Self-Balancing Bicycle

Philip M. Salmony

Department of Engineering

University of Cambridge

Supervisor: Dr. Fulvio Forni

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

Signed **Date**

Self-Balancing Bicycle

Philip Salmony, Wolfson College

Abstract

Contents

1 Introduction **1**

2 Modelling **1**

2.1 Geometry and Definitions . . . 2

2.2 Second Order Model . . . 3

2.2.1 Derivation . . . 3

2.2.2 Stability . . . 5

2.3 Fourth Order Model . . . 6

2.3.1 Stability . . . 7

2.4 Parameter Estimation . . . 9

2.4.1 Transfer Functions . . . 10

2.5 Actuators . . . 10

2.5.1 Lego Prototype . . . 11

2.5.2 Full-Scale Bicycle . . . 12

2.6 Model Verification . . . 13

3 Controller Design and Simulation **13**

3.1 Performance Requirements . . . 14

3.2 Operating Point . . . 14

3.3 PID . . . 15

3.3.1 Lego Prototype . . . 16

3.4 Linear Quadratic Regulator . . . 17

3.4.1 Lego Prototype . . . 18

3.4.2 Full-Scale Bicycle . . . 19

3.5 H_∞ Loop-Shaping . . . 19

3.5.1 Lego Prototype . . . 21

3.6 Non-Linear Methods . . . 23

3.7 Comparison . . . 24

4 Implementation: Lego Prototype **24**

4.1 Hardware 24

4.2 Software 24

4.2.1 Lean Angle Estimation 24

4.2.2 Discrete Form of PID 26

4.3 Results 26

4.3.1 Proportional Controller 26

4.3.2 Proportional-Derivative Controller 28

4.3.3 Final Observations 29

5 Implementation: Full-Scale Bicycle **29**

5.1 Hardware 30

5.1.1 Bicycle Frame 30

5.1.2 Drive Motor 30

5.1.3 Handlebar Actuator 30

5.1.4 Microprocessor and Sensors 30

5.1.5 Miscellaneous 30

5.2 Software 30

5.2.1 Code Layout 30

5.2.2 Lean Angle Estimation 30

5.2.3 Basestation and Telemetry 31

5.3 Results 31

6 Conclusions and Future Work **31**

References **32**

Appendix A Risk Assessment **32**

1 Introduction

Bicycles and their stability properties have been studied ever since their inception in the late 19th century, with the topic continuing to attract interest up to this very day. A vast amount of research on bicycles has been undertaken, most notably by academics such as Whipple (1899, [1]), Papadopoulos (1980s [2]), and Astrom (2005, [3]).

For the rider-less case, the bicycle can be classed as an *underactuated* system, since for this project, the bicycle is only equipped with a drive actuator to provide forward speed and an actuator placed at the handlebars to provide a *steer* torque. and Underactuated, actuator at handlebar commanding steer angle...
This project aims to investigate what factors influence the stability of a bicycle, and additionally explores the modelling of the dynamics, controller design, and finally the real-world implementation. This report details the work completed and the insights gained on bicycle stability and control.

2 Modelling

To be able to design suitable control laws to stabilise the bicycle, it is necessary to provide a mathematical description of the system. Additionally, developing a dynamical model of the system will give an insight into what factors influence the bicycle behaviour in terms of its stability. As previously mentioned, the equations of motion for a bicycle are a set of many, highly non-linear, and coupled differential equations. In this form, they do not aid an intuitive understanding of what makes a bicycle more or less stable and are inconvenient to use for control law design, which is based mainly on linear methods. For simulation however, these non-linear equations of motion are important, as they capture the dynamics over the entire range of operating regimes.
This section aims to explore two commonly used linearised bicycle models of second and fourth order, which will be the basis for controller design in later sections.

2.1 Geometry and Definitions

The definitions of bicycle parameters, geometry, and coordinate system in this report follows the system given by Åström in [3], which is based on the *ISO 8855* standard [4]. A brief explanation of the bicycle parameters, as well as the top, rear, and side views of the bicycle are detailed in Table 1 and Figures 1 and 2.

Parameter	Symbol	Description
Center of Mass (z)	h	Distance of center of mass from ground.
Center of Mass (x)	a	Distance of center of mass from center of rear wheel.
Wheel Base	b	Distance between centers of both wheels.
Trail	c	Distance between intersection of steering axis with ground and point of contact with ground of front wheel.
Head Angle	λ	Angle between steering axis and ground.

Table 1: Bicycle Parameter Definitions

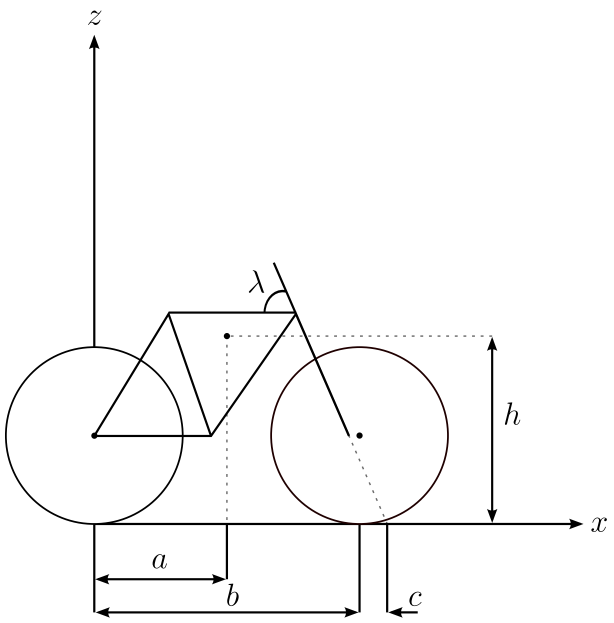
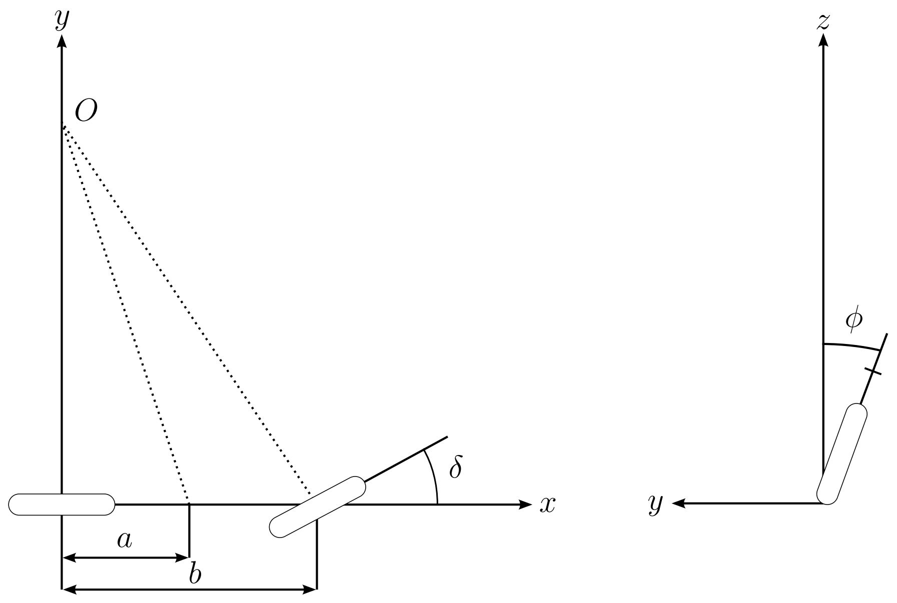


Figure 1: Side View



(a) Top View (b) Rear View

Figure 2: Geometric Definitions

2.2 Second Order Model

The simplest model of a bicycle leads to a second order differential equation relating the steer angle δ to the lean angle ϕ. It does not take the dynamics of the front fork into account and assumes zero trail, a 90° head angle, and a constant forward speed. Even though the model is limited to due its simplifying assumptions, it is useful for both intuitive understand of bicycle stability, as well as for controller design.

2.2.1 Derivation

The first step in deriving the second order model is to consider a bicycle that is rotating about a fixed center O with angular velocity ω_z. Using the method of instantaneous centers, this can be calculated to be

$$\omega_z = \frac{V}{b} \tan(\delta).$$

Then, computing the angular momentum of the system:

$$\underline{L} = \mathbf{I} \cdot \underline{\omega} = \begin{bmatrix} I_x & 0 & I_{xz} \\ 0 & I_y & 0 \\ I_{xz} & 0 & I_z \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ 0 \\ \omega_z \end{bmatrix}$$

Extracting only the component about the x-axis and letting $D = -I_{xz}$ and $J = I_x$ gives

$$L_x = J\dot{\phi} - D\frac{V}{b}\tan(\delta).$$

Taking derivatives, whilst noting that a constant forward speed is assumed, leads to

$$\dot{L}_x \approx J\ddot{\phi} - \frac{D}{b}V\dot{\delta}\sec^2(\delta).$$

Now the torques acting on the system are required, which can be summed and used in conjunction with Newton's second law for moments,

$$\dot{L}_x = \sum_{i=1}^{N} \tau_i.$$

In this model, it is assumed that *two* torques are acting on the system. The first being due to gravity and a non-zero lean angle,

$$\tau_g = mgh\sin(\phi).$$

The second torque is due to circular motion and can be written as follows:

$$\begin{aligned} F_c &= m\omega_z^2 r = \frac{mV^2}{b}\tan(\delta) \\ \tau_c &= F_c h\cos(\phi) \\ &= \frac{mV^2 h}{b}\tan(\delta)\cos(\phi) \end{aligned}$$

Summing both torques results in the final moment balance:

$$J\ddot{\phi} - \frac{DV}{b}\dot{\delta}\sec^2(\delta) = mgh\sin(\phi) + \frac{mV^2h}{b}\tan(\delta)\cos(\phi) \tag{1}$$

This is the final form of the governing second-order, non-linear differential equation of motion and will be used for simulation. To make this model useful for controller design, it needs to be linearised about zero lean and zero steer angle by using the small-angle approximations. Collecting terms, the linearised equation of motion then becomes,

$$J\ddot{\phi} - mgh\phi = \frac{VD}{b}\dot{\delta} + \frac{mhV^2}{b}\delta. \tag{2}$$

Now taking Laplace transforms, the transfer function for the second order bicycle model becomes:

$$G(s) = \frac{\bar{\phi}(s)}{\bar{\delta}(s)} = \frac{VD}{Jb}\frac{s + mhV/D}{s^2 - mgh/J} \tag{3}$$

Lastly, the moment of inertia terms can be approximated as $D \approx mah$ and $J \approx mh^2$, which leads to the final version of the linearised, second order model,

$$G(s) = \frac{Va}{hb}\frac{s + V/a}{s^2 - g/h}. \tag{4}$$

This transfer function, relating the steering angle δ (in degrees) to the lean angle ϕ (in degrees), will be used as the basis for controller design and is *strongly* dependent on forward speed V. It is composed of a variable gain, variable left-half plane zero, and a fixed pair of symmetric real poles with one in the left-half plane and one in the right-half plane. Notice that imposing a steer angle in one direction, will cause the bicycle to lean in the *opposite* direction. The stability properties of this model can now be investigated.

2.2.2 Stability

Even though the model is substantially simplified, it provides a good insight into the stability properties of the bicycle. Four key aspects are as follows:

1. **Always unstable**: The poles are fixed and do not vary with forward speed. They are always symmetric about the imaginary axis, with one stable and one unstable pole.

2. **Speed-dependent gain**: The effectiveness of applying a steer angle input decreases as speed decreases. Therefore, controllability of the system decreases as well. Far larger inputs are needed at lower speeds, and smaller inputs are required at higher speeds.

3. **Speed-dependent zero**: The zero present in the system is moved further into the left-half plane as the forward speed increases. This in effect pulls the root-locus further into the left-half plane as well, making the system easier to stabilise.

4. **Center of mass**: The poles are located at $\pm\sqrt{g/h}$. Since g is fixed, the location of the center of mass above ground h is the only variable that changes the locations of the poles. Thus, the instability of the bicycle decreases as h increases, with the limiting case of marginal stability when $h \to \infty$. This is analogous to the stabilisation of a vertical beam: a longer beam is easier to stabilise.

Furthermore, a simple proportional controller of the form $\delta = -k_\delta \phi$ can stabilise the bicycle in theory if and only if the gain k_δ satisfies the following condition:

$$k_\delta > \frac{bg}{V^2} \tag{5}$$

Which shows that an ever-increasing gain is needed as the forward speed decreases. This, of course, is practically infeasible and shows that it is impossible to balance a stationary bicycle via just controlling the handlebars. Interestingly, a longer wheel base (b) in turn requires the feedback gain to be larger as well.

2.3 Fourth Order Model

A natural evolution from the second order model is the fourth order model, which takes the dynamics of the front fork into account. This model can explain why the bicycle can keep itself self-stable under certain conditions. However, the control input is now the *torque* acting on the handlebar as opposed to the steer angle. Thus, this model is not suited for control design for two reasons: firstly, driving the handlebar motor in torque is not feasible, since both motor current and voltage measurement, as well as knowledge of the internal motor parameters, is needed to estimate the torque. Secondly, if not most importantly, using a servo at the handlebars essentially eliminates the front fork dynamics by enforcing a steer angle. Therefore, the second order model will be used for controller design.
Nonetheless, this fourth order model is incredibly useful in describing the self-stable dynamics of a rider-less bicycle and will be briefly investigated here. A full derivation is given by Papadopoulos in [5]. The model takes the following standard form:

$$\mathbf{M} \cdot \ddot{\underline{q}} + \mathbf{C}_1 \cdot v \cdot \dot{\underline{q}} + (\mathbf{K}_0 + \mathbf{K}_2 \cdot v^2) \cdot \underline{q} = \begin{bmatrix} 0 \\ \tau_\delta \end{bmatrix}$$

Where $\mathbf{M}$, $\mathbf{C}$, and $\mathbf{K}$ are mass, damping, and stiffness matrices respectively. The vector $\underline{q}$ is composed of $\begin{bmatrix} \phi & \delta \end{bmatrix}^T$.

Alternatively, this can be converted to a four-state, state-space representation. For the full-scale bicycle, this results in the following system matrices:

$$\dot{\underline{x}} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 17 & -0.76 - 1.9v^2 & -0.51v & -1.1v \\ 1.4 & 10 - 0.60v^2 & 5.1v & -1.3v \end{bmatrix} \underline{x} + \begin{bmatrix} 0 \\ 0 \\ -0.57 \\ 5.8 \end{bmatrix} \tau_\delta$$

Where the state vector is given by $\underline{x} = [\phi, \dot{\phi}, \delta, \dot{\delta}]^T$.

2.3.1 Stability

In comparison to the second order model given previously, the fourth order model does not give us a direct insight into factors that make the bicycle more or less stable. However, it does show the fact that when the front fork dynamics are taken into account, the bike will be self-stable across a certain range of forward speeds. In addition, it describes various modes apparent in the bicycle dynamics. This is illustrated in Figure 3 below, which shows the real parts of the eigenvalues of the system dynamics across a range of forward speeds for the full-scale bicycle.

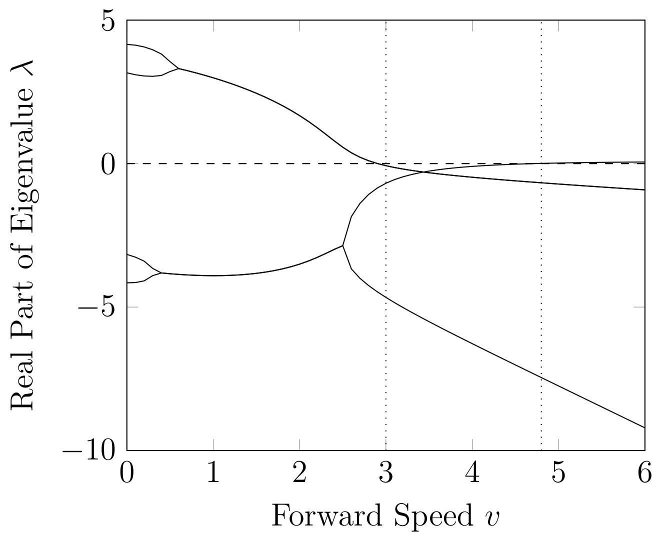


Figure 3: Pole Locations vs Forward Speed

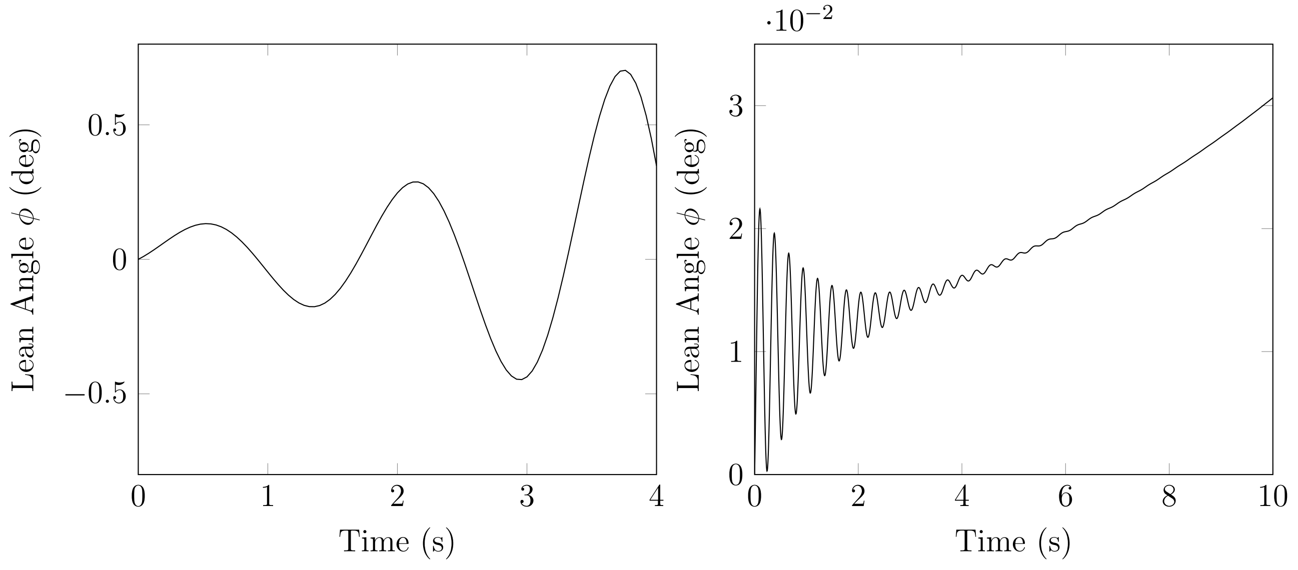
The plot can be divided into three sections:

1. **Unstable and weaving**: Up to approximately $3ms^{-1}$, the bike is unstable, as two of the four poles are in the right-half plane. The poles start off as four distinct, real poles and form into two complex-conjugate pairs. These complex poles give rise to an oscillatory, weaving motion, where the frequency of oscillation is dictated by the magnitude of the imaginary components of the poles.

2. **Self-stable**: From $3ms^{-1}$ to roughly $4.8ms^{-1}$, the bike is self-stable with all poles in the left-half plane. This stability is due to the fact that as the bicycle leans over the fork is now able to steer into the direction of the fall due to its geometry, thus *catching* the bike from from falling down further and restoring it to an upright position.

3. **Capsize**: Contrary to popular belief, increasing the speed does not improve stability indefinitely. After $4.8ms^{-1}$, a real pole again moves into the right-half plane. However, this pole is very slow and thus the system is easier to stabilise via feedback compared to slower speeds. This mode is commonly termed the *capsize* mode.

The weave and capsize modes can be excited by a lean torque disturbance when the bicycle is travelling at the relevant forward speed. This is shown in Figure 4 below for the full-scale bicycle model.



(a) Weaving ($v = 2.8\mathrm{m\,s^{-1}}$) (b) Capsizing ($v = 10\mathrm{m\,s^{-1}}$)

Figure 4: Modes of Full-Scale Bicycle

The oscillations of the unstable weave mode dominate the response in a). On the other hand in b), the high frequency oscillations due to the stable complex-conjugate poles die out quickly with the very slow, non-oscillatory capsize mode dominating the response and finally causing the bicycle to fall over. Notice that even after ten seconds, the lean angle has only deviated by a couple hundredths of a degree. As stated before, this mode is therefore very easily stabilised.

2.4 Parameter Estimation

Python script

The estimated parameters for the Lego prototype can be found in Table 2 below.

Parameter	Symbol	Lego	Full-Scale
Mass	m	0.76kg	28kg
Center of Mass (x)	a	0.10m	0.26m
Center of Mass (z)	h	0.11m	0.55m
Wheel Base	b	0.23m	1.1m
Moment of Inertia (xz)	D	$0.0084\text{kg}\,\text{m}^2$	$8.3\text{kg}\,\text{m}^2$
Moment of Inertia (x)	J	$0.0092\text{kg}\,\text{m}^2$	$3.9\text{kg}\,\text{m}^2$

Table 2: Lego Prototype and Full-Scale Bicycle Parameters

2.4.1 Transfer Functions

Using the estimated bicycle parameters for the Lego prototype and full-scale bicycle in conjunction with the second order model, the transfer functions in terms of forward speed can be given for both the Lego prototype, as well as the full-scale bicycle.

Lego Prototype

$$G_{Lego}(s) = 3.4 \cdot V \cdot \frac{s + 10 \cdot V}{s^2 - 74} \tag{6}$$

Full-Scale Bicycle

$$G_{FS}(s) = 0.44 \cdot V \cdot \frac{s + 3.8 \cdot V}{s^2 - 18} \tag{7}$$

2.5 Actuators

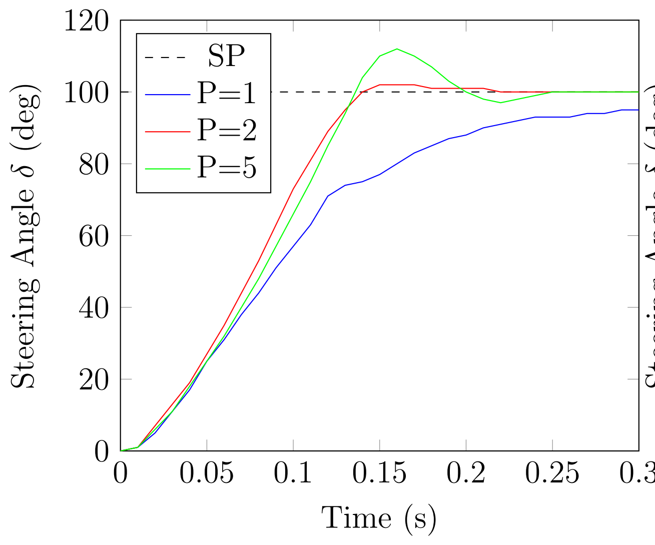
The modelling of control actuators is also of vital importance and needs to be included in the overall system representation. This is due to the fact that actuators cannot change their position instantaneously and therefore will introduce lag into the system, which may cause further instability.

The actuator models were identified by applying a step input to the handlebar servos and recording the corresponding output response. It is important to mention that the step responses were under *loaded* conditions, meaning that the servos were mounted on the front forks, supporting the full weight of the bicycle.

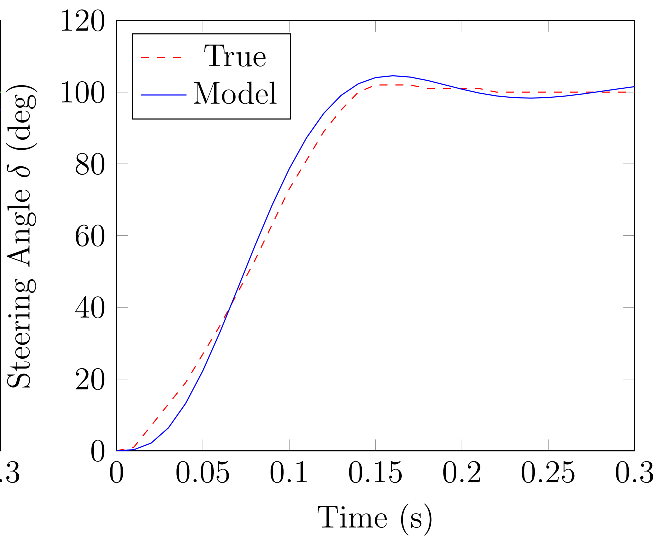
An initial guess was made at the form of the transfer function, such as the model order and form. Then, the Matlab *System Identification Toolbox* was used to identify the locations of the poles and zeros, as well as the magnitude of any possible delays.

2.5.1 Lego Prototype

Interfacing with the Lego handlebar motor can only be achieved by issuing a speed command, not an absolute position command. This meant that it was necessary to implement an additional controller to set the motor position. Luckily, an encoder was already fitted to the servo, meaning that positional measurement for feedback was available. After experimentation, it was decided to use a proportional-only controller, since it satisfied the requirements and was easy to implement. The step responses for a set of proportional gains can be seen in Figure 5 a) below. A proportional gain of 2 was chosen to give a critically damped response, with a fast rise time, and minimal steady-state error.



(a) True Servo Responses



(b) Model Verification (P=2)

Figure 5: Lego Prototype Servo Responses

The closed-loop, proportional-control motor system was given a step input of magnitude 100 degrees. The *System Identification Toolbox* was then used to fit transfer function parameters to the acquired step response data. By iteration, a third order model, consisting of a real pole and an under-damped pair of poles, of the following form was chosen:

$$G_a(s) = \frac{1}{(T_1 s + 1) \cdot (T_w^2 s^2 + 2\zeta T_w s + 1)} \tag{8}$$

Where $T_1 = 0.056$, $T_w = 0.030$, and $\zeta = 0.37$. The identification algorithm suggested a near 95% match between true data and simulated data from this third order model. A comparison between the true and model step responses, visually confirming a good agreement, can be seen in Figure 5 b).

This servo model is then cascaded with the bicycle model to give an overall more accurate representation of the real-world system.

2.5.2 Full-Scale Bicycle

The same process can now be repeated for the full-scale bicycle. However, direct positional control of the motor was already implemented by the manufacturer and thus the step response data could be directly acquired.

Again, a third order model was fitted to the step response data, giving the following parameters: $T_1 = 0.07$, $T_w = 0.073$, and $\zeta = 0.48$. The comparison between the step response of the true system and the model can be seen in Figure 6 below, again indicating a near-ideal fit.

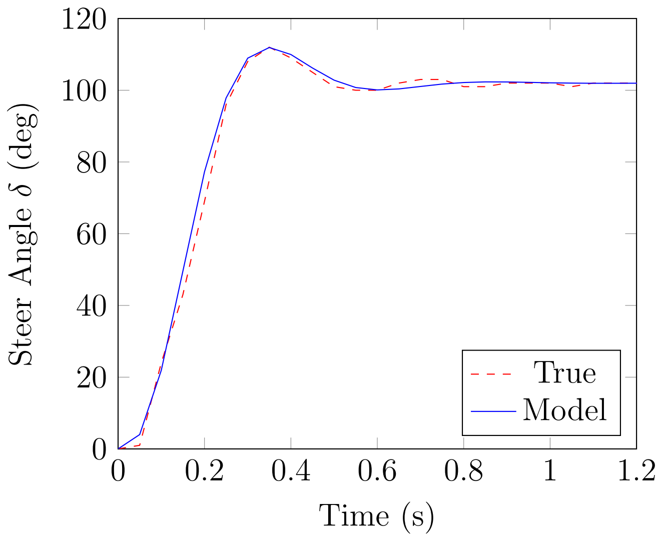


Figure 6: Full-Scale Bicycle Servo Step Response

2.6 Model Verification

Since the bicycle is an unstable system, accurate model verification without a stabilising controller is near impossible. Even with a stabilising controller in place, it is challenging to attain any kind of step or frequency response data. Therefore, model verification will be undertaken implicitly by analysing if the theoretically obtained control laws do indeed give stability when implemented on the real-world system.

3 Controller Design and Simulation

SISO design, only can actuated handlebars, single output which is lean angle.. Currently only using lean angle feedback, to keep on straight path would need some way of knowing heading - future work?

Simulation used non-linear model to determine performance away from equilibrium point, was also compared to linear model. But plots only show non-linear simulation.

3.1 Performance Requirements

When designing a new control system, it is important to be aware of any requirements and specifications that need to be met as to tailor the control law to those needs. In the case of the rider-less bicycle, the main goal is to ensure stability across a relatively broad range of forward speeds.

In addition, should stability be achieved, it is desirable to have a controller that is focused on reducing the effects of disturbances acting on the bicycle, as opposed to tracking the reference. Disturbances could be as small as an uneven road or path that the bicycle is riding over, and as large as a strong gust of wind giving the bicycle a sideways push (which is equivalent to a lean angle disturbance). As a further consideration, the amount of overshoot should be limited, so as not to excite any unmodelled non-linearities by deviating too far from the linear operating point the controller was designed for.

Cross-over needs to be above frequency of RHP pole, as we need the phase lead provided by the pole to ensure an encirclement of the -1 point on the Nyquist diagram.
Overall however, the bicycle, as an under-actuated system with only a handlebar actuator, is an incredibly difficult system to control. This therefore limits the control performance one can expect and reduces the number of performance constraints that can be placed on it.

3.2 Operating Point

As the bicycle models are *strongly* dependent on forward speed, it needs be to ensured that that the controllers can cope well with changes in velocity. However, to simplify controller design and to reduce the amount of symbolic expressions, an operating point is selected (i.e. a specific forward speed) and controllers designed for that specific operating point. This process can then be repeated over a large range of forward speeds and then during the final implementation, the controllers can be gain-scheduled using gain look-up tables.

For the Lego prototype, the maximum achievable speed was found to be $V = 0.44ms^{-1}$, which is thus chosen as the operating point. An increase in speed would be desirable to enable better control performance, however this was limited by the motors available.

For the full-scale bicycle, the operating point was empirically selected to be $V = 3ms^{-1}$. This was chosen to be deliberately low for safety reasons, as not to damage the bicycle or any surrounding objects, should the bicycle become uncontrollable. Additionally, a control system that works for low speeds is harder to achieve, and therefore any increase in speed will make the control design easier.

3.3 PID

A first controller design using a Proportional-Integral-Derivative (PID) controller was investigated, for which the block diagram, including actuator $G_a(s)$ and plant dynamics $G(s)$, can be seen in Figure 7 below. PID controllers are used extensively in control engineering for their simplicity and overall good performance in a multitude of scenarios.

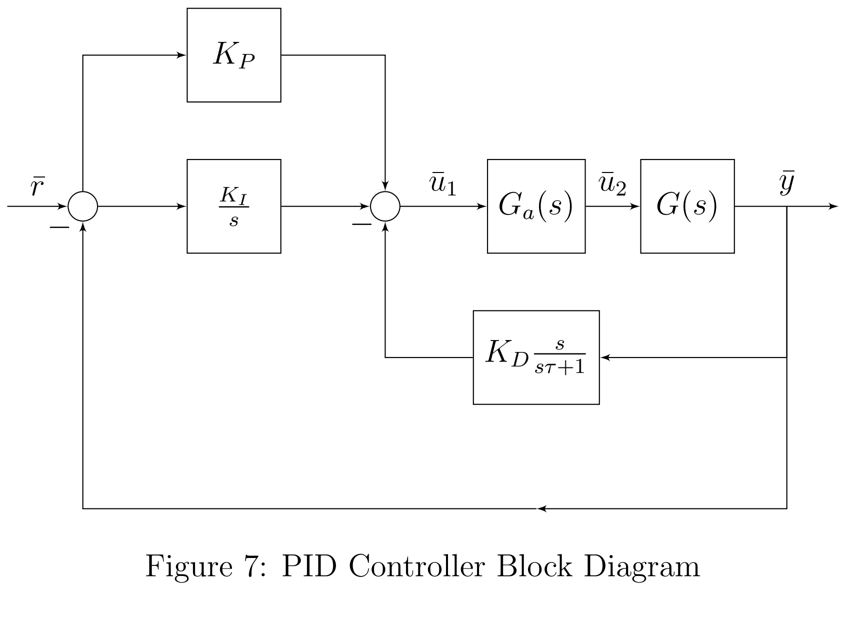


Figure 7: PID Controller Block Diagram

From the block diagram it can be seen that a band-limited differentiator is used, since a pure differentiator is not realisable in practice as the gain increases indefinitely as the frequency increases. Additionally, a pure differentiator amplifies high-frequency sensor noise in the system, which in turn greatly decreases control performance and actuator lifespan. Thus, a practical band-limited differentiator flattens off the gain at a cross-over frequency (in $rads^{-1}$) given by τ^{-1}. It can be seen as a pure differentiator followed by a first-order low-pass filter.

3.3.1 Lego Prototype

Firstly, we wish to achieve closed-loop stability. From equation 5, which gives us the minimum proportional gain required to stabilise the bicycle, we see that we can theoretically stabilise the bicycle with solely a proportional controller with gain:

$$K_P > 11.4 \tag{9}$$

However, this is without considering the effects of the actuator. Cascading the plant with the actuator gives the following root locus diagram:

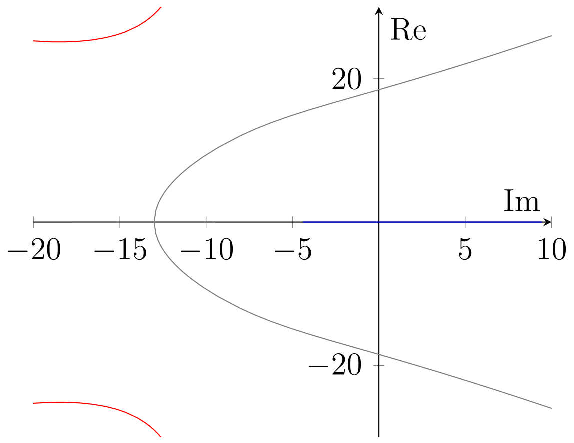


Figure 8: Root Locus for $K_P > 0$

As can be seen from the plot, we now also have an additional upper bound on the gain, as the pair of complex conjugate poles move towards and into the right-half plane as K_P increases further. Thus, the closed-loop system is stable for a proportional gain in the range of:

$$11.4 < K_P < 14.3 \tag{10}$$

After choosing a proportional gain, both an integral gain and derivative gain can be chosen using the same root-locus approach, be reforming the controller and plant into the respective Evan's form.

3.4 Linear Quadratic Regulator

Optimal linear control was investigated by means of an infinite-horizon linear quadratic regulator (LQR), which aims to minimise the cost function for the state-space realisation $\dot{\underline{x}} = \mathbf{A}\underline{x} + \mathbf{B}\underline{u}, \underline{y} = \mathbf{C}\underline{x}$,

$$J = \int_0^\infty (\underline{x}^T \mathbf{Q} \underline{x} + \underline{u}^T \mathbf{R} \underline{u}) dt.$$

Where $\mathbf{Q}$ and $\mathbf{R}$ are positive semi-definite matrices, which respectively penalise having non-zero states and non-zero control inputs. The minimum cost is achieved by solving the *Control Algebraic Riccatti Equation* (CARE):

$$\underline{0} = \mathbf{C}^T\mathbf{C} + \mathbf{XA} + \mathbf{A}^T\mathbf{X} - \mathbf{XBB}^T\mathbf{X}$$

To give the stabilising feedback law,

$$\underline{u} = -\mathbf{B}^T\mathbf{X}\underline{x}.$$

The second-order bicycle model can be transformed into state-space form using controllable canonical form and results in the following system matrices:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ \frac{mgh}{J} & 0 \end{bmatrix}$$
$$\mathbf{B} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$
$$\mathbf{C} = \begin{bmatrix} \frac{mhV^2}{bJ} & \frac{VD}{bJ} \end{bmatrix}$$

As before, the control input u is the steer angle δ and the output y is the lean angle ϕ. For simplicity, the actuator model was *not* taken into account, as to keep the state dimension low.

Unfortunately, the states are not directly measurable. However, since the rank of the observability matrix $\mathcal{O}(A, C)$ is equal to the state dimension and the system is thus observable, an observer can be used to estimate the internal states of the system, given measurable inputs and outputs. For simplicity, it was decided to use the *Luenberger* observer given by,

$$\dot{\underline{\hat{x}}} = \mathbf{A}\underline{\hat{x}} + \mathbf{B}\underline{u} + \mathbf{L}(\underline{y} - \underline{\hat{y}}).$$

Where $\underline{\hat{x}}$ denotes the state estimate. The rate of change of the estimation error $\underline{\dot{e}} = (\mathbf{A} - \mathbf{LC})\underline{e}$ is determined by the eigenvalues of the matrix $\mathbf{A} - \mathbf{LC}$. Thus, $\mathbf{L}$ must be appropriately chosen to ensure that the estimation error approaches zero at a sufficiently fast rate.

The state estimates of the observer can then be used in conjunction with the state-feedback law given by the LQR.

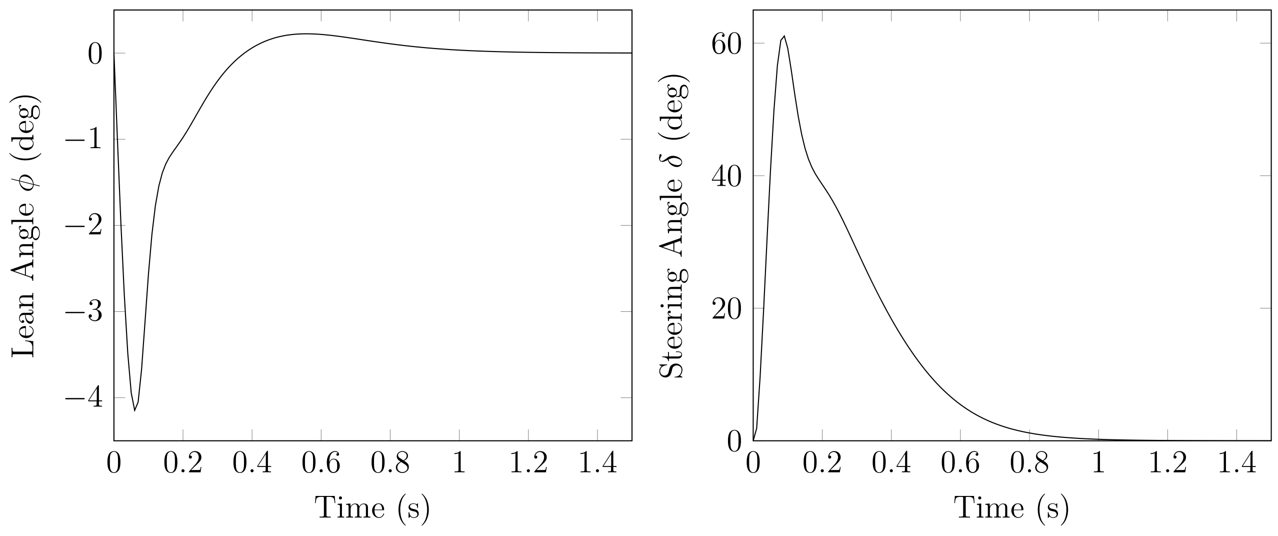
The choice of $\mathbf{Q}$ and R, which for a single-input system is a scalar, in general is not trivial. However, some constraints can be placed on the magnitudes of both. For the bicycle, it is desirable to reduce actuator effort, which is achieved by increasing the magnitude of R. Rapid stabilisation in turn is brought into effect by increasing the elements of $\mathbf{Q}$, thus heavily penalising a non-zero state.

3.4.1 Lego Prototype

Values for $\mathbf{Q}$ and R for the Lego prototype model were chosen by iteration. The solution to the CARE and thus the resulting state-feedback gains were computed using the Matlab command `lqr`. In addition, the observer gain matrix $\mathbf{L}$ was selected via pole placement and was iteratively found to give the best estimation results when the poles are located at $-2, -4$:

$$\mathbf{L} = \begin{bmatrix} 0.69 & 1.0 \end{bmatrix}^T$$

It was observed that changing the values of both $\mathbf{Q}$ and R over a broad range resulted in minimal changes in the controller gains. Thus, $\mathbf{Q}$ and R were simply set to the identity matrix and 1 respectively. The resulting response to an input disturbance is shown in Figure 9 below.



(a) Lean Angle Response (b) Controller Effort

Figure 9: Lego LQR Controller Response

3.4.2 Full-Scale Bicycle

The same **Q** and **R** matrices were chosen for the full-scale bicycle, resulting in control performance that can be seen in Figure XXX below.

3.5 H_∞ Loop-Shaping

The second order model of the bicycle is an extremely simplified formulation of the actual system dynamics and parameters of the bicycle are only approximately known. Furthermore, the plant changes significantly with forward speed. Thus, it is important to maximise the robustness of the closed-loop system to plant perturbations, so that stability is achieved for a broad range of uncertainties. H_∞ loop-shaping is concerned with robust stabilisation and thus an ideal candidate for this problem. The aim is to find a stabilising controller that maximises robustness, whilst trying to achieve a desired loop-shape of the return ratio. This section briefly covers aspects of the theory and then describes the prototype controller design process using this method.

For a general transfer function matrix $G(s)$, which is analytic in the right-half plane and satisfies $\sup_{Re(s)>0} |G(s)| < \infty$, the H_∞ norm of the system is defined as,

$$\|G(s)\|_\infty = \sup_\omega \bar{\sigma}(G(j\omega)).$$

Where $\bar{\sigma}$ denotes the maximum singular value. For a single-input, single-output system the problem reduces to finding the supremum of the magnitude for all ω.
Since the bicycle transfer function used for controller design has poles in the right-half plane, it is not in H_∞. Therefore, a normalised co-prime factorisation of the plant must be found, so that

$$G_0(s) = \tilde{M}^{-1}\tilde{N}$$
$$\tilde{M}\tilde{M}^* + \tilde{N}\tilde{N}^* = I.$$

With $\tilde{M}$ and $\tilde{N}$ both in H_∞ and no common zeros in the extended right-half plane. Here, the factorisation is left-coprime. By then considering uncertainties in the coprime factorisations (Δ_M and Δ_N, both in H_∞), the uncertain plant can be modelled as:

$$G_\Delta = (\tilde{M} + \Delta_M)^{-1}(\tilde{N} + \Delta_N)$$

with

$$\|[\Delta_M, \Delta_N]\|_\infty \leq \epsilon.$$

Then, by the *Small Gain Theorem*, it can be shown that the closed-loop system is internally stable for all $\|[\Delta_M, \Delta_N]\|_\infty \leq \epsilon$ if and only if,

$$b(G, K) = \left\| \begin{bmatrix} K \\ I \end{bmatrix} (I - GK)^{-1} \begin{bmatrix} I & G \end{bmatrix} \right\|_\infty \geq \epsilon.$$

$b(G, K)$ is known as the stability margin. H_∞ loop-shaping aims to find a stabilising controller K that maximises $b(G_0W, K)$, where the weighting function $W(j\omega)$ shapes the return ratio as desired.

In practice, this is achieved using the Matlab *Robust Control Toolbox*, where functions are available for computing normalised coprime factorisations (`ncfmr`), as well as the optimal, stabilising controller using the *Glover-McFarlane* method (`ncfsyn`) proposed in [6].

Furthermore, it may be more convenient to use the ν-gap metric (denoted by $\delta_\nu(G_0, G_1)$) to check for robustness. In general, a gap between two plants G_0 and G_1 gives the smallest value of $\|[\Delta_M, \Delta_N]\|_\infty$ that perturbs G_0 into G_1. Thus, if the closed-loop system is stable for G_0 and K, then K will also stabilise G_1 if and only if,

$$\delta_\nu(G_0, G_1) < b(G_0, K).$$

Thus, after having computed a stabilising H_∞ optimal controller, the ν-gap metric can be used to determine the range of forward speeds, as well as the maximum uncertainties in system parameters, that can be tolerated to maintain closed-loop stability.

Table 3 below gives numerical values for the ν-gap between the nominal plant, again including the actuator model, and expected perturbations in plant parameters.

Parameter	**Variation**	**ν-gap**
Forward speed v	$\pm 30\%$	0.046,0.0615
Mass m	$\pm 5\%$	negligible
Center of mass h	$\pm 20\%$	0.057,0.047
Wheel Base	b	$0.23m$
Moment of Inertia	D	$0.0084kgm^2$
Moment of Inertia	J	$0.0092kgm^2$

Table 3: Lego Prototype Parameters

3.5.1 Lego Prototype

The frequency response of the nominal system, including the actuator model, is shown in Figure 10 below.

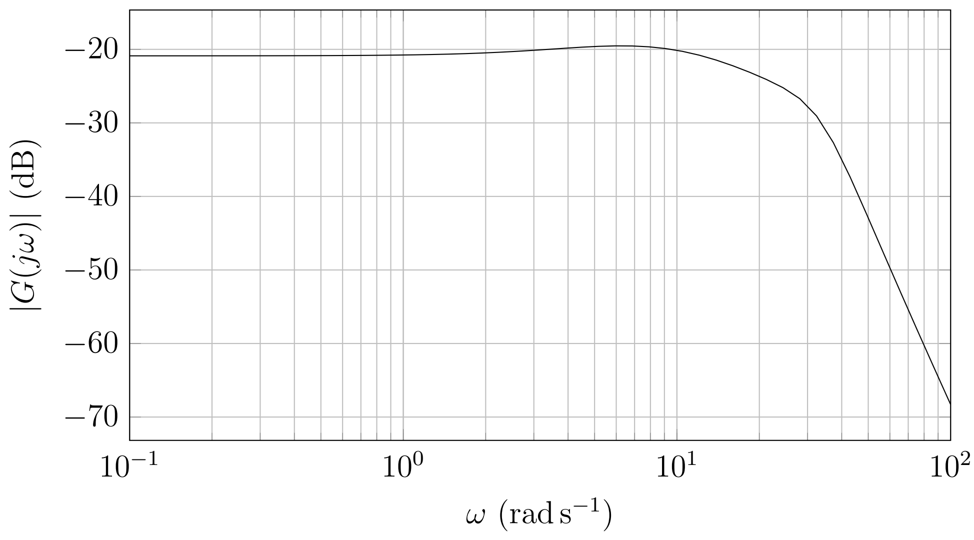


Figure 10: Bode Magnitude Plot for Lego Prototype Model

As previously mentioned, it is clear that a proportional gain of $K_P > 11.4$ is needed for stability at this nominal forward speed. Thus, as a first step, a proportional gain of $K_P = 12$ is included in the weighting function $W(j\omega)$. The plant can now be further *shaped* so that the closed-loop system has desirable properties, such as the typical requirements of reduced sensitivity at low frequencies (for good tracking and disturbance rejection) and reduced complementary sensitivity at high frequencies (to reduce the effects of sensor noise). Additionally, it is important to note that actuator limits must be taken into account, as not to make the demands on the system too large. This can be avoided by ensuring that the loop gains are not made arbitrarily large and that the target bandwidth of the system is bounded reasonably.

An appropriate weighting function, to give $b(G_0, K) \geq 0.2$, was then found by iteration and is comprised of three elements:

1. *Proportional Gain*: The gain of 12 is included in the weighting function before the controller is synthesized. Aids the controller in maintaining stability.

2. *Integrator*: Improves disturbance rejection and tracking by providing gain at low-frequencies.

3. *Zero*: Increases the bandwidth of the system and ensures the slope of the magnitude

flattens off around the cross-over frequency. This helps improve the phase-margin, by keeping the phase away from -180° at 0dB.

The overall weighting function is therefore,

$$W(s) = 12 \cdot \frac{s+1}{s}.$$

The frequency response of the shaped plant, in comparison to the nominal plant, is shown in Figure 11 below.

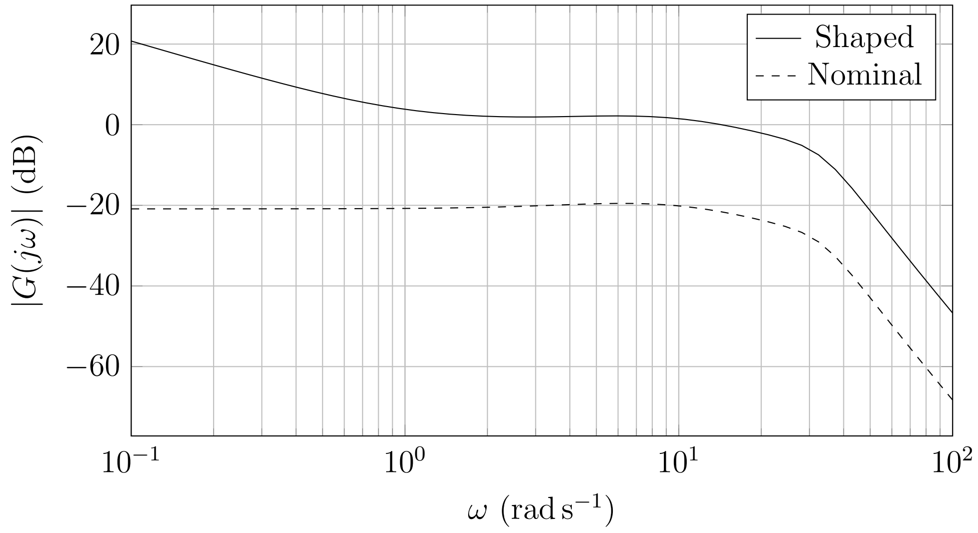


Figure 11: Bode Magnitude Plot for Shaped Lego Prototype

Using Matlab to then synthesize the controller gave a stability margin of $b(G_0, K) = 0.24$ with a sixth-order controller of the following form:

$$K(s) = W(s)K_\infty(s) = \left(12 \cdot \frac{s+1}{s}\right)\left(4 \cdot \frac{(s+0.17)(s+8.9)(s+18)(s^2+25s+1110)}{(s+1.1)(s+4.3)(s+48)(s^2+32s+2370)}\right)$$

3.6 Non-Linear Methods

As opposed to having several linear controllers gain-scheduled for different operating points, want a single controller that works sufficiently well at *any* forward speed.

3.7 Comparison

Comparison of control schemes, LQR difficult because control performance not only depends on sensors but also on observer, PID simplest to implement and tune on the fly, H-infinity

4 Implementation: Lego Prototype

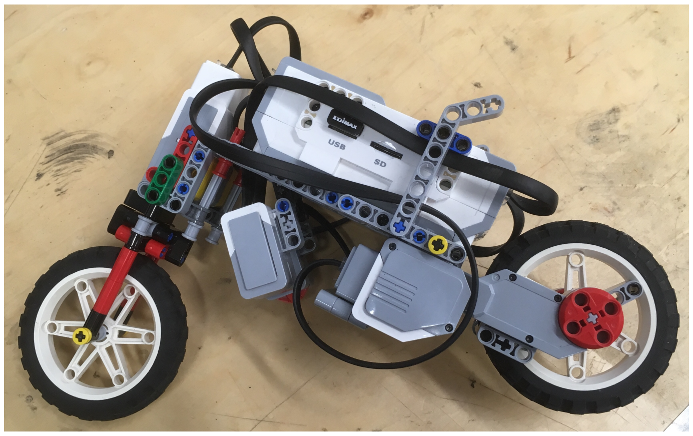


Figure 12: Lego Prototype Bicycle

4.1 Hardware

Two drive motors to give maximum speed (to aid stability), limited by single gyro sensor (measuring rate of change of lean angle), limited by maximum speed

4.2 Software

RobotC, Choice of sample time (dependent on bandwidth of system)

4.2.1 Lean Angle Estimation

The Lego bicycle is only equipped with a gyroscopic sensor, meaning that only the rate of change of the lean angle is directly measurable. Direct integration of this measurement to give an estimate of the lean angle is not possible, since the measurements are corrupted by noise and the gyro has a time-varying bias. This noise and gyro bias will cause a drift in the estimated lean angle over time and thus deviate quickly from the true lean angle. Therefore, it is necessary to estimate the lean angle at every timestep using a continuous-discrete *Extended Kalman Filter* (EKF). The EKF fuses the approximately-known, linearised, continuous system dynamics with discrete, noisy sensor readings to give an overall improved state estimate. The state vector and model dynamics are taken from the fourth-order linearised model given in section 2.3. Note that this entire state estimation step would not be necessary if an accelerometer were available, which could be used as reference to correct for drift.

The continuous-discrete EKF can be divided into two steps. Firstly, the *prediction* step uses the system dynamics to estimate the state vector a timestep T in the future:

$$\begin{aligned}\underline{\hat{x}} &= \underline{\hat{x}} + T \cdot \mathbf{A} \\ \mathbf{P} &= \mathbf{P} + T \cdot (\mathbf{AP} + \mathbf{PA}^T + \mathbf{Q})\end{aligned}$$

Where $\underline{\hat{x}}$ is the state estimate vector, $\mathbf{A}$ is the linearised state-transition matrix, $\mathbf{P}$ is the error covariance matrix, and $\mathbf{Q}$ is the model noise matrix.

Following this, the *update* step fuses the linearised dynamics with the noisy sensor readings:

$$\begin{aligned}\mathbf{K} &= \mathbf{PC}^T(\mathbf{R} + \mathbf{CPC}^T)^{-1} \\ \mathbf{P} &= (\mathbf{I} - \mathbf{KC})\mathbf{P} \\ \underline{\hat{x}} &= \underline{\hat{x}} + \mathbf{K}(\underline{y} - h(\underline{\hat{x}}, \underline{u}))\end{aligned}$$

Where $h(\underline{\hat{x}}, \underline{u})$ is the output function, $\mathbf{C}$ is the Jacobian of the output function, $\mathbf{R}$ is the measurement noise matrix, and $\mathbf{K}$ is the Kalman gain, which effectively weights the measurements and internal model predictions.

Lastly, it is important to note that an approximate model of the dynamics is used in conjunction with a noisy, biased sensor. Even though the resulting state estimation performance is much improved over using only the gyroscope readings, the inevitable inaccuracies will accumulate and degrade the lean angle as time progresses. This of course has a direct impact on control performance and also only allows the final system to be tested for short periods of time before requiring a reset.

4.2.2 Discrete Form of PID

To be able to implement a PID controller in software, the continuous-time transfer function needs to be transformed into discrete-time. This is achieved by using the Tustin (Bilinear) transform, which gives the best possible match in the frequency domain between continuous and discrete forms. Furthermore, stability is preserved between the mappings. We start with the transfer function of a *parallel* type PID representation:

$$K_a(s) = K_P + \frac{K_I}{s} + K_D \frac{s}{s\tau + 1}$$

To discretize this transfer function, we now apply the Tustin mapping from the s to the z-domain:

$$s \to \frac{2}{T}\frac{z-1}{z+1}$$

Where T is the sample time in seconds. After substituting and rearranging, this leads us to the discrete transfer function representation of the PID controller:

$$K_d(z) = K_P + K_I \frac{T(z+1)}{2(z-1)} + K_D \frac{2(z-1)}{z(2\tau + T) - (2\tau - T)}$$

The terms are intentionally kept separated, as in software these can be seen as three distinct blocks which are then summed to give the controller output. The discrete integrator and differentiator can then be implemented simply by difference equations with constant coefficients (assuming a constant sample time T).

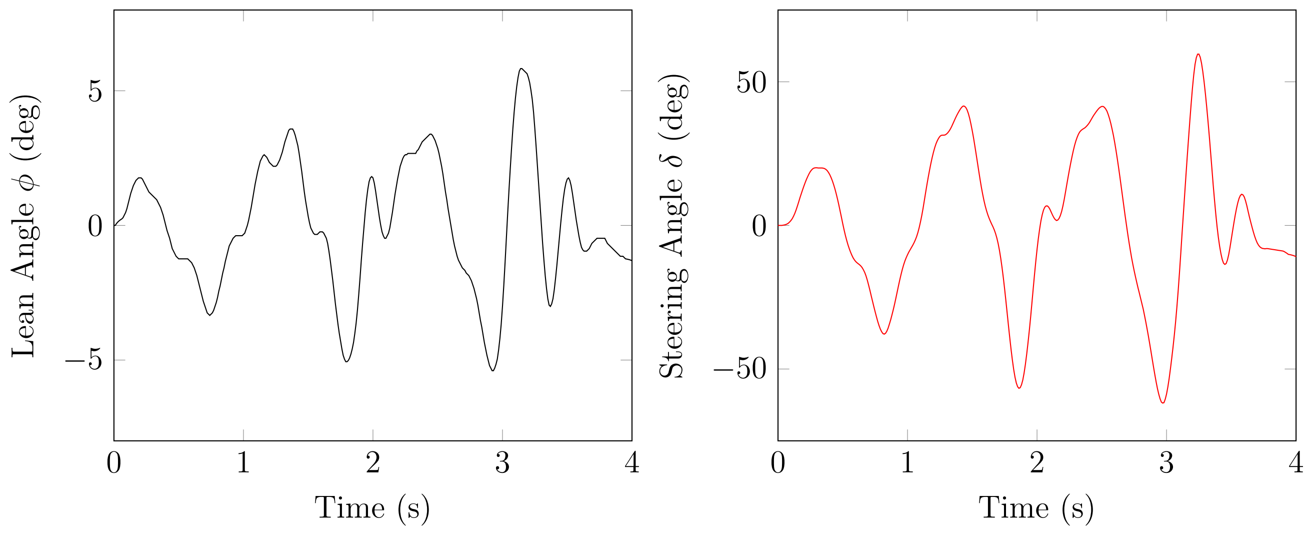
Integrator wind-up is prevented by clamping. The output is also clamped to prevent the actuator from moving beyond fixed limits.

4.3 Results

4.3.1 Proportional Controller

A proportional-only controller was indeed able to stabilise the bicycle. As predicted by the second-order model and simulation, the bicycle could be stabilised at the given forward speed of $V = 0.44ms^{-1}$ by a gain greater than 11.4.

Figures 13 a) and b) show an excerpt of the response and controller effort of the Lego prototype for a proportional gain of $K_P = 12$.



(a) Lean Angle Response

(b) Actuator Effort

Figure 13: P-Controller Response ($K_P = 12$)

The response for the simple proportional controller is certainly not ideal, even though stability is achieved. The magnitude of the lean angle is bounded by approximately 5°, which is an acceptable deviation from the desired setpoint of 0°. However, there is an oscillation present in the response, indicating that the system is on the boundary of stability. Knocking the bicycle gently, as to add an output disturbance, destabilises it. Thus the closed-loop system is not robust and sensitive to disturbances in its current state. This is to be expected however, since we are using a proportional gain which places the closed-loop poles very close to the imaginary axis. As shown, inevitable model uncertainties will deteriorate the control performance from the ideal calculations and simulations. Furthermore, this *jittering* behaviour also causes the bicycle to move off-track easily and not follow a straight path. Lastly, the controller effort is larger than ideal. However, this large actuator command is required for stability.

As a brief aside, the servo model is again verified by running a simulation in Matlab but using the real-world controller output data. As can be seen in Figure 14, the match is near perfect and confirms the actuator model identified in an earlier section of this report.

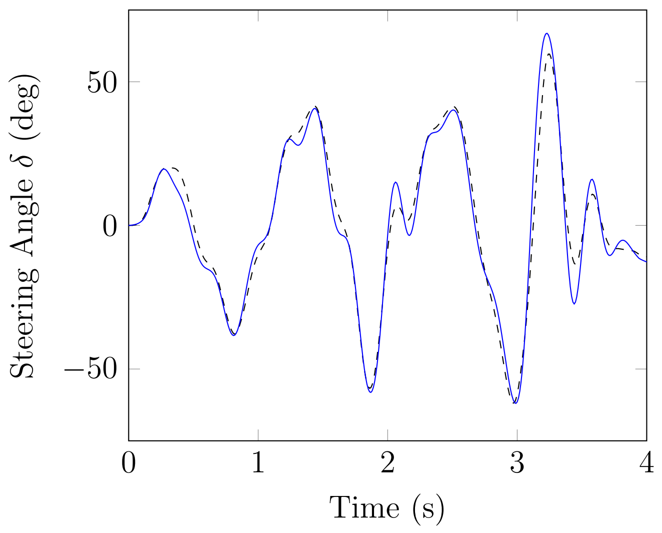


Figure 14: Lego Servo Model Verification

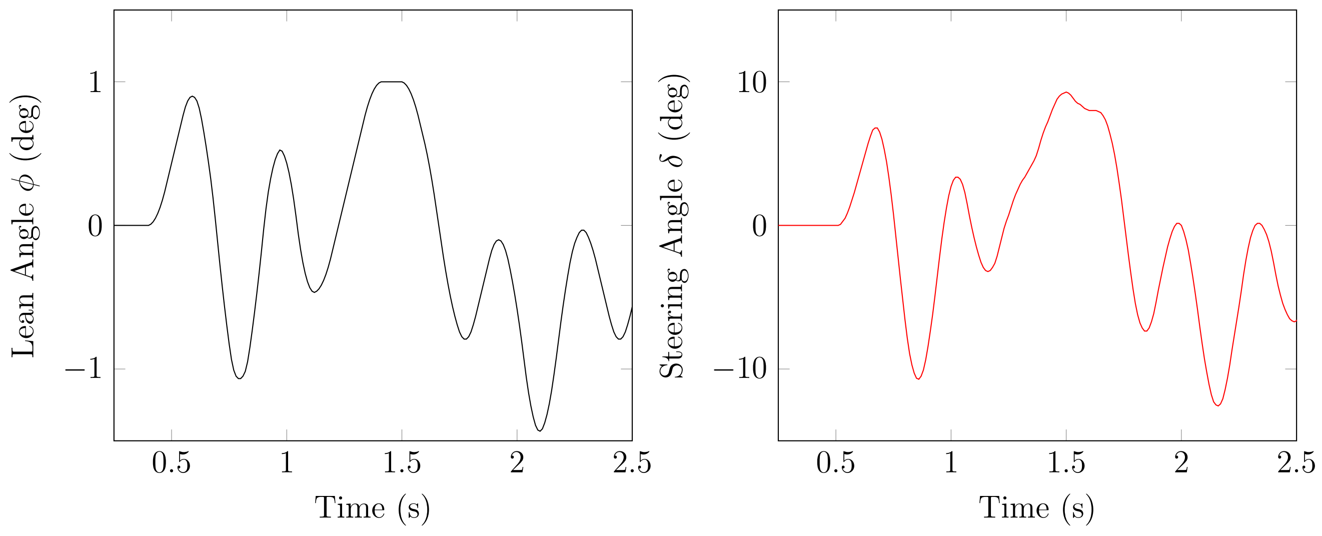
4.3.2 Proportional-Derivative Controller

As a next step, we will aim to improve control performance by adding phase lead to the controller, as discussed in the section on controller design. This is achieved by the use of a lead compensator. This can also be seen as a parallel combination of a proportional gain and a band-limited differentiator. The lead compensator increases the phase margin of the system in the hope of reducing the oscillations currently present in the closed-loop response. HOW DO WE SPLIT THIS BETWEEN CONTROLLER DESIGN AND IMPLEMENTATION???

$$K_{Lead}(s) = 14 \cdot \frac{s + 1.1}{s + 2.0} \tag{11}$$

Or in PD form,

$$K_{PD}(s) = 8 + 3\frac{s}{s \cdot 0.50 + 1}. \tag{12}$$



(a) Lean Angle Response

(b) Actuator Effort

Figure 15: PD-Controller Response ($K_P = 8$, $K_D = 3$, $\tau = 0.5$)

Small oscillations still remain but better damped , controller effort is greatly reduced for a better performance! Derivative is heavily filtered (tau=0.5), so that the jiterring which would otherwise occur due to noise is redueced. Lean angle only just above 1 deg.

4.3.3 Final Observations

hard to stabilise since h is small and time constants are short, not robust to disturbances. Have hardly any time to react. Additional limitation is maximum speed of lego model, which causes it to be very unstable. Stabilisation can be achieved but not a very good performance. Performance could probably be improved if we were to use a better angle estimation system, i.e. use an IMU. As soon as gyro estimate starts drifting we're in trouble (which turns out to be pretty quickly)...

5 Implementation: Full-Scale Bicycle

Having successfully stabilised prototype bicycle...

(a) Unmodified Form (b) After Modification

Figure 16: Full-Scale Bicycle Before and After Modification

5.1 Hardware

Arduino (32-bit instead of 8-bit) as controller, why particular servo, servo mounting design, battery, voltage regulators, drive motor, motor controller disassembly, etc.

5.1.1 Bicycle Frame

5.1.2 Drive Motor

5.1.3 Handlebar Actuator

5.1.4 Microprocessor and Sensors

5.1.5 Miscellaneous

5.2 Software

5.2.1 Code Layout

Loops at different sample times: IMU, controller, telemetry, etc.

5.2.2 Lean Angle Estimation

Again, it was necessary to estimate the lean angle from raw sensor data. In the case of the full-scale bicycle however, an *inertial measurement unit* (IMU) was available, comprised of three gyroscopic sensors and three accelerometers, each aligned along an orthogonal axis.

This meant that gyroscopic drift could now be directly accounted for without having to resort to the computationally expensive Kalman filter used for the Lego prototype.

At rest, an accelerometer outputs a signal directly proportional to the gravitational vector, this can then be used as a reference to correct for gyroscopic drift. However, when in accelerated motion, the accelerometer output will be *corrupted* by other accelerations terms, such as linear or Coriolis.
The problems present in both accelerometers and gyroscopic sensors are alleviated by the use of a complementary filter. Simply put, an accelerometer is useful over a long duration of time, while a gyroscopic sensors useful over a short period of time. It is therefore possible to low-pass filtere Combining both favourable aspects of these sensors leads to the use of a complementary filter.
Complementary filter, gyro HPF and acc LPF, more stable since we have accelerometer to correct for drift

5.2.3 Basestation and Telemetry

Packet handling, vary controller parameters on the fly, live data logging and display, safety start/stop, etc..

5.3 Results

6 Conclusions and Future Work

For the actual real-world implementation, controller design is actually a small part of the work. Need to worry about acquiring good estimates of states, writing all the peripheral software, drivers, sizing actuators, modelling the system, estimating parameters, etc, etc.
A controller that works in theory actually takes quite a lot of effort to work in practice.
Extremely difficult to stabilise this system, made even harder due to underactuation and difficulty in verifying model. Possibly could try adding flywheel to provide lean torque. Maybe better modelling.

References

[1] Francis J. W. Whipple. *Stability of the Motion of a Bicycle.* The Quarterly Journal of Pure and Applied Mathematics, Vol XXX, 1899.

[2] A. L. Schwab, J. P. Meijaard, J. M. Papadopoulos. *A Multibody Dynamics Benchmark On The Equations of Motion of an Uncontrolled Bicycle.* 5th Nonlinear Dynamics Conferencec, 2005.

[3] Karl J. Åström, Richard E. Klein, Anders Lennartsson. *Bicycle Dynamics and Control: Adapted bicycles for education and research.* IEEE Control Systems, 10.1109/MCS.2005.1499389, September 2005.

[4] International Standard ISO 8855. *Road Vehicles - Vehicle Dynamics and Road-Holding Ability - Vocabulary.* Second Edition, December 2011.

[5] J. M. Papadopoulos, R. S. Hand, A. Ruina. *Bicycle and Motorcycle Balance and Steer Dynamics.* Draft, July 1990.

[6] Duncan McFarlane, Keith Glover. *A Loop Shaping Design Procedure Using H_∞ Synthesis.* IEEE Transactions on Automatic Control, Vol. 37 No. 6, June 1992.

Appendix

A Risk Assessment