Modeling and Control System for an Auto-Balancing Bicycle

Gyan Prayaga*

Grinnell College

Dated: March 24, 2021

Abstract

Bicycles are commonplace but the mechanics of their stability is less commonly understood. In designing a mechanism for stabilizing a falling bicycle, we first sought to model the dynamics of an idealized bicycle as a mass-rod system. We found that the reaction time τ needed to stabilize the perturbed bicycle was less than the human reaction time. We subsequently constructed a computational representation of a bicycle in Python which could be visualized and manipulated. Finally, we designed a control system using a flywheel mounted about the bicycle's frame, which could be accelerated by a motor to re-stabilize the bicycle at small tilt angles.

1 Introduction

Learning to ride a bicycle is a cherished milestone for many people. Graduating from a tricycle to training wheels exposed us, kinesthetically and psychologically, to physical forces which threatened to tip us over. We noticed that when we picked up speed, our training wheels did less work. Once the training wheels came off, these intuitions carried over into balancing the bike ourselves. By pedaling faster and straightening the steering, we empowered the bicycle's innate gyroscopic capability in keeping us upright.

*Independent study supervised by William Case and Charles Duke

Some experienced riders might not even need to keep their hands on the steering. Once they've done the necessary steering, they can place their hands behind their back, relying on a careful shifting of weight with their core muscles to keep the bike upright. The careful balancing of several hundred pounds of distributed mass with a combination of intuition and rapid learning is no doubt a momentous human feat.

Yet what if this feat is not uniquely human? Could we train a robot to balance its own bike, ingesting sensor data for velocity and tilt rather than the qualia of a human? We could begin by creating a computational "shadow" of a bicycle. Once we enter our training environment, we can apply various conditions to this bicycle; e.g. change the tilt or steering angle; and watch how our model responds. If the response results in an unbalanced bicycle, we have failed.

In response to an unbalanced bike, a human rider can correct the bicycle's posture by:

1. Shifting their body weight in the other direction
2. Accelerating (by upshifting and/or pedaling faster)
3. Decelerating (by downshifting and/or pedaling slower)
4. Turning the handlebars
5. Braking

The rider could apply one or more of these mechanisms. An experienced human rider does not analyze each of these responses; rather, they come to them intuitively having honed their behavior over thousands of previous iterations.

A robot, however, is limited in its physical response. If we intend for this robot to be synonymous with the bicycle itself, hence making the bicycle self-balancing (analogous to a self-driving car), then we should expect for the mechanics of re-balancing to be largely internal and hidden from view. This opens up the bicycle to traditional use cases (human rider) as well as new ones (distracted or occupied human rider, cargo transport, food delivery, last-mile delivery, etc.). The lightweight delivery of food and cargo in cities could be particularly useful from a commercial standpoint.

Let us assume, for simplicity, that a self-balancing bicycle has a single gear electric motor transmission, a common convention in electric cars. This means it is fit-for-purpose for the self-balancing use case and does not allow traditional use by a human rider.

Thus, when unbalanced, a robotic bike can correct its posture by:

1. Accelerating (increasing energy to motor)
2. Decelerating (decreasing energy to motor)
3. Turning the handlebars
4. Braking

However, one shortcoming of a self-balancing robotic bike is the inability to shift its posture like a human rider could. Numerous factors influence the stability of a riderless bicycle, including steering geometry and the minimal role of the gyroscopic effect (*Jones*). This project will seek to examine if a flywheel mounted above the bicycle's frame wheel could generate enough torque to re-balance a perturbed bicycle.

2 Theory

We can begin to think about a bicycle's dynamics from first principles. Furthermore, we can break down the motion of a bicycle into several smaller investigations.

2.1 Characteristic Time for a Falling Bicycle

Before we attempt to stabilize the bicycle, we want to investigate if, when a bicycle falls, there is even enough time for a stabilizing mechanism to engage and re balance the bicycle.

A bicycle, in its most idealized form, can be described as a mass m affixed to a rod which is balancing on the ground. The rod has an equal distribution of m_{rod} which is negligible compared to the point mass affixed to it (the "rider").

The mass-rod system is in a state of unstable equilibrium. The characteristic time τ is the reaction time needed to correct the bicycle's perturbation before it falls.

In order to find the characteristic time, we first want to solve for theta's acceleration.

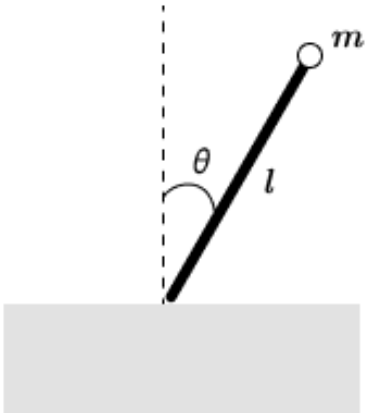

Figure 1: Falling rod

$$\ddot{\theta} = \frac{N}{I} \tag{1}$$

,

for torque N and moment of inertia I.

The torque is:

$$N = lmg\theta \tag{2}$$

for small angles of the rod's tilt angle θ

The moment of inertia I for a point mass is:

$$I = ml^2 \tag{3}$$

Our equation for $\ddot{\theta}$ now looks like:

$$\ddot{\theta} = \frac{g}{l}\theta \tag{4}$$

where l is the length of the rod and m is the mass of the rider

We are looking for the characteristic time τ. When $t = \tau$, our value for theta will equal e, right before the angle exponentially grows. We are looking for some coefficient A of our differential equation such that $C(\theta) = \ddot{\theta}$. Let us twice derive the equation with respect to time:

$$\theta = e^{t\sqrt{\frac{g}{l}}} = e^{\frac{t}{\tau}} \tag{5}$$

$$\tau = \sqrt{\frac{l}{g}} \tag{6}$$

Let us assume the rod is 2 meters in length. Then,

$$\tau = \sqrt{\frac{2}{9.8}} = 0.45s$$

The human reaction time is about 0.1 seconds, and so a characteristic time of 0.45 seconds appears to show that identifying the bicycle's *falling* state before it falls over (and cannot be stabilized) is a feasible task. It seems safe assume that the reaction time of an accelerometer would be superior to that of a human.

This calculation rests on a few assumptions. First, that the mass of the rod is negligible compared to that of the point mass (the "rider"), and thus the rod can be disregarded. Second, the moment of inertia I is approximate (an idealized bicycle frame) and thus it is largely a function of the center of mass.

2.2 Stabilizing the bicycle with a flywheel

Now that we have found how long it will take for the bicycle to fall over, we are curious to see how a falling bicycle may be stabilized. We hypothesized that one approach to stabilizing a bicycle is to attach a small flywheel to the top of its frame, which could produce enough angular acceleration to counteract the bicycle's fall for small angles. In this simplified model, we imagined the bicycle as an upright rod connected to a stabilizing disk, which is propelled

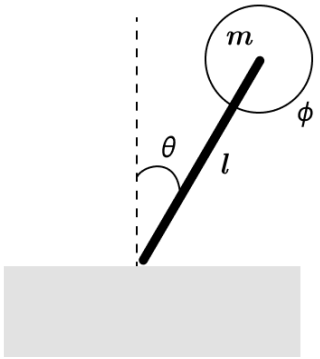

Figure 2: Flywheel on rod

by a stepper motor. *Figure 2* thus looks very similar to *Figure 1*, except with a disk instead of a point mass attached to the rod.

We first setup our system by computing the torque on the rod.

$$\tau = \frac{dL}{dt} \tag{7}$$

The equation for angular momentum is

$$L = I\omega \tag{8}$$

The angular momentum generated by the system (rod and flywheel) is:

$$L = I_{rod}\dot{\theta} + I_{flywheel}\dot{\phi} \tag{9}$$

where ϕ is controlled by a motor.

$$I_{rod} = ml^2 \tag{10}$$

for a rod of mass m and length l

$$I_{flywheel} = \frac{1}{2} m_d r^2 \tag{11}$$

for a disk of mass m_d and radius r

We derive the angular momentum with respect to time to find the system's net torque:

$$N = \frac{dL}{dt} = ml^2\ddot{\theta} + I\ddot{\phi} \tag{12}$$

Let us setup the left-hand side of the torque equation like so:

$$N = r \times F \tag{13}$$

The only external force on the disk-flywheel system is gravity, so the system's torque is:

$$N = r \times F_g = rF_g \sin\theta \tag{14}$$

We are only concerned with stabilizing small angles of theta, so we can approximate $\sin\theta = \theta$. Our full torque equation now becomes:

$$lmg\theta = ml^2\ddot{\theta} + I\ddot{\phi} \tag{15}$$

We can rearrange this equation to solve for $\ddot{\phi}$, the required flywheel acceleration:

$$\ddot{\theta} = \frac{g}{l}\theta - \frac{I_{flywheel}}{ml^2}\ddot{\phi} \tag{16}$$

for a given tilt angle θ and $\dot{\theta}$.

We want our flywheel to control the rod such that the rod's motion is that of a damped harmonic oscillator, which "swings back" to unstable equilibrium. This is represented by this second-order differential equation:

$$\ddot{\theta} = -\gamma\theta - \zeta\dot{\theta} \tag{17}$$

where ζ is the damping coefficient

This second-order differential equation can be split into two first-order equations, which are used in *Method* to find values for θ and $\dot{theta}$ over a time series:

$$\dot{\theta} = \omega(t) \tag{18}$$

$$\dot{\omega} = -\zeta\omega - \gamma\theta \tag{19}$$

The γ value can be arbitrarily set, and will determine the size of the necessary flywheel acceleration. If the resulting flywheel acceleration is too large, we can modulate γ accordingly. For a harmonic oscillator, γ is found by:

$$\gamma = \omega^2 \tag{20}$$

for angular frequency ω.

Critical damping would provide the quickest return to unstable equilibrium ($\theta = 0$). To find the damping coefficient ζ, we can use the equation:

$$\zeta = 2\omega_0^2 \tag{21}$$

for initial tilt velocity ω_0.

3 Method

After exploring the theory of a falling bicycle through separate investigations, we build a computational replica of a bicycle. This took the form of a *Bicycle class* in Python, which provided a public interface with methods *tilt*, *rotate*, and *assemble*. The assembly method was responsible for "drawing" the bicycle in 3D space.

3.1 Bicycle assembly

The digital bicycle was composed of four structural components: 1) the front wheel, 2) the rear wheel, 3) the steering column, and 4) the frame axle.

The rear wheel was constructed first by placing the contact point of the rear wheel on the origin. This was done because the relative location of the rear wheel never changes with transformations.

Dictionaries for the bicycle's structure matrices and point vectors were stored in the Bicycle class and can be accessed publicly using the *map* method. The structure matrices are the sets of all vectors for each constituent part of the bicycle. When the bicycle is tilted or a specific part of the bicycle is rotated, a method iterates through each structural vector and manipulates it accordingly.

The steering column assembled has a zero rake and a relatively small trail.

3.2 Rotating a bicycle part

First, the method iterates over the individual vectors of the part's structure matrix. Second, a unit vector is created for the specified rotation axis. A 3D vector codirectional to the axis of rotation (the "rotation vector") is produced using SciPy's spatial transform package. The rotated vector is computed from the dot product of the rotation vector and the given vector. After all vectors of the matrix have been rotated, the method saves the updated structure matrix in the *Bicycle* class.

Our goal was to be able to apply various transformations to the bicycle or its subcomponents and then see how important points or characteristics of the bicycle had changed.

3.2.1 Computing the trail

Trail is an important physical characteristic of a bicycle, and also affects its stability. Negative or small trail tends to diminish stability, while very large trail does the same. In his article, *Jones* describes the role of trail on the bicycle's stability in great detail.

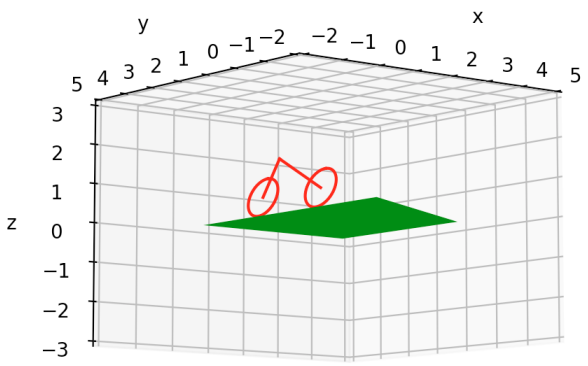

Figure 3: Visualization of bicycle structure matrix with tilt

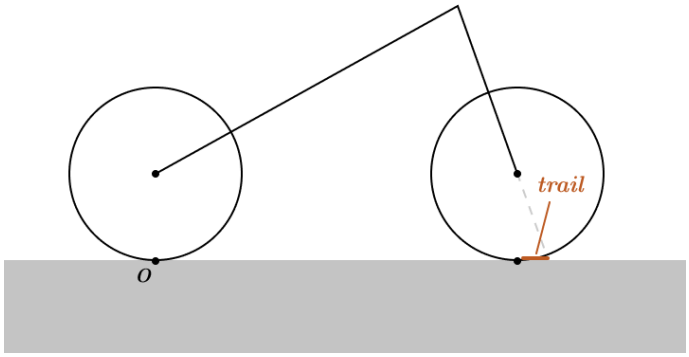

Figure 4: 2D bike structure with points and trail indicated

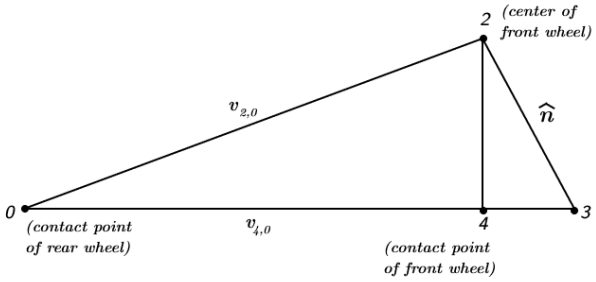

Figure 5: Finding the trail

Geometrically, trail is the distance between the contact point of the front wheel and the contact point of the steering axis. Using the above diagram, we can solve for trail d using the three given points and by finding an additional point: the intersection of the steering axis onto the ground plane. Using simple vector math, we see that trail t is found by:

$$t = \vec{v}_{30} - \vec{v}_{40} \tag{22}$$

where $\vec{v}_{40}$ is already known

To solve for for $\vec{v}_{30}$:

$$\hat{k} \mid \vec{v}_{30} = \vec{v}_{2\theta} + \hat{n}d \tag{23}$$

Which can be rewritten as:

$$0 = \hat{k} \cdot \vec{v}_{20} + \hat{n} \cdot \hat{k}(d) \tag{24}$$

where $\hat{n}$ is the steering axis

Rearranging to solve for d:

$$d = -\frac{\hat{k} \cdot \vec{v}_{20}}{\hat{n} - \hat{k}} \tag{25}$$

for steering axis $\hat{n}$

Substituting our equation for d back into the equation for $\vec{v}_{30}$:

$$\vec{v}_{30} = \vec{v}_{20} - \hat{k}(-\frac{\hat{k} \cdot \vec{v}_{20}}{\hat{n} - \hat{k}}) \tag{26}$$

Having found $\vec{v}_{30}$, we can now compute the trail using Eq. 22. The actual trail computation is done in the *Bicycle class* using the steps outlined here.

3.2.2 Calculating flywheel acceleration over a time series

Flywheel acceleration can be calculate computationally according to the following process. First, *Equation 17* is solved for θand over a time series. We can then find the required flywheel acceleration by computing *Equation 16* for each θ and $\dot{\theta}$ over the time series.

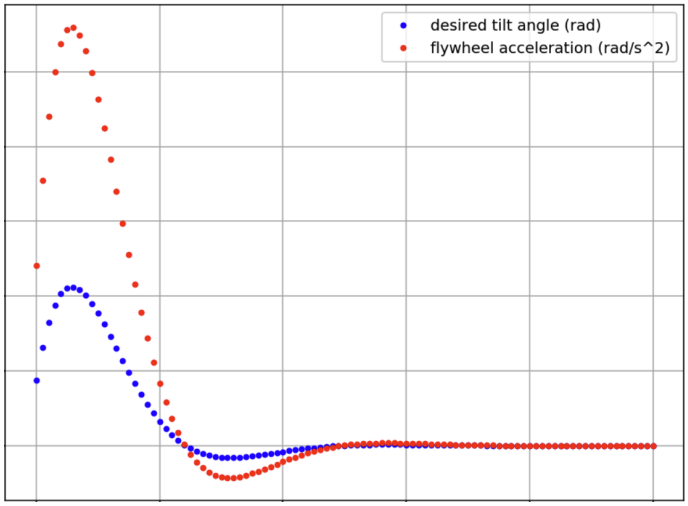

Figure 6: Normalized relationship between flywheel acceleration $\ddot{\phi}$ and desired bicycle tilt θ

3.2.3 The role of the center of mass

The digital bicycle was instantiated with an arbitrary center of mass 3D vector which is manipulated alongside the structure vectors. This means that a tilt transformation applied to the bicycle will also result in a new center of mass. We can compute the "relative center of mass" by calculating the difference between the ${}_0 and \mathrm{COM}$ vectors.

After collating the COM vector returned by the Bicycle class' *map* method for various tilt angles θ, we can construct the function:

$$COM(\theta) \tag{27}$$

for any tilt angle θ. This gives us insight into the bicycle's stability at varying inclinations.

4 Discussion

There are a number of additional investigations which deserve further time and attention. First, this research focused largely on the stationary bicycle case, but a moving bicycle presents its own challenges. One could explore the role of centripetal force on the bicycle's steering, which is related to its stability.

In our simulation, we assume that the flywheel acceleration responds instantaneously to the perturbed bicycle frame. In a real bicycle, however, there would be a slight delay from

the moment the bicycle is tilted to the flywheel's correction, largely due to sensor reading and processing. This means that a bicycle's flywheel will respond to past conditions. We could include this in our simulation by computing the flywheel acceleration for older values of θ and $\dot{theta}$. Then our acceleration function looks like: $\ddot{\phi}_t(\dot{\theta}_{t'}, \theta_{t'})$ for some $t' = t - \delta t$, where δt is the flywheel response "delay".

In the future, a user might feed a "test run" to the Bicycle program with the bicycle's initial tilt angle θ and a time series with corresponding values for θ and steering angle Φ. Then the program would modulate the acceleration accordingly, with the goal of minimizing the use of the flywheel and stabilizing the bicycle through other means, such as the inbuilt steering geometry.

References

[1] Jones, D. E., Physics Today (1970). p.34-40
`https://doi.org/10.1063/1.2364246`

[2] Wilson, D. G., Schmidt, T. (2020).
`https://mitpress.mit.edu/books/bicycling-science-fourth-edition`