.

[6] Meijaard, J. P., Papadopoulos, J. M., Ruina, A., & Schwab, A. L. (2007, August). Linearized dynamics equations for the balance and steer of a bicycle: a benchmark and review. In Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences (Vol. 463, No. 2084, pp. 1955-1982). The Royal Society.

[7] Papadopoulos, J. M. (1987). Bicycle steering dynamics and self-stability: a summary report on work in progress. Cornell Bicycle Research Project, Cornell University, Ithaca, NY.

[8] He, J., Zhao, M., & Stasinopoulos, S. (2015, December). Constant-velocity steering control design for unmanned bicycles. In Robotics and Biomimetics (ROBIO), 2015 IEEE International Conference on (pp. 428-433). IEEE.

[9] Michini, B., & Torrez, S. (2007). Autonomous stability control of a moving bicycle. In AIAA NE Regional Student Conference.

[10] Mutsaerts, J. (2010). Nxtbike-gs (Doctoral dissertation, Master's thesis, Delft University of Technology).

[11] Budaciu, C., & Apostol, L. D. (2016, October). Dynamic analysis and control of lego mindstorms NXT bicycle. In System Theory, Control and Computing (ICSTCC), 2016 20th International Conference on (pp. 145-149). IEEE.

[12] Basso, M., Innocenti, G., & Rosa, A. (2013, December). Simulink meets lego: Rapid controller prototyping of a stabilized bicycle model. In Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on (pp. 330-335). IEEE.

[13] Sasaki, M., Tanaka, H., & Ito, S. (2012). Development of an Autonomous Two-Wheeled Vehicle Robot. In Advanced Engineering Forum (Vol. 2, pp. 390-395). Trans Tech Publications.

[14] Randløv, J., & Alstrøm, P. (1998, July). Learning to Drive a Bicycle Using Reinforcement Learning and Shaping. In ICML (Vol. 98, pp. 463-471).

[15] Cook, M. (2004). It takes two neurons to ride a bicycle. Demonstration at NIPS, 4.