MEAM 5170 Final Report: Trajectory Planning for a Bicycle

Joshua Ren, Zac Gong, Yifan Wang

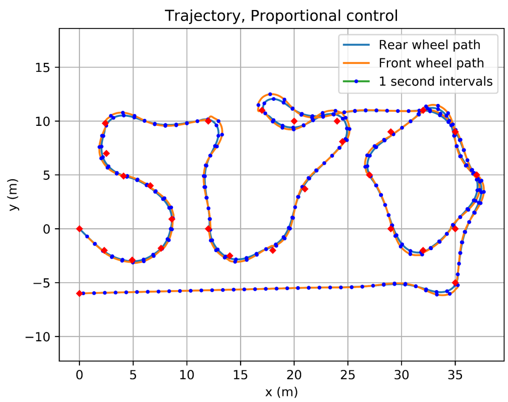

Fig 1: Bike test trajectory, waypoint following

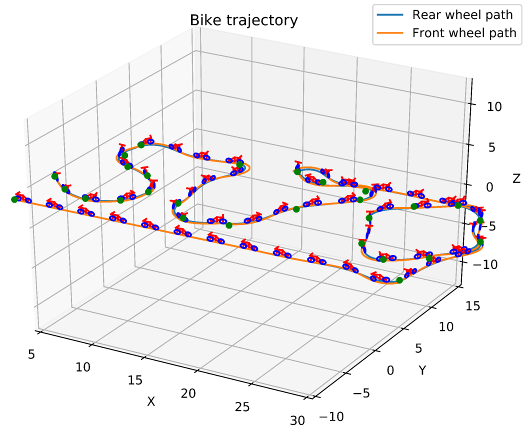

Fig 2: Bike test trajectory, waypoint following

***Abstract*— In this project, we developed and compared control strategies for a bicycle, targeting the dual goals of balancing and directing it simultaneously. During the process of developing the controllers, we investigated some of the interesting dynamics and control properties of the vehicle. Overall, LQR performed slightly better than proportional control, but the behaviors were similar.**

1. Introduction: Open Loop Bicycle Model

Our first goal was implementing an open loop dynamic bicycle model that would allow us to simulate steering into the fall. Any control strategy must allow the controller to both stabilize and direct the bike to a target, and this controller must steer into the fall to remain stable-i.e. when leaning to the right, you must continue steering to the right to prevent falling further to the right. Furthermore, the bicycle is a non-minimum phase system. If you want to turn to the right while going straight, you must first steer to the left to get the bike leaning to the right (steering the bike out from under you) and any controller must take this into account. We implemented a bicycle model that takes these factors into account using nonlinear equations of motion that capture the roll dynamics of the bicycle, the roll torque produced by steering and the heading change (model is from Dr. Jason K. Moore [1]).

Model assumptions:

1. The bicycle and rider are a single rigid body.
2. Control input allows direct control of the steering rate of the front steering assembly.
   a) Ex: The inertia of the steering assembly does not slow down steering control, and input torque from the road does not steer the wheels in a self-righting manner.
3. There are no gyroscopic effects from the spinning wheels.
4. We made an additional assumption that the bike travels at a constant forward speed.

Model variables:

m: Combined mass of the bicycle and the rider

h: Height of the center of mass

A: Distance from rear wheel to the projection of the center of mass

B: Wheelbase

v_r **and** v_f**:** Speed at rear and front wheels respectively

I_x, I_y, **and** I_z**:** Principal moments of inertia

δ**:** The steering angle, and θ**:** The roll angle

ψ**:** The heading angular velocity

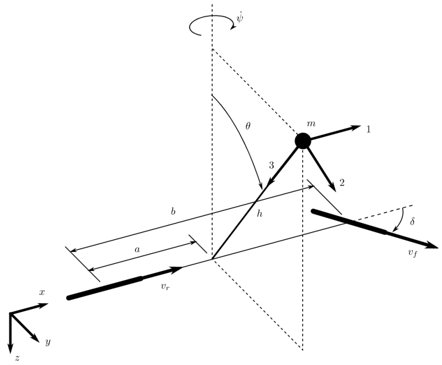

Figure 3: Bicycle dynamic model diagram [1]

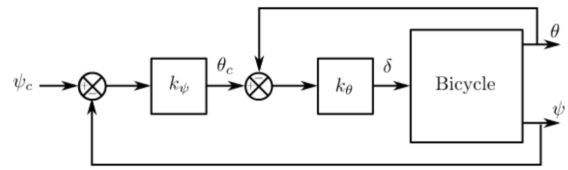

Figure 4: Proportional feedback diagram [1]

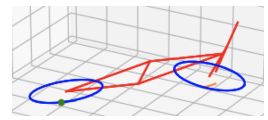

Figure 5: Our bike

The nonlinear equation of motion we used is the first of the following equations, which describes the natural roll dynamics on the left hand side and the roll torque from steering on the right hand side. This goes along with the differential equation for heading, which is the second equation shown [1].

$$(I_x + mh^2)\,\ddot{\theta} + \left(I_z - I_y - mh^2\right)\left(\frac{v_r tan\delta}{b}\right)^2 sin\theta\, cos\theta - mgh\, sin\theta = - mh\, cos\theta\left(\frac{av_r}{bcos^2\delta}\dot{\delta} + \frac{v_r^{\,2}}{b}tan\delta\right) \quad \text{(Eq. 1)}$$

$$\dot{\psi} = \frac{v_r}{b}tan\delta$$

2. Proportional controller implementation

We implemented a proportional controller as described in [1], which is a sequential proportional feedback controller with an inner loop to maintain bike stability and an outer loop to direct the bike to a desired heading (figure 4). The desired heading is either a heading hold or a heading to a waypoint target, which is then used to generate a heading error. The desired roll angle then uses feedback on heading error: if we need to change our heading to the right, we need to roll to the right. The steer angle then uses positive feedback on the roll error by steering into the turn, and the control input on steering angle rate uses negative feedback on steer angle error which initiates the initial opposite direction turn. Gains (3x gains) were chosen either manually or by a parameter sweep that maximized distance traveled by the bike in a given time, a heuristic for straightening out the underdamped trajectory. This script also selected gains that kept the control (steer rate) within a rate limit. Possible extensions could include adding derivative feedback, dynamic gain adjustment based on maneuvering or control filtering.

3. Time-varying LQR controller implementation

Time-varying LQR is designed to derive a controller that will allow the bike to follow the specific path in a stable manner. First, we linearized the bicycle dynamics for time-varying LQR. The dynamics of the bike can be packaged into the standard form as described in (2), where the status x contains roll angle θ, roll angle rate $\dot{\theta}$, heading angular ψ, and steering angle δ. The input is expressed in (2), which is the steering angle rate $\dot{\delta}$. The dynamic system for the bike is expressed as in equation (3).

$$x = \left[\theta,\ \dot{\theta},\ \psi,\ \delta\right], u = \left[\dot{\delta}\right] \quad (2) \qquad\qquad x_{k+1} = f(x_k, u_k) \quad (3)$$

$$x_{k+1}^{\ *} = f\left(x_k^{\ *}, u_k^{\ *}\right),\ \forall t \in \left[0,\ t_f\right] \quad (4) \qquad\qquad x_e = x_k - x_k^{\ *}, u_e = u_k - u_k^{\ *} \quad (5)$$

Then, suppose the bike is tasked with following some desired path $x_k^{\ *}$, and that the control input $u_k^{\ *}$ will drive the bike along this path during the t=[0, t_f] as shown in (4). The error x_e and the control error u_e is in (5).

Time-varying LQR could be used to create a stable equilibrium at $x_e = 0$ by taking the time derivative of the error x_e and linearize f at (x_e, u_e) to approximate the dynamics. And we derive the equation as (6). With the Taylor expansion for error dynamics, we could derive the A_k and B_k as in (7):

$$x_{k+1} - x_{k+1}^{*} = f\left(x_k, u_k\right) - x_{k+1}^{*} \approx \frac{df}{dx}\left(x_k^{*}, u_k^{*}\right)\left(x_k - x_k^{*}\right) + \frac{df}{du}\left(x_k^{*}, u_k^{*}\right)\left(u_k - u_k^{*}\right) = A_k x_e + B_k u_e \tag{6}$$

$$A_k = \frac{df}{dx}\left(x_k^{*}, u_k^{*}\right) = [[0,\ 1,\ 0,\ 0],\ [facA,\ 0,\ 0,\ facB],\ [0,\ 0,\ 0,\ facC],\ [0,\ 0,\ 0,\ 0]], \tag{7}$$

$$B_k = \frac{df}{du}\left(x_k^{*}, u_k^{*}\right) = [0,\ facD,\ 0,\ 1]$$

where $facA = \dfrac{-\frac{\left(I_z - I_y - mh^2\right)v_r^2 \tan^2\delta \cos 2\theta}{b^2} + mgh\cos\theta + mh\sin\theta\left(\frac{av_r\dot{\delta}}{b\cos^2\delta} + \frac{v_r^2\tan\delta}{b}\right)}{I_x + mh^2}$, $facB = \dfrac{-\frac{2\left(I_z - I_y - mh^2\right)v_r^2 \sin\delta\, \sin\theta\, \cos\theta}{b^2\cos^3\delta} - \frac{2mhav_r\cos\theta\dot{\delta}\sin\delta}{b\cos^3\delta} - \frac{mhv_r^2\cos\theta}{b\cos^2\delta}}{I_x + mh^2}$,

$facC = \dfrac{v_r}{b\cos^2\delta}$, $facD = -\dfrac{mhav_r\cos\theta}{(I_x \_ mh^2)b\cos^2\delta}$.

With the linearized dynamics above, the error x_e can be stabilized by solving the optimal control problem associated with the LQR cost in (8), which can be solved numerically to obtain a stabilizing controller.

$$J = \frac{1}{2}\int_0^{t_f} x_e(t)^T Q x_e(t) + u_e(t)^T R u_e(t) dt + \frac{1}{2} x_e(t_f)^T Q_f x_e\left(t_f\right) \tag{8}$$

4. Results: Controller evaluation and comparison

We compared our controllers in heading hold mode by having each of them drive the bike straight for several seconds, then change heading 90 degrees to the left, and repeat to draw a square. The LQR controller performed better than proportional control, either by using less control effort (steering rate) for similar settling time or similar control effort for better settling time.

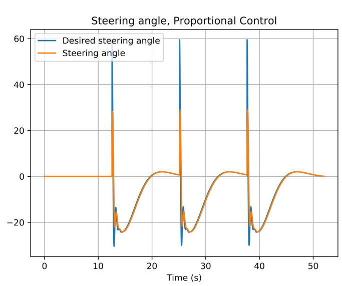

Fig. 6: Proportional control, manual tuning. Note the requested impulse on steering angle (bad)

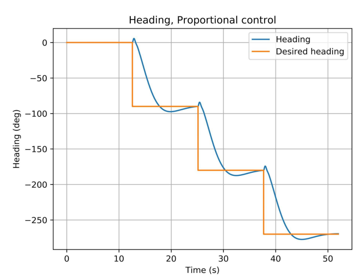

Fig. 7: Proportional control, manual tuning. Note the long settling time.

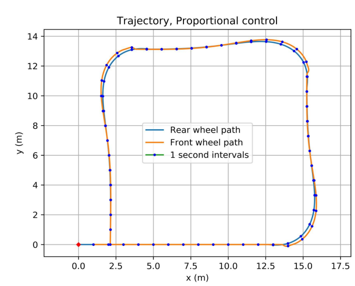

Fig. 8: Trajectory, proportional control, manual tuning. Note the front wheel path exhibiting countersteer (desired)

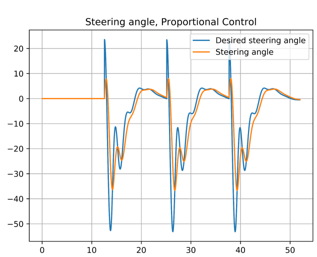

Fig 9: Proportional control, parameter sweep tuning while filtering for rate limits. Note that the maximum steer angle is decreased, while damping is compromised.

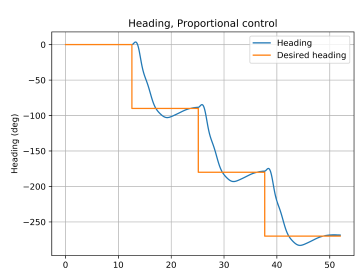

Fig 10: Proportional control, parameter sweep tuning. Rise time and settling time slightly improved.

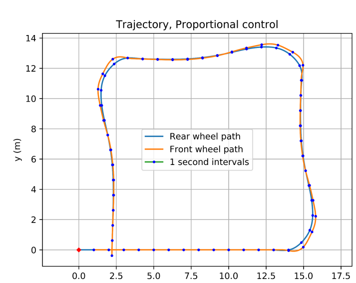

Fig 11: Proportional control, parameter sweep tuning.

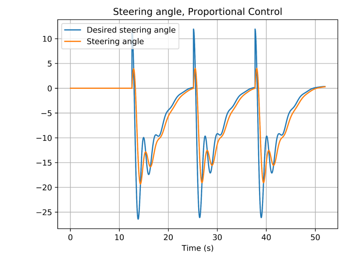

Fig 12: Proportional control, parameter sweep for min. heading overshoot.

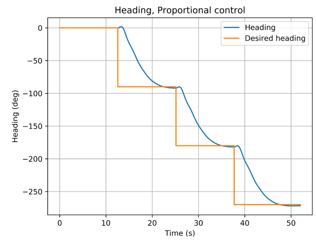

Fig 13: Proportional control, parameter sweep for min. heading overshoot.

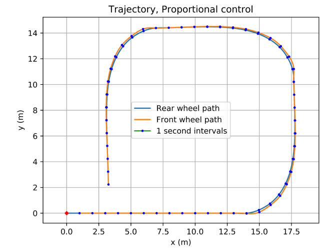

Fig 14: Proportional control, parameter sweep for min. heading overshoot.

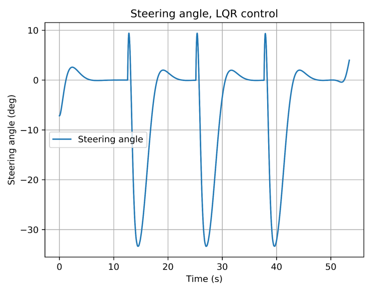

Fig. 15: Time-varying LQR control. Note the wider band on steer angle input at the initial countersteer request (Less sharp demand, spread out in time) (good).

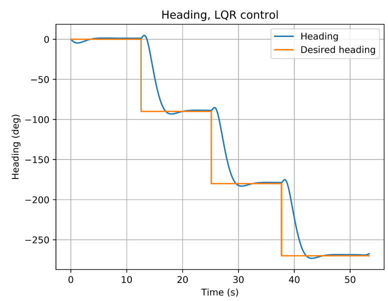

Fig. 16: Time-varying LQR control. Note the good settling time.

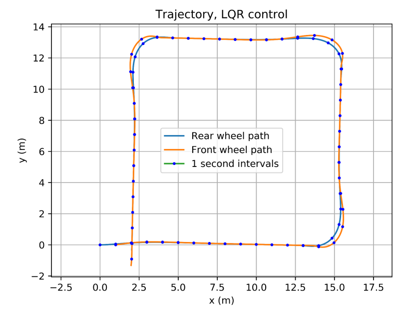

Fig. 17: Time-varying LQR control.

We also tried some other trajectories for LQR control, like a circle. From this trajectory, we found that the designed control could keep the bicycle stable at a constant turn rate. This means that when the bicycle is following a circle trajectory, it would have a relatively constant steering angle, which meets the common sense.

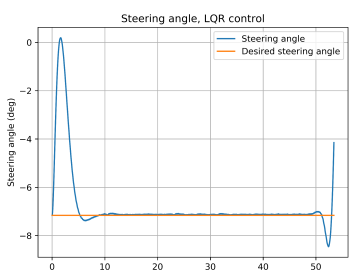

Fig. 18: Time-varying LQR control, circular trajectory.

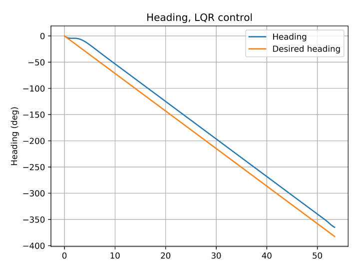

Fig. 19: Time-varying LQR control, circular trajectory.

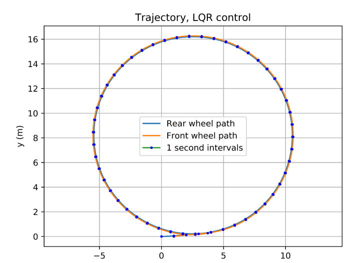

Fig. 20: Time-varying LQR control, circular trajectory.

4. Conclusion

The bicycle is a highly complex nonlinear system, and surprisingly proportional control can do a decent job of controlling it. Derivative feedback and filtering could perhaps help the proportional controller, but LQR seems like it would be slightly better still. We suspect, however, that MPC would be an even better choice given that bicycle input limits (like steer angle and steer angle rate) are probably more relevant in determining the limits of the physical system. With more time we would next implement MPC and evaluate its performance.

References

1. Dr. Jason K. Moore: https://plotly.com/python/v3/ipython-notebooks/bicycle-control-design/
2. Matthew Cooke: https://www.gwern.net/docs/reinforcement-learning/model-free/2004-cook.pdf
3. Andy Ruina: http://www.andyruina.org/research/topics/bicycle_mechanics/BicyclePaper1Andyv31.pdf