Proceedings, Bicycle and Motorcycle Dynamics 2019
Symposium on the Dynamics and Control of Single Track Vehicles,
9–11 September 2019, University of Padova, Italy

Linear and nonlinear controllers of an autonomous bicycle have almost identical basins of attraction in a non-linear simulation

D. E. Meehan*, A. L. Ruina#

Mechanical Engineering
Cornell University
566 Upson Hall, 124 Hoy Rd, Ithaca NY, USA
* e-mail: dem292@cornell.edu
# e-mail: ruina@cornell.edu

ABSTRACT

We compare controllers to stabilize a robotic bicycle. We measure a controller's performance in non-linear simulation, by the size of the set of initial conditions that can be recovered (i.e., the size of the controller's basin of attraction). We only consider initial perturbations in lean and lean rate, taking the initial steer angle as zero. The basin of attraction of a stable controller always includes points on this curve: a) the total lean energy (a lean term and a lean rate term) is equal to the energy of a upright bicycle; and b) lean and lean rate have opposite signs. This is the set of states corresponding to the eigenvector associated with the negative eigenvalue of an inverted pendulum. These are the initial conditions that lead to straight ahead motion, with no control whatsoever. It turns out that, for the controllers we have tried, the basin boundaries are very close to being parallel to this curve. Thus, we characterize the robustness of a controller with a single scalar "Basin Width". In simulation, using a simplified point-mass bicycle model, we compare two control architectures; **1)** an LQR (linear quadratic regulator) controller based on linearized equations of motion; and **2)** a dynamic programming optimal controller found using value iteration on a non-linear model. For both controllers we penalize control actions and deviations from the target state. For the non-linear controller, we add a large penalty for falling, and we constrain both the steer angle rate and maximum steer angle. Both controllers are tested in a nonlinear simulation with constrained steer angles and steer rates. For both controllers, the basin width goes to zero, approximately linearly, as speed goes to zero. For both controllers, as speed gets high, the basin boundary approaches the failed state. For bicycles with more restricted steer rate and steer angle, the value iteration controller performs slightly better than the LQR controller. For bicycles with less restrictive actuator limits, the linear quadratic regulator and value iteration controller have almost equivalent Basin Widths. Our provisional conclusion is that, for practical design purposes, we can limit ourselves to consideration of linear controllers with little cost in system robustness.

Keywords: control, autonomous bicycle, value iteration, basin of attraction, Basin Width.

1 INTRODUCTION

Autonomous bicycles, bicycles that can balance themselves, without a human rider, could be used as part of bike-share system, to deliver goods, or to map roads. In addition, technology from autonomous bicycles could enable balance-assist features for human-rideable safer and easier-to-ride fly-by-wire electric bicycles.

Robotic bicycles are sometimes used as a case-study for feedback controller design: the unstable, non-minimum phase [1] system presents a nontrivial problem. However, the low dimensionality of the problem allows us to try modern exhaustive methods.

Our control design problem is to find a controller that, in practical usage, is maximally resistant to falling down. That is, we want it to be maximally robust (where we use robust in the English language sense, meaning reliable). For our robotic bicycle, avoiding falls takes precedence over other performance metrics.

Ultimately we would like to compare controllers by how well they work in the real world. Short of that, we would like to evaluate controllers using a high-fidelity non-linear model that can be tested with the full range of expected uncertainties in terrain, disturbances, and model properties.

For this initial study, as a proxy for robustness, we use the size of the basin of attraction of the controller when tested on a simple non-linear model, described in the next section.

A natural question to ask is "Is there substantial benefit in using an optimized non-linear controller instead of just a linear controller?" Perhaps the answer to this question, only considering simple bicycle models, will extend to more complex realistic models. To address this question we compare two controllers: one, an optimized non-linear controller, and the other, a classic linear controller.

Reinforcement learning methods have been used in simulation to develop bicycle controllers for balance and navigation [3] and for bicycle stunts [4]. We use value iteration in our non-linear controller. The key, though, is not the method, but that given our objective function, we find an approximation to the theoretically best possible controller.

In contrast to such modern machine-learning type of nonlinear approaches, historically more people have applied linear control theory to balance real autonomous bicycles [5] [6] [7] and a steer-by-wire bicycle [8] using steering and/or thrust control. Indeed, building on previous work [9], the Cornell Autonomous Bicycle Project Team has balanced a physical robotic bicycle using a Linear-Quadratic-Regulator (LQR) controller [10].

We wish to know whether an optimized non-linear controller provides substantially better control than a classical linear controller. If so, applying optimal non-linear control to a real bicycle would enhance robustness. But, instead, if linear control gives nearly equal results, we would have no need for the complexities of, for example, reinforcement learning methods. So, we investigate the potential benefits of a non-linear optimized controller compared to an LQR controller. As noted, because a robotic bicycle must not fall, we measure robustness by the size of the set of initial states from which a controller can recover a bicycle to straight-ahead motion.

2 PROBLEM SETUP

For this initial study, we compare different bicycle balance controllers by their performance on a computer simulation of a bicycle. Our model is the nonlinear, point-mass, Boussinesq bicycle model, the so-called "simplest" bicycle model [11]. We assume zero radius wheels, so we treat the wheels as skates (with no friction along the direction of travel). Figure 1 describes the bicycle model.

For this model, with no trail and no inertia in the steer assembly, we can use steer rate ($\dot{\delta}$) as the control variable. That is, we assume we can control $\dot{\delta}$ an inner steering control loop that is fast enough so that it does not couple to the lean dynamics of the bicycle. So, all of the balance controllers we consider here take the form

$$\dot{\delta} = f(\phi, \dot{\phi}, \delta), \tag{1}$$

Where the function f is the control policy. The Cornell Autonomous Bicycle Project Team has

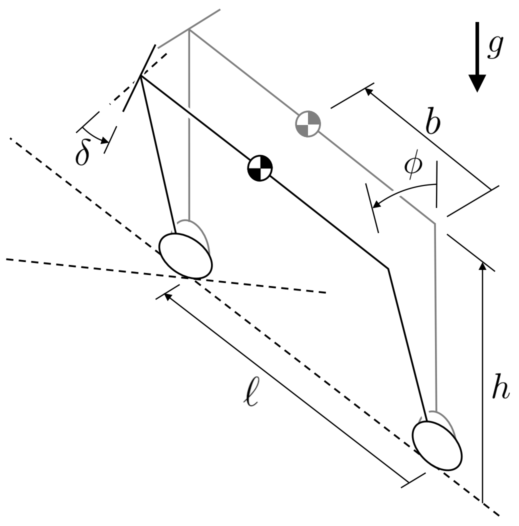


Figure 1. **Point Mass Boussinesq Bicycle Model.** An upright bicycle is shown in gray. A leaned and steered bicycle is shown in black. The relevant dynamical state is set by the lean angle (ϕ), lean rate ($\dot{\phi} \equiv \frac{d}{dt}\phi$) and steer angle ($\delta$). Position on the plane and bicycle orientation are ignorable variables for the balance problem. Parameters ℓ (wheel base), h (height of center of mass), and b (distance from rear wheel to center of mass projected onto the ground) define the bicycle geometry. g is the acceleration due to gravity. The magnitude of the bicycle mass m does not appear in the governing equations.

successfully used controllers of this form [10]. That is, the isolation of time scales between steering and lean does, in practice, seem to allow us to use steer rate $\dot{\delta}$ as a control variable for balance.

Controlling steer rate, as opposed to steer torque, allows us to neglect the steer dynamics of a bicycle. This simplifies the bicycle model. The lean dynamics of a bicycle seem to be well approximated with a point mass model, whereas proper accounting of the steer dynamics would demand a more complicated model. In effect, we overwhelm the natural steer dynamics with an inner steer controller. This has great advantage for robustness of control. For example, a steer torque controller would be more susceptible to model errors: a more complicated model, including steer dynamics, would have more parameters to be estimated. Also, a steer torque controller would be more susceptible to torque disturbances (such as a scrubbing torque between the front wheel and the ground).

The continuous-time equation of motion of the point mass bicycle model is given by Equation 2 below, with variables and parameters defined in Figure 1 [12]. The real-wheel speed is v.

$$\begin{aligned} -h^2\,\ddot{\phi} + g\,h\,\sin\phi - \frac{h\,v^2\tan\delta}{\ell} - \frac{b\,\dot{\delta}\,h\,v}{\ell\cos^2\delta} + \frac{h^2\,v^2\tan^2\delta\tan\phi}{\ell^2} \\ -\frac{b\,h\,\dot{v}\tan\delta}{\ell} - \frac{b\,h\,\dot{\phi}\,v\tan\delta\tan\phi}{\ell} = 0 \end{aligned} \tag{2}$$

The bicycle simulation runs in discrete time with a frequency of 50 Hz, reflecting the frequency of control updates on typical hardware.

3 BALANCE CONTROLLERS

3.1 Linear Quadratic Regulator (LQR)

First we find an LQR (linear quadratic regulator) controller. For a linear system, subject to a quadratic cost function, an LQR controller with a perfect model and perfect state feedback minimizes the total cumulative cost given by Equation 3 [13].

$$J = \int_0^\infty (\boldsymbol{x}^T \boldsymbol{Q} \boldsymbol{x} + \boldsymbol{u}^T \boldsymbol{R} \boldsymbol{u}) dt \tag{3}$$

The LQR design process produces a gain matrix $\boldsymbol{K}$ such that the controller $\boldsymbol{u} = -\boldsymbol{K}\boldsymbol{x}$ minimizes the expression 3. A user provided $\boldsymbol{Q}$ penalizes errors in state and a user provided $\boldsymbol{R}$ penalizes actuator effort.

For us, the state vector is $\boldsymbol{x} = [\phi, \dot{\phi}, \delta]^T$ and the control is $\boldsymbol{u} = [\dot{\delta}]$. To apply the LQR algorithm, we linearize the nonlinear bicycle equation of motion about upright riding. That is, we linearize Equation 2 about $\boldsymbol{x}^* = [0, 0, 0]^T$ and $\boldsymbol{u}^* = [0]$.

This linearization [12] provides a continuous-time linear system described by $\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}$, with

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{g}{h} & 0 & \frac{-v^2}{h\ell} \\ 0 & 0 & 0 \end{bmatrix} \text{ and } \boldsymbol{B} = \begin{bmatrix} 0 \\ \frac{bv}{h\ell} \\ 1 \end{bmatrix}.$$

When designing the cost matrices, we mostly want to penalize lean angle, but make other terms non-zero to keep the controller well behaved. Here are the weights we used for state and control penalties, respectively.

$$\boldsymbol{Q} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.05 & 0 \\ 0 & 0 & 0.05 \end{bmatrix}. \text{ and } \boldsymbol{R} = [0.003].$$

We treat the system as discrete with a period of $T = 0.02s$. We use the MATLAB command `lqrd(A, B, Q, R, T)` to calculate the optimal feedback gains ($\boldsymbol{K}$) of the discrete time feedback controller by specifying matricies $\boldsymbol{A}$ and $\boldsymbol{B}$ of the continuous system.

3.2 Nonlinear, Value-Iteration Controller

We use value iteration to develop an optimal, nonlinear controller for the bicycle. We treat a discrete time version of the bicycle balance problem as an infinite horizon Markov Decision Process (MDP). An MDP consists of a state space S, action space A, state transition function, reward function $R(\boldsymbol{x}, \boldsymbol{u})$, and discount factor γ [2]. Let a state $\boldsymbol{x} \in S$ be defined as before with $\boldsymbol{x} = [\phi, \dot{\phi}, \delta]^T$. Similarly, let $\boldsymbol{u} \in A$ be defined as $\boldsymbol{u} = [\dot{\delta}]$. We assume the bicycle has a *deterministic* state transition function $T(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{x}'$. $T(\boldsymbol{x}, \boldsymbol{u})$ encodes the system dynamics where taking action $\boldsymbol{u}$ in state $\boldsymbol{x}$ moves the bicycle to state $\boldsymbol{x}'$ at the next timestep. γ can discount future rewards, but we do not discount future rewards, so $\gamma = 1$.

We would like the bicycle (an agent) to act optimally given the system's dynamics (an environment). That is, we would like the bicycle to maximize the sum of cumulative rewards.

We set the reward of a given state and action according to Equation 4.

$$R(\boldsymbol{x}, \boldsymbol{u}) = -(1\phi^2 + 0.05\dot{\phi}^2 + 0.05\delta^2 + 0.003\dot{\delta}^2) \tag{4}$$

That is, for a given state and action, the reward equals the negated path cost of the LQR controller (Equation 3). Therefore, a controller which maximizes the cumulative rewards (Equation 5) will minimize the cumulative cost (Equation 3).

Define the utility, sometimes called value, of a state, $U(\boldsymbol{x})$, to be the sum of rewards the bicycle receives starting in state $\boldsymbol{x}$ and acting optimally [14]. That is, if $[(\boldsymbol{x}_0, \boldsymbol{u}_0), (\boldsymbol{x}_1, \boldsymbol{u}_1), (\boldsymbol{x}_2, \boldsymbol{u}_2), ...]$ is an optimal trajectory, Equation 5 defines $U(\boldsymbol{x_0})$ [14] (our notation differs from [14]).

$$U(\boldsymbol{x}_0) = R(\boldsymbol{x}_0, \boldsymbol{u}_0) + R(\boldsymbol{x}_1, \boldsymbol{u}_1) + R(\boldsymbol{x}_2, \boldsymbol{u}_2) + ... \tag{5}$$

The recursive relationship of utilities can be described with the Bellman equation [14] (our notation differs from [14]).

$$U(\boldsymbol{x}) = \max_{\boldsymbol{u}}(R(\boldsymbol{x}, \boldsymbol{u}) + U(\boldsymbol{x}')) \tag{6}$$

where $\boldsymbol{x}' = T(\boldsymbol{x}, \boldsymbol{u})$.

We set the *utility* (not reward) of all fallen states to be a large in magnitude and negative. Specifically, we set the utility of a fallen state to be less (larger magnitude) than the cumulative reward the bicycle would receive if it got the minimum (most negative) possible reward for 200 seconds = 10,000 timesteps. Therefore, the bicycle should, whenever possible, avoid falling.

First, we describe how, given an accurate utility function, the bicycle can act optimally. Second, we describe how to estimate the utility function using value iteration.

3.2.1 Using the Utility Function to Generate a Policy

Given the utility function, $U(\boldsymbol{x})$, and a way to simulate the bicycle, we can find an optimal policy by, at every state, choosing the action that maximizes the reward of the present state, action pair plus the utility of the next state, $\boldsymbol{x}'$, reached by taking action $\boldsymbol{u}$ in state $\boldsymbol{x}$ [2] (our notation differs from [2]).

$$\pi(\boldsymbol{x}) = \text{argmax}_{\boldsymbol{u}}(R(\boldsymbol{x}, \boldsymbol{u}) + U(\boldsymbol{x}')) \tag{7}$$

We encode the utility function as a lookup table from state to utility. To generate the points of the lookup table, we discretize the state space (eg, into 487,123 points). But, the bicycle operates in a continuous state space. So, to estimate the value at any point in continuous state space, we trilinearly interpolate utilities in the lookup table. Trilinear interpolation estimates the value at any continuous state by, for each dimension in turn, linearly interpolating the values at the nearest discrete points in the lookup table [15]. Thus, we can estimate the value of any continuous point in state space.

Given an accurate utility function, the bicycle chooses a control according to Equation 7. To calculate argmax$_{\boldsymbol{u}}$, we use use a nonlinear optimizer, specifically the Python function `scipy.optimize.minimize_scalar`. This requires simulating the bicycle forward one timestep to state $\boldsymbol{x}'$ and interpolating the value lookup table to predict $U(\boldsymbol{x}')$.

The value iteration policy approximates the optimal policy of the nonlinear system. In contrast, the linear quadratic regulator produces an optimal policy for the linearized system. In particular, value iteration accounts for nonlinearties in the nonlinear equations of motion and actuator limits.

3.2.2 Estimating the Utility Function

Given an estimate of the utility function, we can choose the optimal action according to Equation 7. Value iteration estimates $U(\boldsymbol{x})$.

Value iteration, an iterative algorithm, generates progressively better estimates of the utility (value) function using the update rule of Equation 8 [2] (our notation differs from [2]).

$$U_{i+1}(\boldsymbol{x}) \leftarrow \max_{\boldsymbol{u}}(R(\boldsymbol{x}, \boldsymbol{u}) + U_i(\boldsymbol{x}')) \tag{8}$$

where $T(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{x}'$ as before and U_j is the estimate of the value function at iteration j.

When simulating a test of bicycle (as in Section 3.2.1), we only need to calculate the control using Equation 7 at every point along the trajectory. So, when simulating a bicycle, we use a nonlinear optimizer to calculate argmax$_{\boldsymbol{u}}$ (see Section 3.2.1). But, when estimating the value function, we must calculate the optimal policy for all points in state space. Therefore, using a nonlinear optimizer to calculate $\max_{\boldsymbol{u}}$ is prohibitively slow. So, we calculate the optimal action by taking the max of a fixed set of k actions (say $k = 11$). That is, for each of the actions, $\boldsymbol{u}_k$, in the fixed set we calculate $Q_k = R(\boldsymbol{x}, \boldsymbol{u}_k) + U(\boldsymbol{x}'_k)$, where $\boldsymbol{x}'_k = T(\boldsymbol{x}, \boldsymbol{u}_k)$. As $\boldsymbol{x}'_k$ is not a generally one of the points in the utility function lookup table, the utility $U(\boldsymbol{x}'_k)$ is calculated by trilinear interpolation of utility function lookup table. Then, we set $U(\boldsymbol{x}) = \max_k Q_k$.

The actions we consider remain the same for all iterations of the value update rule. Therefore, we can precompute both the reward function of the current state, $R(\boldsymbol{x}, \boldsymbol{u})$, and the transition function to the next state, $T(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{x'}$ for all $\{\boldsymbol{x}, \boldsymbol{u}\}$ pairs before performing the iterative updates. Therefore, during the value iteration update, we do not to calculate any dynamics. In each iteration of the value update, we only compute, using trilinear interpolation and the value estimates at the previous iteration, the estimate of each $U(\boldsymbol{x'})$. Therefore, each value iteration update can be quickly performed with vectorized array computations. We repeat the utility updates until the estimated utility function is sufficiently close to convergence. That is, until the change in the utility function, for all states, between successive updates, is smaller than a threshold.

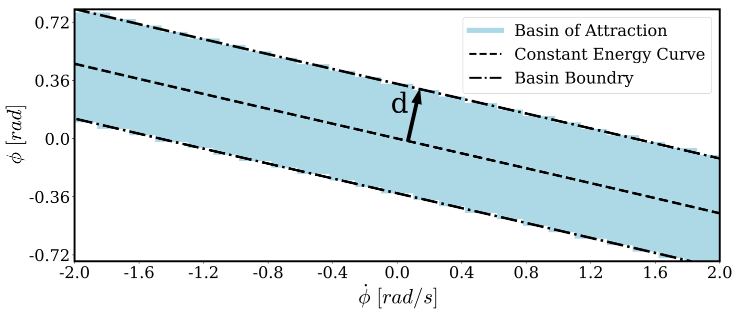
4 BASIN OF ATTRACTION

To evaluate the effectiveness of a given controller, we examine the basin of attraction of the closed-loop system generated by that controller. A controller's basin of attraction is the set of states from which that controller can recover a bicycle to the stable state (upright riding) [16]. While the bicycle has a three dimensional state space, we limit our study to the two dimensions of lean (ϕ) and lean rate ($\dot{\phi}$). The more states a controller can recover from, the better that controller is. Figure 2 plots the basin of attraction for two linear controllers, one at $v = 0.5m/s$ and another at $v = 2m/s$.

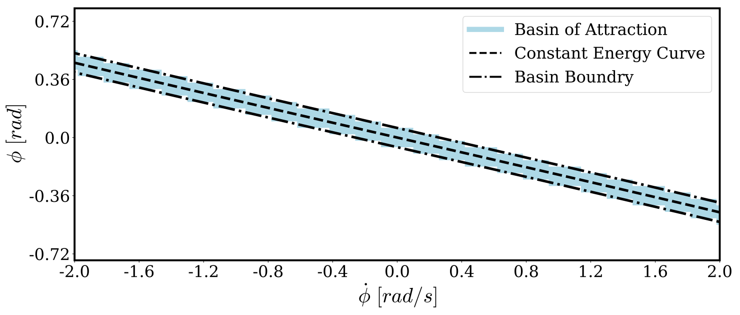
To determine if a state $\boldsymbol{x} = [\phi_0, \dot{\phi}_0, \delta_0]$, is within a controller's basin of attraction, we simulate the bicycle starting from that state. Since we limit our search to ϕ and $\dot{\phi}$, all tests start with $\delta = 0$. See Figure 5a for a trajectory generated for a linear quadratic regulator and value iteration controller.

4.1 Constant Energy Curve

As the bicycle speed decreases, the size of the basin of attraction also decreases (compare Figure 2a to Figure 2b). Indeed, a slow bicycle is harder to stabilize than a fast bicycle. But, the basin of attraction shrinks into a band, not, as one might expect, a ball around the origin.



(a) Basin of Attraction at 2m/s



(b) Basin of Attraction at 0.5m/s

Figure 2. Basin of Attraction for a linear quadratic regulator at $v = 2m/s$ and $v = 0.5m/s$. A constant energy curve is plotted on each figure. Figure 2a shows the Basin Width (d). The basin boundary curves are plotted by shifting the constant energy curve a distance d in both directions perpendicular to the constant energy curve. These boundary lines align with the experimental basin boundaries (the edges of the blue region) found through bicycle simulations.

Specifically, this band of recoverable states surrounds a curve of states with equal energy (see Constant Energy Curve in Figure 2). The states on this curve have energy equal to the energy of an upright bicycle. That is, this curve passes through the origin. There are two such curves in the two dimensional state space of lean (ϕ) and lean rate ($\dot{\phi}$). One curve corresponds to a bicycle leaned in the positive direction and falling in the positive direction (bicycles along this curve fall down). The other curve, of interest, corresponds to bicycles leaned in the positive direction, but falling in the negative direction. So, without any control action (without adding or removing any energy from the system), a bicycle on this curve will fall up to the unstable equilibrium of upright riding. Intuitively, picture a bicycle leaned over, but thrown up at the perfect initial velocity so as to come to a rest exactly at the upright equilibrium point.

The constant energy curve, of interest, is given by Equation 9 with sign(ϕ) $\neq$ sign($\dot{\phi}$). Equation 9 is derived in Appendix A.

$$g\cos(\phi) = g - \frac{h}{2}\dot{\phi}^2 \tag{9}$$

Equation 9 is satisfied in two cases: when ϕ and $\dot{\phi}$ have same sign and when they have opposite signs. So, Equation 9 describes two curves. The curve at the center of the basin of attraction is described by Equation 9 where ϕ and $\dot{\phi}$ have opposite signs.

Equation 10 describes states on the eigenvector, with a negative eigenvalue, of the open-loop linearized system (see Appendix B). To a first order approximation, Equation 9 (the constant energy curve), equals Equation 10 (the eigenvector with negative eigenvalue). So, a first order approximation of the constant energy curve is the eigenvector of the linearized system with negative eigenvalue.

$$\phi = -\sqrt{\frac{g}{h}}\dot{\phi} \tag{10}$$

4.2 Basin Width

To score controllers, we wish to measure the size of the basin of attraction. The basin of attraction is a band. So, we can use Basin Width (d), the width of the basin of attraction as measured through the origin, as a proxy for the size of the basin. Figure 2a shows d overlaid on the basin of attraction. Figure 2 shows good agreement between the experimental basin boundary (the edge of the blue region) and the Basin Width estimate (the "Basin Boundary" curves are a plotted parallel to, and a distance d away from, the constant energy curve). Therefore, we can measure the size of the two dimensional basin of attraction with one number.

To measure the Basin Width, we conduct a one dimensional search along a line perpendicular to and intersecting the constant energy curve at the origin: $\phi = \sqrt{\frac{h}{g}}\dot{\phi}$. Let $[\phi_{max}, \dot{\phi}_{max}, 0]^T$ be the state farthest from the origin, along this line, from which the controller can stabilize the bicycle. Using this state, $[\phi_{max}, \dot{\phi}_{max}, 0]^T$, define the Basin Width (d) according to Equation 11.

$$d = \sqrt{(\frac{\phi_{max}}{0.78 \text{ rad}})^2 + (\frac{\dot{\phi}_{max}}{3.19 \text{ rad s}^{-1}})^2} \tag{11}$$

To make the Basin Width dimensionless, we normalize both ϕ and $\dot{\phi}$ in Equation 11. We divide the lean by $\frac{\pi}{4}$ rad (the lean angle at which we declare the bicycle has fallen) and the lean rate by 3.19 rad/s, the lean rate of a bicycle, let fall from just off of the upright equilibrium position, when it reaches a lean angle of $\frac{\pi}{4}$ rad.

Figure 3 shows the relationship of Basin Width (d) vs rear wheel speed (v) for a linear controller.

5 COMPARISON OF LINEAR AND NONLINEAR CONTROLLERS

We compare a linear quadratic regulator to a value iteration controller at various speeds for two sets of steer and steer rate limits. Specifically, we test (1) a set of "reasonable" actuator limits, with $|\dot{\delta}| < 2$ rad/s and $|\delta| < 1.047$ rad $= 60°$ and (2) a set of "restrictive" actuator limits with $|\dot{\delta}| < 1.18$ rad/s and $|\delta| < 0.8$ rad $= 46°$. The steer rate limit reflects that the steer motor cannot

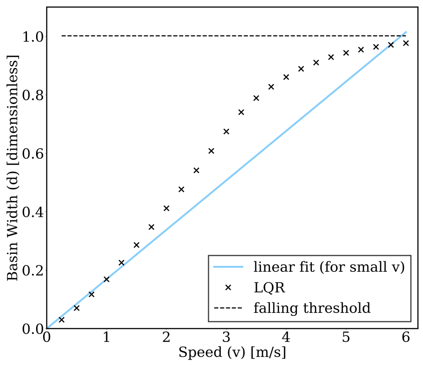


Figure 3. Width of the basin of attraction vs speed. These tests used a linear quadratic regulator (LQR) at each speed. For small speeds, the size of the basin of attraction grows approximately linearly with speed. But, at large speeds, Basin Width grows more slowly as it approaches an asymptote at which the bike is started in a fallen state. The blue line is a linear fit of the Basin Width for the speeds of $v \leq 1.25$ m/s.

spin infinitely fast. The steer angle limit respects the limitations of our bicycle model. Equation 2 has a singularity at $\delta = \pm\frac{\pi}{2}$ rad. Additionally, as $\delta \to \pm\frac{\pi}{2}$ rad, the front wheel becomes more likely to slip, a behavior not captured by our model. So, the steer angle should be kept away from $\pm\frac{\pi}{2}$ rad.

5.1 Comparison with Reasonable Steer Limits

We test a linear quadratic regulator and value iteration controller with reasonable actuator limits ($|\delta| < 1.047$ rad and $|\dot{\delta}| < 2$ rad/s). Over a range of speeds, the nonlinear, value iteration controller has a Basin Width almost equal to that of the linear quadratic regulator. See Figure 4. So, with reasonable actuator limits of $|\delta| < 60°$ and $|\dot{\delta}| < 2$ rad/s, the value iteration controller and linear quadratic regulator perform equally well.

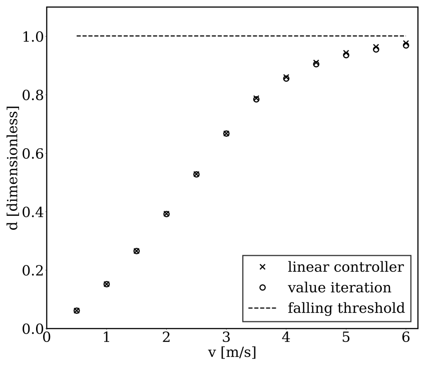
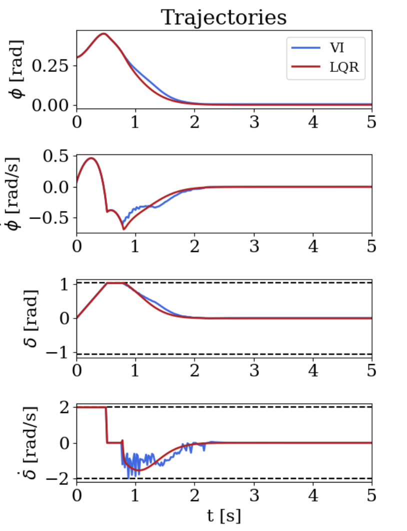
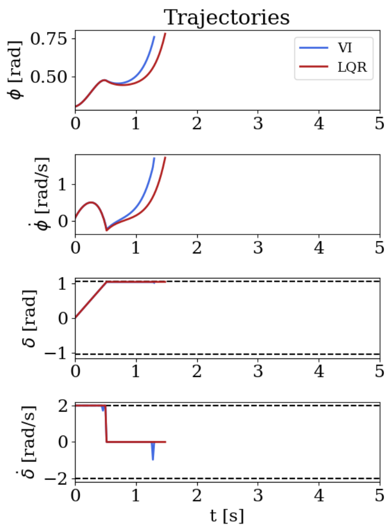


Figure 4. Basin Width (d) vs speed (v) for LQR and value iteration controllers. Both controllers are tested with $|\delta| < 1.047$ rad $= 60°$ and $|\dot{\delta}| < 2$ rad/s. The markers for both controllers, particularly for slow speeds, overlap. The controllers have almost equivalent Basin Widths over a range of speeds.

Figure 5a shows trajectories for the two controllers starting for a state just inside their basins of attraction. Note that both controllers encounter the nonlinearities of saturating the steer motor and reaching the steer angle limit. Figure 5b shows trajectories for both controllers when they fail to stabilize the bicycle, starting for a state just outside of both basins of attraction. In Figure 5b, both controllers saturate the steer rate, reach the steer angle limit, and then fall. So, when saturating the steer motor is the optimal action, and the linear controller saturates the steer motor, the value iteration controller cannot do any better than the linear controller.



(a) Trajectories starting from an initial state of $\boldsymbol{x_0} = [0.3, 0.07, 0]^T$. Both controllers stabilize the bicycle to upright riding. So, this state is within the basin of attraction of both controllers.



(b) Both controllers fail to stabilize the bicycle, starting from an initial state of $\boldsymbol{x_0} = [0.306, 0.075, 0]^T$. This initial state is just on the other side of the boundary of the basin of attraction from the initial state in Figure 5a.

Figure 5. Trajectories of a value iteration controller (VI) and Linear Quadratic Regulator (LQR) at $v = 2m/s$ with $|\delta| < 1.047$ rad and $|\dot{\delta}| < 2$ rad/s. For both initial conditions, both controllers saturate $\dot{\delta}$ and reach the limit of δ. But, only in Figure 5a do the controllers stabilize the bicycle.

5.2 Comparison with Restrictive Steer Limits

Separately, we test a bicycle with "restrictive" actuator limits of $|\delta| < 0.8$ rad and $|\dot{\delta}| < 1.18$ rad/s. Figure 6 plots the Basin Width for LQR and value iteration controllers, subject to these limits, over a range of speeds. As expected, the controllers with this restrictive limit (Figure 6) have smaller Basin Widths than the controllers with higher limits (Figure 4). During the training (when estimating the utility function), the value iteration controller accounts for the steer rate and steer angle limits. That is, when estimating the utility function, the limits on δ and $\dot{\delta}$ are enforced. When deriving the linear quadratic regulator, the steer angle and steer rate limits are not considered. Compared to larger steer and steer rate limits (as in Section 5.1), smaller actuator limits (as in Section 5.2) affect more trajectories. So, the value iteration controller, which directly accounts for those limits, can perform better than the linear quadratic regulator, which does not.

Figure 7a plots the trajectories of both controllers starting from a state just inside both basins of attraction. Figure 7b plots both controllers, but from an initial state just outside of the basin of attraction of the linear quadratic regulator. From this state, only the value iteration controller

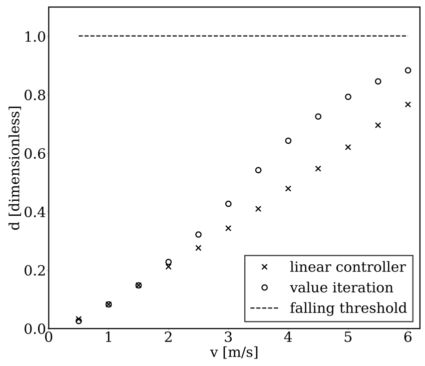
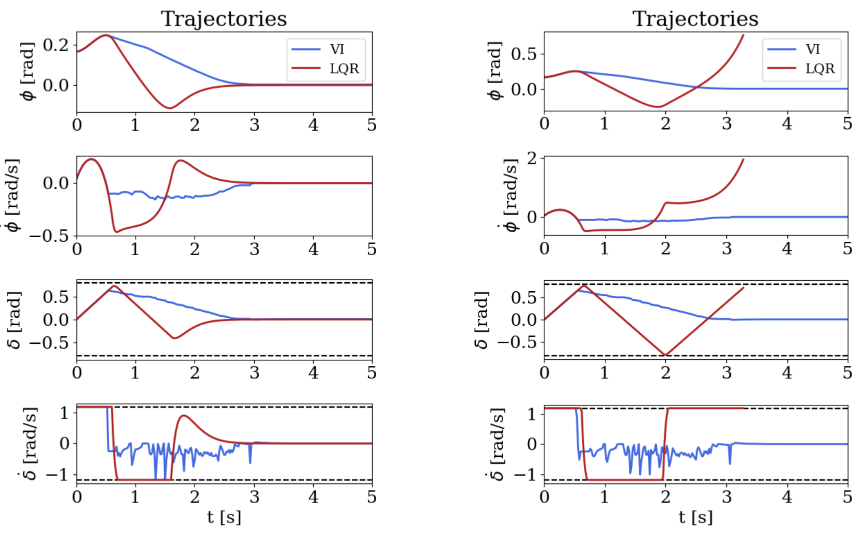


Figure 6. Basin Widths (d) vs speed (v) for LQR and value iteration controllers with restrictive steer-motor limits of $|\dot{\delta}| < 1.18$ rad/s and $|\delta| < 0.8$ rad. For large speeds, the value iteration controller exhibits a larger Basin Width than the LQR controller.



(a) Both the value iteration controller (VI) and linear quadratic regulator (LQR) stabilize the bicycle, starting from an initial state of $\boldsymbol{x}_0 = [0.164, 0.037, 0]^T$.

(b) The value iteration controller (VI) stabilizes the bicycle while the linear quadratic regulator (LQR) does not. Both controllers start from $\boldsymbol{x}_0 = [0.165, 0.04, 0]^T$.

Figure 7. Trajectories of a value iteration controller (VI) and Linear Quadratic Regulator (LQR) at $2m/s$ with $|\delta| < 0.8$ and $|\dot{\delta}| < 1.18$. The different behaviors in Figure 7b reflect the larger Basin Width of the value iteration controller (see Figure 6).

stabilizes the bicycle. In Figure 7b, the value iteration controller better handles the saturation of $\dot{\delta}$ and the limit on δ. The steer rate ($\dot{\delta}$) command of the value iteration controller is less smooth than for the linear controller, but these discontinuous commands help the bicycle to balance.

6 CONCLUSIONS

Basin Width provides a useful quantity to characterize the set of recoverable states of a bicycle balance controller. This scalar allows us to easily compare different controller architectures at different speeds.

When the bicycle is restricted to small steer angles and small steer rates, an optimal, nonlinear, value-iteration controller has a larger basin of attraction than a linear quadratic regulator. But, for bicycles with larger, more realistic steer angle and steer rate limits, an optimal, nonlinear, value-iteration controller and a linear quadratic regulator perform equally well. This parity suggests that, using linear control, we can develop bicycle controllers with close-to-maximal basins of attraction.

REFERENCES

[1] H. Getz, "Control of balance for a nonlinear nonholonomic non-minimum phase model of a bicycle." In *Proceedings of 1994 American Control Conference-ACC'94*, vol. 1, pp. 148-151. IEEE, 1994.

[2] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, MIT press, 2018.

[3] J. Randløv and P. Alstrøm, “Learning to Drive a Bicycle Using Reinforcement Learning and Shaping”, In *ICML*, **98**, pp. 463-471. 1998.

[4] J. Tan, Y. Gu, C.K. Liu, and G. Turk, 2014. “Learning bicycle stunts”. *ACM Transactions on Graphics (TOG)*, **33**(4), p.50.

[5] J. He, M. Zhao and S. Stasinopoulos, “Constant-velocity steering control design for unmanned bicycles”, In *2015 IEEE international conference on robotics and biomimetics (ROBIO)*, pp. 428-433. IEEE, 2015.

[6] J. Yi, D. Song, A. Levandowski and S. Jayasuriya. “Trajectory tracking and balance stabilization control of autonomous motorcycles", In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006*, pp. 2583-2589. IEEE, 2006.

[7] Y. Tanaka and T. Murakami, “Self sustaining bicycle robot with steering controller”, In *The 8th IEEE International Workshop on Advanced Motion Control, 2004. AMC'04.*, pp. 193-197. IEEE, 2004.

[8] A. L. Schwab and N. Appelman. “Dynamics and control of a steer-by-wire bicycle”, In *Proceedings of Bicycle and Motorcycle Dynamics 2013 Symposium on the Dynamics and Control of Single Track Vehicles*, 2013.

[9] A. Sharma, S. Wang, Y. M. Zhou and A. Ruina, “Towards a maximally-robust self-balancing robotic bicycle without reaction-moment gyroscopes nor reaction wheels”, *Proceedings, Bicycle and Motorcycle Dynamics*, 2016.

[10] D. Meehan, R. Bandaru, O. Xiang, K. Tian, *et. al.*, “Cornell Autonomous Bicycle Project Team Spring 2018 Report”, unpublished, 2018.

[11] J. P. Meijaard, J. M. Papadopoulos, A. Ruina, and A. L. Schwab, “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* **463**(2084), 2007, pp. 1955-1982.

[12] S. Wang, “Dynamic model derivation and controller design for an autonomous bicycle”, unpublished, 2014.

[13] K. J. Aström and R. M. Murray, *Feedback systems: an introduction for scientists and engineers*, Princeton university press, 2010.

[14] S. J. Russell, and P. Norvig, *Artificial intelligence: a modern approach*, Malaysia; Pearson Education Limited, 2016.

[15] P. Bourke, "Interpolation methods", *Miscellaneous: projection, modelling, rendering*, 1 (1999).

[16] S. Tarbouriech, G. Garcia, J. M. G. da Silva Jr and I. Queinnec, *Stability and stabilization of linear systems with saturating actuators*, Springer Science & Business Media, 2011.

A DERIVATION OF CONSTANT ENERGY CURVE

To calculate the curve of states with equal energies, we must calculate the energy of the point mass bicycle model.

The energy, E, of a system is given by

$$E = E_K + E_P \tag{12}$$

where E_K is the system's kinetic energy and E_P is the system's potential energy.

The bicycle's potential energy is given by

$$E_P = mgh\cos(\phi) \tag{13}$$

Since we use a point mass bicycle model, the kenetic energy of the system is

$$E_K = \frac{1}{2}m\|\boldsymbol{v}_G\|^2 \tag{14}$$

where $\boldsymbol{v}_G$ is the velocity of the point mass.

$\boldsymbol{v}_G$ is given by the three term velocity formula

$$\boldsymbol{v}_G = \boldsymbol{v}_C + \boldsymbol{v}_{G/C} + \boldsymbol{\omega}_\beta \times \boldsymbol{r}_{G/C} \tag{15}$$

where point G is the center of mass and point C is the contact point of the rear wheel. Frame β has its origin at C. β translates and yaws, but does not lean, with the bicycle. Since we model the wheels as skates, $\boldsymbol{v}_C$ is the speed of the rear wheel, in its direction of travel. Let $\hat{\lambda}$ be a unit vector in the direction of travel of the rear wheel. Then, $\boldsymbol{v}_C = v\hat{\boldsymbol{\lambda}}$ where v is the rear wheel speed as before. $\boldsymbol{v}_{G/C}$ is the velocity of G relative to C. $\boldsymbol{\omega}_\beta$ is the angular velocity of frame β. $\boldsymbol{r}_{G/C}$ is the position vector from C to G.

$$\boldsymbol{\omega}_\beta = \dot{\psi}\hat{\boldsymbol{k}} \tag{16}$$

where $\hat{\boldsymbol{k}}$ is a unit vector perpendicular to the ground plane and in the opposite direction as gravity. $\dot{\psi}$ is the yaw rate of the bicycle.

From [12], $\dot{\psi} = \frac{v\tan(\delta)}{l\cos(\phi)}$. We only consider the basin of attraction of states with $\delta = 0$. So, set $\delta = 0$ to get $\boldsymbol{\omega}_\beta = 0$.

$$\boldsymbol{v}_{G/C} = \dot{\phi} h \hat{\boldsymbol{e}}_{\boldsymbol{\phi}} \tag{17}$$

where $\hat{\boldsymbol{e}}_{\boldsymbol{\phi}}$ is a unit vector perpendicular to $\boldsymbol{r}_{G/C}$ and $\hat{\boldsymbol{\lambda}}$ and pointing to the left of the bicycle.

Therefore,

$$\boldsymbol{v}_G = v\hat{\boldsymbol{\lambda}} + \dot{\phi} h \hat{\boldsymbol{e}}_{\boldsymbol{\phi}} \tag{18}$$

So,

$$E_k = \frac{1}{2} m(v^2 + \dot{\phi}^2 h^2) \tag{19}$$

So, the total energy of the bicycle, E is

$$E = \frac{1}{2} m(v^2 + \dot{\phi}^2 h^2) + mgh\cos(\phi) \tag{20}$$

To calculate the set of states with energy equal to an upright bicycle, set the total energy of a bicycle in a leaned state ($\phi \neq 0$) equal to the total energy of an upright bicycle ($\phi = 0$). Cancel the terms involving forward speed to get Equation 21.

$$mgh = mgh\cos(\phi) + \frac{mh^2\dot{\phi}^2}{2} \tag{21}$$

Solving 21 gives ϕ as a function of $\dot{\phi}$ as

$$\phi = \arccos(1 - \frac{h}{2g}\dot{\phi}^2) \tag{22}$$

When taking inverse cosine, take care to ensure ϕ has the correct sign, since $\cos(\phi) = \cos(-\phi)$. In particular, ϕ and $\dot{\phi}$ should have opposite signs.

B EIGENVECTOR OF LINEARIZED SYSTEM

We could like to compare this constant energy curve (Appendix A) to an eigenvector of the linearized system. Taking the state vector $\boldsymbol{x} = [\phi, \dot{\phi}, \delta]^T$, the state matrix ($\boldsymbol{A}$), is given in [12].

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{g}{h} & 0 & \frac{-v^2}{h\ell} \\ 0 & 0 & 0 \end{bmatrix} \tag{23}$$

$[-\sqrt{gh}, g, 0]$ is an eigenvector of A corresponding to a negative eigenvalue $-\sqrt{\frac{g}{h}}$.

States on this eigenvector obey

$$\phi = -\sqrt{\frac{g}{h}}\dot{\phi} \tag{24}$$

.

Now, we take a Taylor Series approximation of $\cos$ in Equation 21. Simplifying and ignoring higher order terms gives equation 25.

$$g\phi^2 = h\dot{\phi}^2 \tag{25}$$

.

Solving for ϕ and taking the correct branch of the square root function gives Equation 26.

$$\phi = -\sqrt{\frac{g}{h}}\dot{\phi} \tag{26}$$

.

Equation 24 is the same as Equation 26. So, the eigenvector described by Equation 24 provides a linear approximation for Equation 21.