SISO Control of a Bicycle-Rider System

Chad Findlay, Jason Moore, Claudia Perez-Maldonado

Introduction

The inherent dynamic instability of two-wheeled vehicles naturally leads to questions of control. The bicycle-rider system itself is a dynamically complex, multiple-degree-of freedom system. Thus, how is it possible to maintain the bicycle upright? The seemingly simple task of riding a bicycle is made possible by the complex control system of the rider. Although the rider controls various inputs, this paper briefly examines how only the front wheel steering input may be used to control the bicycle roll angle. If a true controller were to be realized, its design could take various forms.

The previously derived single degree-of-freedom model of a bicycle-rider system will be re-stated. The linear model presented by Karnopp was used as a reference in the derivation of this model [1]. This model will then be used to develop a single input, single output controller. The controller is developed first by classical, frequency domain techniques. This strategy is then compared to full-state feedback control wherein a regulator designed by pole-placement is compared to an LQR controller.

System Model

The previously developed model of the bicycle-rider system treated the bicycle and rider as a single rigid body. The rear wheel velocity at the contact point was assumed to be constant, and the wheels were assumed to be in pure rolling contact with the ground with no sideslip. The inherent stability characteristics associated with an offset, angled front fork and the rotating wheels were neglected. This led to the nonlinear, second order differential equation given by

$$\left(I_1 + mh^2\right)\ddot{\theta} + \left(I_3 - I_2 - mh^2\right)\left(\frac{v_r \tan\delta}{b}\right)^2 \sin\theta\cos\theta$$

$$- mgh\sin\theta = -mh\cos\theta\left(\frac{av_r}{b\cos^2\delta}\dot{\delta} + \frac{v_r^2}{b}\tan\delta\right). \quad \textbf{(1)}$$

The system parameters as shown in Figure 1 and Figure 2 are as follows:

I_1 Principal moment of inertia about the roll axis $[kg\,m^2]$

m Bike & Rider mass $[kg]$

h Height of the center of mass when the bicycle is upright $[m]$.

a Distance from the projection of the center of mass on the ground plane to the contact point of rear wheel $[m]$.

b Distance along the ground between the wheel contact points [m].

v_r Forward velocity of rear frame $[m/s]$

g Local acceleration due to gravity $[m/s^2]$

δ Front wheel steering angle. $[Rad]$

θ Bicycle roll angle $[Rad]$

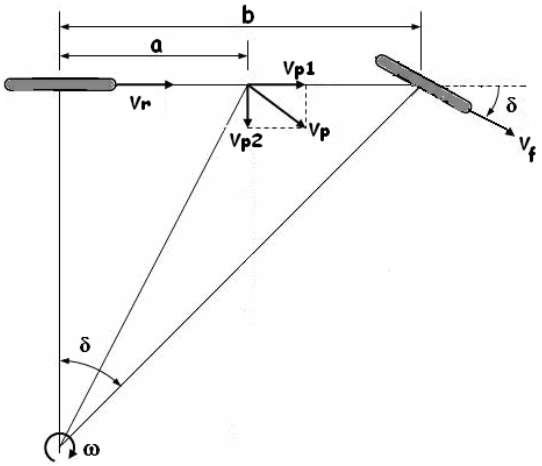


Figure 1. Ground plane geometry of bicycle-rider system in a turn.

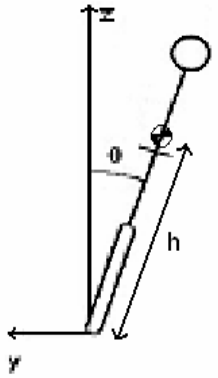


Figure 2. Rear view of bicycle-rider system. Taken from Åström et al. [2005]

In linearizing Equation (1), both of the angles θ and δ were assumed to be small (< 35°). The equilibrium point for both of the parameters is zero, and this can only be assumed if the bicycle is traveling in a fairly straight path with a relatively high rear wheel velocity. Because the angles are small, the squared term on the left hand side of Eq. 1 can be eliminated. Substituting the small angle approximations $\sin\theta \cong \theta$, $\cos\theta \cong 1$, and $\tan\delta \cong \delta$ are into the equation yields the linear form

$$\left(I_1 + mh^2\right)\ddot{\theta} - mgh\theta = -\frac{mh}{b}\left(av_r\dot{\delta} + v_r^2\delta\right). \quad \textbf{(2)}$$

Validity of Linear Model

The validity of the linear model is tested by comparing the solutions of Equation 1 and Equation 2 from an initial roll angle, θ_o of 5 degrees and an initial roll rate of 0 degrees per second. The steering angle δ and its derivative are set to zero. The constant forward velocity v_r is 5m/s. These solutions are plotted in Figure 3 below. Both solutions reveal that the bike falls over, as expected. Note that the solutions do not significantly diverge until a roll angle of 90 degrees. That is, the linear model is a reasonable approximation until the bike has fallen completely over. For our assumptions, the bike roll angle does not exceed the maximum normal riding condition of 35 degrees. In this case, the solution to the linear equation is nearly identical that of the nonlinear equation.

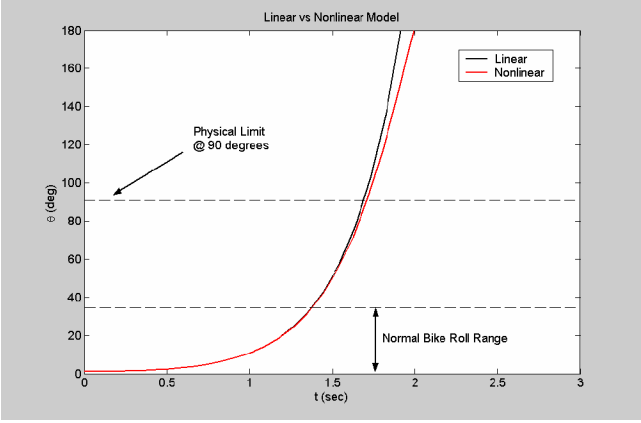


Figure 3. Linear & nonlinear solutions to initial roll angle

State Equations

Due to the derivative of the steer angle input in Equation 2, derivation of the state equations is not as simple as substituting variables x_1 and x_2 for θ and its derivative and deriving the first order equations. Thus, the state equations were previously developed by hand and resulted in the *controller companion form* given by

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}\delta(t)\\ \theta(t) &= \mathbf{C}\mathbf{x} + \mathbf{D}\delta(t)\end{aligned} \quad \textbf{(3)}$$

with matrices

$$\mathbf{A} = \begin{bmatrix} 0 & \dfrac{mgh}{I_1 + mh^2} \\ 1 & 0 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} \dfrac{-mhav_r}{b(I_1 + mh^2)} & \dfrac{-mhv_r{}^2}{b(I_1 + mh^2)} \end{bmatrix}$$

$$\mathbf{D} = [0]$$

Unfortunately, the physical significance of the states x_1 and x_2 is difficult to interpret in this state space. This is most apparent by noting the **C** matrix; the output θ is a linear combination of the states. We desire to specify an initial roll angle θ_o to test the abilities of various controllers to bring the bike to the upright position. In the controller companion form, specifying the initial state $x_1(0)$ is, therefore, complicated by the fact that

$$x_1(0) \neq \theta(0).$$

Converting to the observer companion form alleviates this inconvenience. Transposing the above **A** matrix, letting **B** equal $\mathbf{C}^T$, and letting **C** equal $\mathbf{B}^T$, the state space in observer companion form is given by

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \dfrac{mgh}{I_1 + mh^2} & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} \dfrac{-mhav_r}{b\left(I_1 + mh^2\right)} \\ \dfrac{-mhv_r^2}{b\left(I_1 + mh^2\right)} \end{bmatrix}\delta$$

with the output equation

$$\theta = \begin{bmatrix} 1 & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + [0]\delta. \quad \textbf{(4)}$$

From the output equation, it is now obvious that state x_1 is the bike roll angle, θ. By solving the first state equation for x_1, we can write the second state, x_2, in terms of $\dot{x}_1$ (which is $\dot{\theta}$). Thus, both states in observer companion form are given by

$$x_1 = \theta$$
$$x_2 = \dot{\theta} + \frac{mhav_r}{b\left(I_1 + mh^2\right)}\delta. \quad \textbf{(5)}$$

So, specifying an initial value for state x_1 is, in fact, specifying the initial roll angle from which to run simulations. The equation for state x_2 also reveals some physical significance. For a constant forward velocity, x_2 represents a linear combination of roll rate and a term comparable to a steer rate. This "steer rate" is evident in the units of the second term. Recall that we are considering nonzero, constant values for the forward velocity term, v_r. Thus, by specifying a zero initial condition for state x_2, not only is the initial roll rate zero, but also the initial steer angle.

Roll Angle Performance Criteria

In order to compare control strategies, we specify reasonable performance criteria to meet the goal of bringing the bike to an upright position from an initial roll angle of 5 degrees while traveling at a constant velocity of 5m/s. The desired roll angle response from these initial conditions was decided to be

- Overshoot (OS) < 1deg
- 2% Settling time (t_s) < 2sec

To limit the control input required to meet these criteria, a limit on the steer angle was decided to be

- δ < 20 deg.

Although by minimizing the input, less "energy" is required to bring the system into equilibrium. So it is desired to minimize δ, possibly keeping it under 5 degrees.

Classical Control Design

The linear model derived in Equation (2) was mapped into the frequency domain by the use of the Laplace transform. In the frequency domain, the transfer function of the system derived by expressing the ratio of the roll angle to the steering angle of the bicycle-rider system is given by

$$\frac{\theta(s)}{\delta(s)} = \frac{-\frac{mhv_r}{b}(as + v_r)}{(I_1 + mh^2)s^2 - mgh}. \quad \textbf{(6)}$$

In order to stabilize the system, a proportional controller was used. In order to find the values of K that keep the system stable the Routh-Hurwitz criteria was utilized.

Table 1. Routh-Hurwitz criteria

S^2	1	9.42-K348
S^1	-87K	0
S^0	9.42-K348	0

From Table 1 it can be seen that in order for -87K to be positive, and keep the system stable, K has to be less than zero.

The step response of the system is presented in Figure 4. It can be seen that the system settles in 0.246 seconds. However, the %OS is of 10.5% which is completely off the design criteria specified.

The bicycle-rider system was modeled as an ideal 2nd order system. Although ideal 2nd order systems have only two poles and no zeros, the bicycle-rider system had one zero at -10 and two poles at ± 3.07. Even when the zero affected the transient response of the system, the 2nd order approximation was used in order to simplify the calculation of the location of the poles and zeros for the compensation system. The transient response equations, (7) and (8), that are valid for ideal 2nd order systems were used to aid in the calculations.

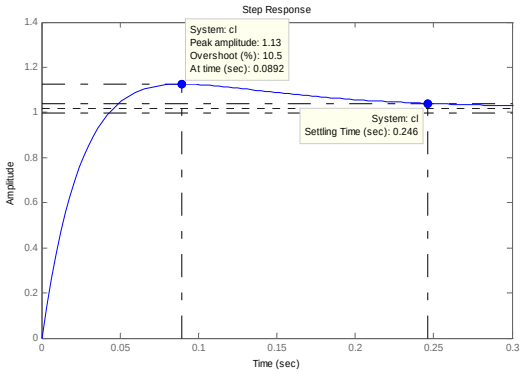


Figure 4. Step response of proportionally controlled system.

In order to meet the performance criteria, Equations (7) and (8) were used to obtain the damping ratio and the natural frequency of the uncompensated system.

$$T_s = \frac{4}{\zeta\omega_n} \quad \textbf{(7)}$$

$$\varsigma = \frac{-\ln(\%OS/100)}{\sqrt{\left(\pi^2 + \ln(\%OS/100)^2\right)}} \qquad \textbf{(8)}$$

The values of ω_n and ς yield the desired operating point in the complex plane as shown in Figure 5. In order for the system to have the desired response, the root locus has to pass through this operating point. This point was found to be s=-26.4+18j.

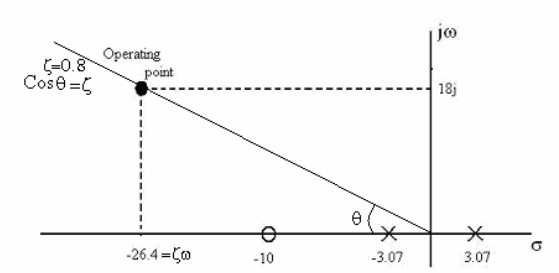


Figure 5. Desired operating point and location of poles & zeros of the plant

Using MATLAB to analyze the root locus (Fig 6), it was found that the root locus does not pass through the operating point. Thus, additional compensation was necessary to shift the root locus accordingly.

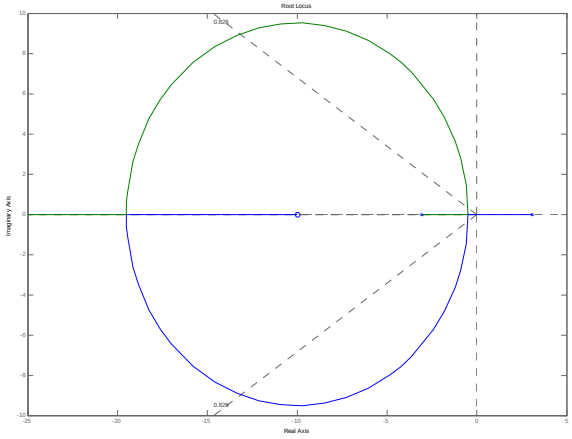


Figure 6. Root locus of proportionally controlled system

To meet the performance criteria and decrease the steady-state error, a lead-lag controller was designed. Use of a lead-lag controller precludes the use of the more expensive and energy-intensive circuitry associated with PID controllers. PID controllers are considered the ideal controllers for improving transient response and eliminating steady state error [Nise, 2004]. However, the real controllers that are used to correct those deficiencies are the lead-lag controllers.

The transfer function of a lead controller has a pole and a zero to be placed strategically so as to modify the root locus of the system. The zero was selected so as to cancel a pole of the plant. Based on its value and the locations of the plant's poles and zeros, the pole of the lead controller was determined.

After selecting the location of the zero, the location of the pole was calculated. In order for a pole to be part of the root locus, the phase of the characteristic equation of the closed loop system needs to be an odd multiple of 180º, as shown in equations (9) and (10).

$$CLTF = \frac{G_c(s)G(s)}{1 + kG_c(s)G(s)} \qquad \textbf{(9)}$$

Closed loop poles

$$1 + kG_c(s)G(s) = 0 \Rightarrow kG_c(s)G(s) = -1$$

$$\Rightarrow kG_c(s)G(s) = 1 \prec (2q+1)180^0 \qquad \textbf{(10)}$$

Where q is an integer.

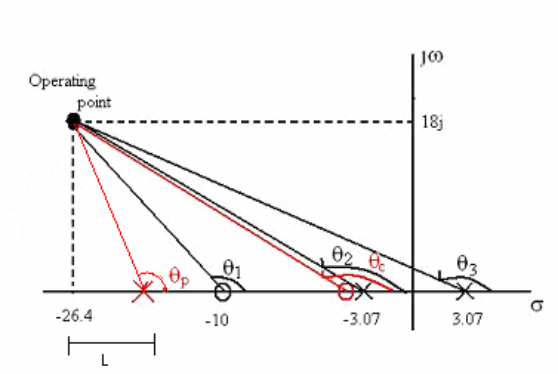


Figure 7. Lead controller pole-zero placement

The angle of the new pole can be obtained from the next equation, which is based on Equation (10) and Figure (7).

$$\theta_p = \theta_c + \theta_1 - \theta_2 - \theta_3$$

$$\Rightarrow \theta_p = 132.34 + 172 - 172 - 172.4 + 180 = 139.94^o$$

With the angle θ_p the location of the pole is found by geometry.

$$L = \frac{18}{\tan(40.06)} = 21.4$$
$$\therefore S = -5$$

Once the pole and the zero are located, the gain of the controller is calculated from Equation (10).

$$kG_c(s)G(s) = 1 \Rightarrow k = \frac{1}{|G(s)||G_c(s)|} \qquad \textbf{(11)}$$

By substituting the magnitudes of all poles and zeros the gain found was k = 2.108

In order to decrease the steady state error, a lag controller was designed. The main objective of a lag compensator is to place a pole and a zero very close to each other and to the origin, so the pole can act like an integrator, but with passive circuitry.

Based on the most common lag controllers [5] the objective was to increase the static constant K_{pc} of the uncompensated system (K_{pu}) by a factor of 5.

$$K_{pc} = 5K_{pu} = K_{pu}\frac{Z_{lag}}{P_{lag}} \qquad \textbf{(12)}$$

$$\Rightarrow \frac{Z_{lag}}{P_{lag}} = 5 \Rightarrow 5P_{lag} = Z_{lag} \qquad \textbf{(13)}$$

By choosing P_{lag} =0.01 then Z_{lag} =0.05

The transfer function of the compensator and the compensated system respectively are presented next.

$$G_c(s) = \frac{2.108(s+3.07)(s+0.05)}{(s+5)(s+0.01)} \qquad \textbf{(14)}$$

$$G_c(s)G(s) = \frac{2.108(s+3.07)(s+0.05)(s+10)}{(s+5)(s+0.01)(s\pm 3.07)} \qquad \textbf{(15)}$$

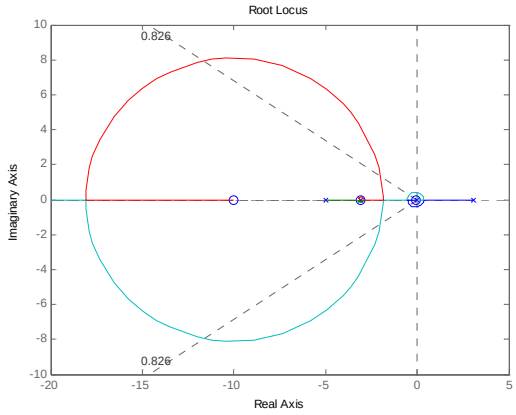


Figure 8. Root locus of the compensated system and the plant

As shown in Figures 6 and 8, the root locus of the compensated and uncompensated systems did not vary considerably because of the pole-zero cancellations generated during the design process of the controller. Note that the lead compensator produced a shift to the left, and such shifting improved the transient response.

Table 2: Classical Control Design

$s_{1,2} = -3.83 \pm 1.71i$
$K = 2.108$
$OS = 0.05\,\text{deg}$
$t_s = 0.75\,\text{sec}$
$\delta_{\max} = 5.1\ \text{deg}$

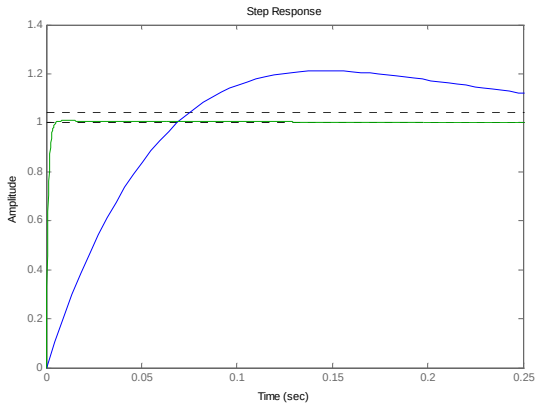


Figure 9. Step response of compensated (green) and uncompensated (blue) systems.

Figure 9 presents the step response of the compensated and uncompensated systems. It can be seen that the transient response was improved after the lead-lag compensation and the steady-state error was decreased from 5% to 1%.

In order to test the system's performance, an initial roll angle of 5 degrees was applied. The instantaneous steering angle required to bring the system upright was measured. Figure 11 shows that to compensate the initial roll angle (Fig. 10), the handle bar had to be rotated 5.1º. This performance meets the design criteria as it can be seen in Table 2.

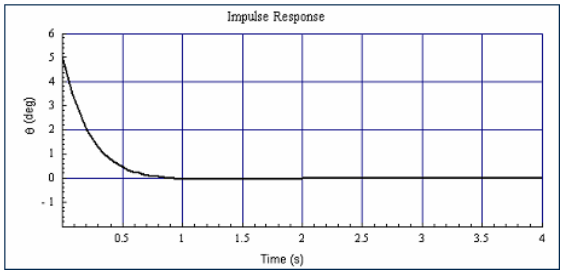


Figure 10. Roll response to initial roll angle of 5º

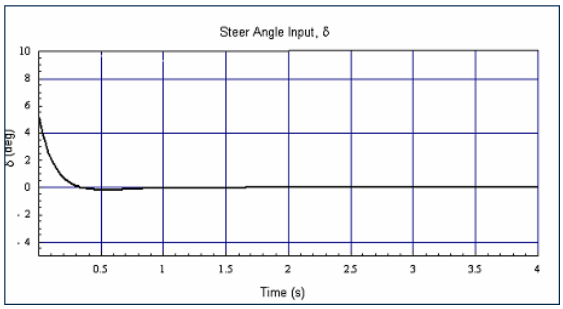


Figure 11. Steer angle used to overcome the initial roll angle

Disturbance Response

A disturbance was included in the system by adding it to the steering angle (see the Simulink diagram in the appendix). A triangle signal with magnitude of 6 was used. With the bicycle upright and at a velocity of 5m/s, Figure 14 shows that the bicycle-rider system leans less than 0.2 degrees as a result of the disturbance. Figure 12 shows the disturbance added to the system. Figure 13 shows that the bicycle initially needs to be steered 6 degrees in the opposite direction of the disturbance turning angle. Then an oversteer almost equal to the magnitude of the roll angle produced is required to bring the system to the upright position.

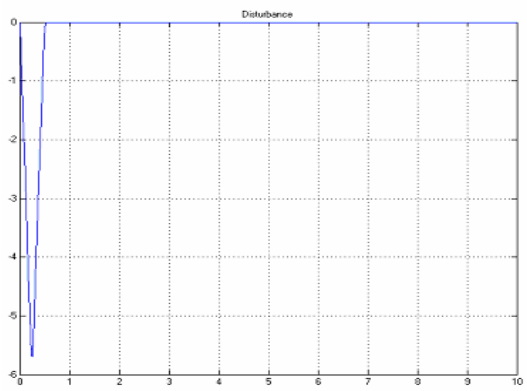


Figure 12. Disturbance signal added to system

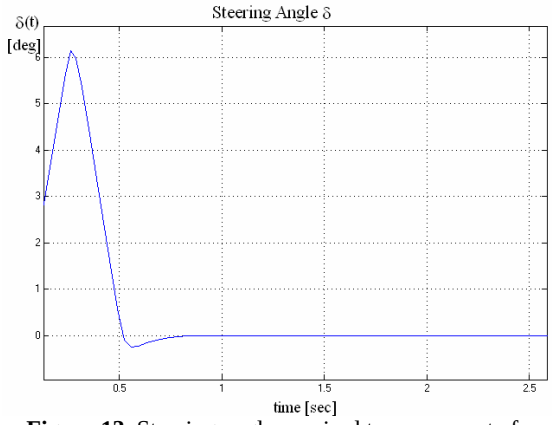


Figure 13. Steering angle required to compensate for a road disturbance

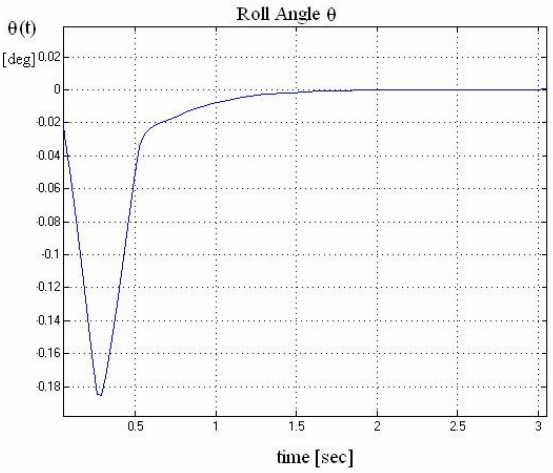


Figure 14. Roll angle caused as a result of disturbance on road

By varying the magnitude of the disturbance, similar behavior was observed. This implies that an instantaneous steering angle of the same magnitude of the disturbance in the opposite direction is required. Then an oversteer equal to the magnitude of the roll angle produced on the system is needed. It should be noted, however, that for large disturbances

the simulation requires steer angles beyond the physical limits of the system. By considering the speed at which the system is traveling (5m/s), it was found in the literature that steering angles above 20 degrees cause roll angles that ultimately cause gyroscopic effects on the system that can lead to instabilities [4]. The simplifications and assumptions used in the linear model do not account for these effects, however, and the simulation allows unrealistic values for steer angle to keep the bicycle upright.

Full State Feedback Regulator

We begin our design of the full state feedback controllers by assuming that the states given by Equations (5) can be measured without error. This assumes, then, that the roll angle, roll rate, and steer angle all can be measured. By also neglecting modeling errors, we can exclude noise inputs to the system. Letting the desired state vector be zero, the control law is simply

$$u(t) = -Kx(t) \qquad \textbf{(16)}$$

where the input, *u(t)*, is the steer angle, δ(t). The closed loop state equations become

$$\dot{x}(t) = (A - BK)x(t). \qquad \textbf{(17)}$$

The matrix (**A**-**BK**) dictates not only the stability of the closed loop system, but also the response of the roll angle output. Thus, the controller gain matrix, **K**, must be designed such that we meet the stability and performance criteria. This is done first by means of placing the two closed loop poles at desired locations in the left half of the complex plane.

Closed-Loop Pole Placement

Design of the controller gain matrix **K** is performed using *Ackerman's pole-placement formula* [2]. The computational steps involved in solving for the **K** matrix are programmed in the MATLAB function *acker*. For this single-input system, the command was used as

```
>>K = acker(A,B,V).
```

Here, **A** and **B** are the state coefficient matrices given in Equations 3 or 4. **V** is a vector containing the two desired closed loop poles. As mentioned earlier, the locations of these closed-loop poles determine the performance when the bike is placed at an initial roll angle. However, the performance criteria can not be translated directly into a second order system damping ratio or natural frequency in the manner of classical root locus design. This is because both states of Equation 5 are being fed back to the controller, yet we are only interested in the response of the first state, x_1, which is the roll angle. Thus, the closed–loop poles are initially selected arbitrarily until the desired response of roll angle is met.

This process of selecting and moving the closed-loop poles around reveals the obvious trade-off that exists between the overshoot and settling time. By moving the poles further inside the left-half plane the initial response was sped up at the expense of increasing the overshoot. This trade-off between response time and overshoot is depicted in Figure 15 below.

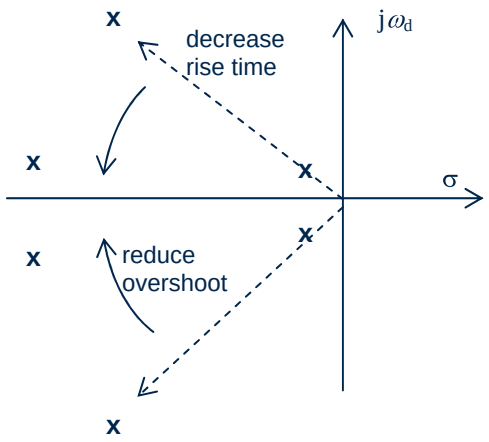


Figure 15. Trade-off between response time and overshoot.

By moving poles to achieve the desired response, we also notice the cost associated with the control input. If a faster response is desired, a larger steer angle is demanded. In some cases, the cost of the steer angle is physically impossible. One example of an excessive front fork angle is shown in Figure 16. In this case, we were able to achieve very good response time with minimal overshoot. However, 53 degrees of *instantaneous* steer angle input was required. This far exceeds the design limit of 20 degrees. It is evident that with higher order systems, arbitrary pole placement can become tedious, at best.

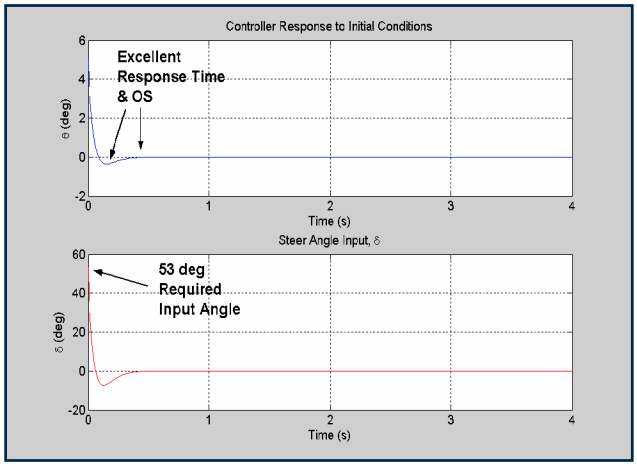


Figure 16. High cost of control input.

One method for placing poles is to employ the *Butterworth pattern* discussed by Tewari [2]. Although well suited for higher order systems, this pattern proved very successful for this second order system. In this pattern, the poles are place on a circle centered at the origin with a radius, *R*. The poles come from the solution of the equation

$$(s/R)^{2n} = (-1)^{n+1} \qquad \textbf{(18)}$$

where *n* is the order of the system. For our simple second order system, the poles are the solutions of

$$\frac{s^2}{R^2} + \frac{\sqrt{2}}{R}s + 1 = 0\,. \qquad \textbf{(19)}$$

By trying various values for the radius, a satisfactory roll angle response was achieved with a cost of 5 degrees of steer angle input. This is shown in Figure 17 below.

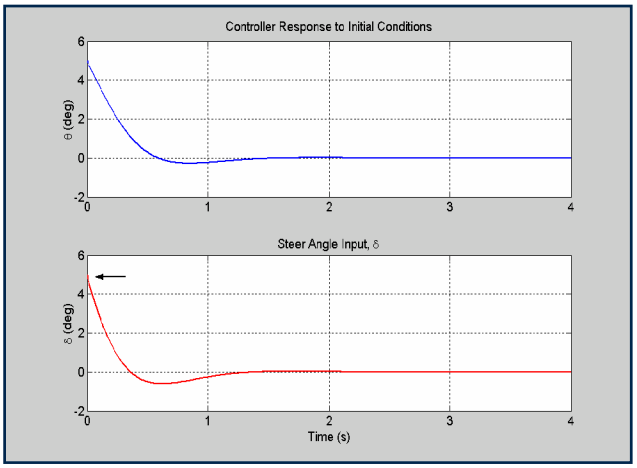


Figure 17. Final design response and input.

The chosen poles and the corresponding gain matrix are listed in Table 3. Note the difference between the gains associated with each state. K_1 is nearly seven times greater than K_2. This may be explained by the fact that our design focuses only on the response of state x_1 (roll angle) and not state x_2.

Table 3: Pole-Placement Design

$$s_{1,2} = -3.01 \pm 3.01i$$
$$K = \begin{bmatrix} -1.001 & -0.150 \end{bmatrix}$$
$$OS = 0.28\,\text{deg}$$
$$t_s = 1.5\,\text{sec}$$
$$\delta_{\max} = 5.0\,\text{deg}$$

LQR Control

We now employ optimal control to design the regulator gain matrix **K**. The LQR method is based on minimizing the quadratic cost functional, J(u).

$$J(u) = \int_0^T \left(\vec{x}^T R\vec{x} + \vec{u}^T \Lambda \vec{u}\right) dt$$

R and Λ are weighting matrices for the states and inputs, respectively. They are both constant and specified by the designer. For the regulator problem with T=∞, it can be shown that minimizing the cost functional gives the optimal input

$$\vec{u}_{opt}(t) = -Kx(t) = -\Lambda B^* P\vec{x}(t)$$

where P is the positive definite solution to the algebraic steady state Riccati Equation. This is only true if the initial conditions are given, R and Λ are both positive definite, (A,B) is controllable, and P(∞) = 0. Our system meets these criteria so this method can be implemented.

The LQR method allows the designer to focus her attention on establishing a cost, or weight, to each state and each input. The weighting is normally guided by the physical nature and limitations of the system parameters. Our system has two states, x_1 and x_2 shown in (5), and one input, δ. We are primarily concerned with x_1 and δ and desire these to both be kept within their physically acceptable ranges. x_2 on

the other hand is a quantity that doesn't have a distinct physical meaning, so we are less concerned with it. With this in mind we were able to begin choosing both **R** and Λ.

Even with a small order system such as ours, the Riccati Equation is nonlinear and thus P is difficult to solve for. MATLAB's function, K=lqr(A,B,R, Λ), solves the nonlinear Riccati equation numerically, thus taking the difficulty out of calculating **K**. We made use of this function and all that was required by the designer was the choices of **R** and Λ.

We began by letting **R** equal the identity matrix and Λ be equal to one. From some trial and error it was determined that simply decreasing the weighting of x_2, yields good results that meet the design criteria. This corresponds to the previous discussion about which parameters are more important for weighting. The state, x_2, is penalized less and the controller doesn't keep it in check as with x_1 and δ. The best results are shown in graphical form in Figure 18 and are also listed in Table 4.

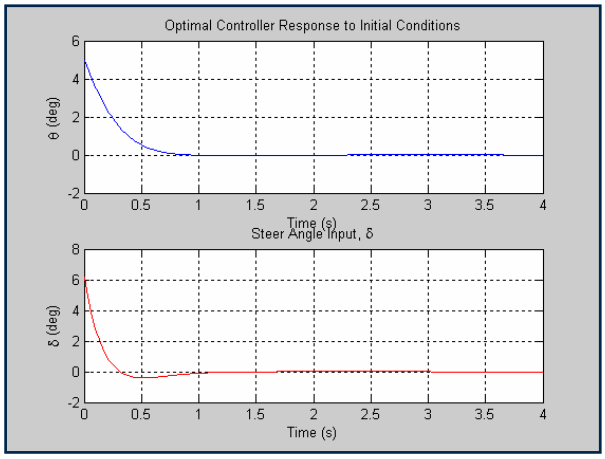


Figure 18. LQR Underdamped Response

Table 4: LQR Design
$s_{1,2} = -4.54 \pm 2.32i$
$K = \begin{bmatrix} -1.231 & -0.254 \end{bmatrix}$
$OS = 0.02\,\text{deg}$
$t_s = 0.86\,\text{sec}$
$\delta_{\max} = 6.2\,\text{deg}$

These results show that a small overshoot and settling time can be achieved with a relatively small input much like the results of the previous two controller designs. But, by manipulating the Λ value the results can be improved slightly. By decreasing Λ the weighting of the input, δ, is decreased. This will effectively let the controller allow a higher value of δ to be used to bring the system into equilibrium at a faster time. The value of δ can then be brought very close to its saturation point. This gives the improved results shown in Figure 19 and Table 5, but at the expense of using more "energy" from the input.

By bringing the steering angle close to its saturation limit, the overshoot can be eliminated. This produces an overdamped response with two negative real poles. This also reduces the settling time slightly. This is an optimum solution, but subject to our lenient saturation point for the steering angle. Figure 19 also shows that the steering rate must be high to achieve this response, which we let go unchecked in our controller. But, if we had realistic specifications for a steering actuator we could determine its limits and keep the controller from requiring an extreme steering angle by properly weighting δ.

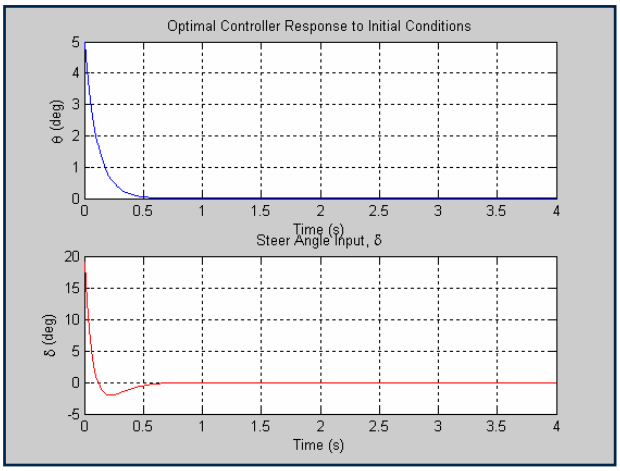


Figure 19. LQR Overdamped Response

Table 5: LQR Design – High δ
$s_1 = -10.05$
$s_2 = -9.14$
$K = \begin{bmatrix} -3.8165 & -.4153 \end{bmatrix}$
$t_s = 0.6\,\text{sec}$
$\delta_{\max} = 19.1\,\text{deg}$

The implementation of the optimal LQR controller met the required design criteria, but was mainly dependent on trial-and-error tactics to find the needed solution. The LQR would more applicable to a

system with state variables and inputs that have a stronger physical meaning and strict limiting criteria.

Control Strategy Comparison

The results of the final controller designs using classical techniques and two state-space methods are summarized in Table 6 below. The poles listed in the Classical column are the dominant poles of the closed loop design. Overall, the classical control strategy achieves the best roll response while minimizing the steer angle input. The LQR controller ranks a close second with its slightly smaller overshoot costing greater steer input and longer settling time.

Table 6. Control Strategy Results

	Classical	**LQR**	**Pole-Placement**
Poles	-3.83±1.71i	-4.54±2.32i	-3.01±3.01i
Gains	2.108	[-1.231 -0.254]	[-1.001 -0.150]
OS	0.05	0.02	0.28
t_s	0.75	0.86	1.5
δ_{max}	5.1	6.2	5.0

The closed loop poles for both state feedback controllers as well as the dominant poles from the root locus design are plotted together in Figure 20.

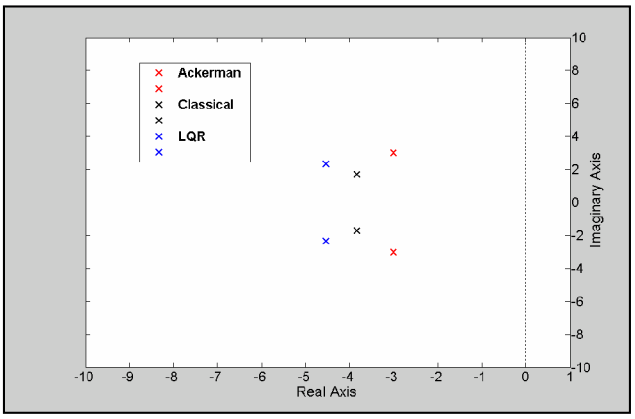


Figure 20. Closed-loop pole locations.

Note that the results from the LQR and the pole placement control design methods yield nearly the same closed loop poles. In fact, by using LQR designed poles in the Ackerman pole placement formula, the same regulator gain matrix will result. Thus, the closed loop matrix (**A**-**BK**) yields the same response.

Required design time and methodology can be examined for further comparison. The classical design method allows the designer to input the design criteria and solve for an "exact" solution for the input parameters. On the other hand, both the pole placement and LQR strategies required much trial and error. Once the solution came close to what was desired, some intuition and more concrete reasoning could be used to either adjust the poles or adjust the weighting matrices to improve the response.

The classical method requires more understanding of the depths of the design process, but for a SISO system like ours it provides a more direct method to the best solution.

Conclusion

For a SISO system classical control methods are hard to beat, for both better results and its unambiguous design methods. The advantages to full state feedback quickly become apparent when the system has multiple inputs and outputs and when dealing with real limited physical systems. Much more complicated bicycle models can be constructed with many more inputs and outputs and more realistic actuators. Once this is done, the classical control method will become less useful and may not be useful at all. The modern control methods discussed in this paper would then have a much clearer advantages to developing a controller for a real bicycle.

References

[1] Karnopp, D. Vehicle Stability. Marcel Dekker. USA. 2004

[2] Tewari, K. S. Modern Control Design with Matlab and Simulink. John Wiley & sons. 2002

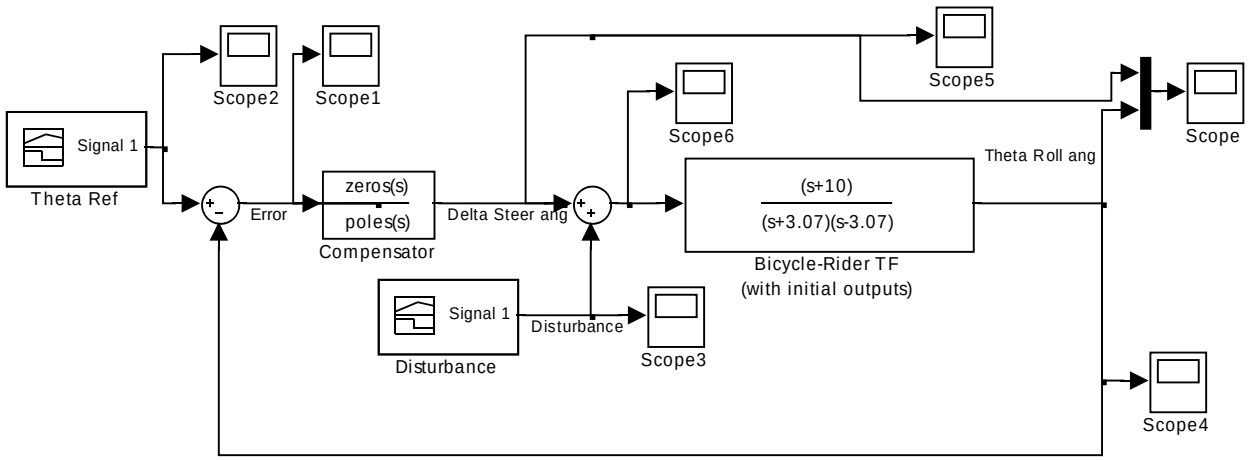
[3] Åström, K.J. et al. Bicycle dynamics and control. IEEE Control Systems Magazine. 25 (4). 26-47. 2005

[4] Fajans. J. Steering in bicycles and motorcycles. American Journal of Physics. 68 (7). 654-659. 2000

[5] Nise, Norman S, Control Systems Engineering, 4th ed. 2004

Appendix

Simulink model of classical control system



Matlab Code for Classical Control

% MAE 272 Classical Control Analysis of bicycle-rider system model

```
m=87;
g=9.81;
I=3.28;
h=1;
a=0.5;
b=1.0;
U=5;

s=tf('s');
G=(m*h*U*(a*s+U))/((I+m*h^2)*s^2-m*g*h);
rlocus(G)
CL=feedback(G,1)
zpk(CL)
ltiview

%Lead & lag control
Gcl=39.65*((s+0.05)*(s+3.07))/((s+5)*(s+0.01));
%Gc=2*(s+.1)*(s+3.07)/((s+0.01)*(s+5.3));
TF=Gcl*G;
rlocus(TF)
sgrid(0.826,0)
CLTF=feedback(TF,1);
```

MATLAB Code for Pole Placement

% 1-DOF Bicycle Model model in both state space and transfer function form.
% Full State Feedback Control using both Pole-placement Designs

```
clear all
```

```
clc
format compact

% Parameters & Constants
degtorad = pi/180;  radtodeg = 180/pi;
g = 9.81;
I = 3.28;   %principle moment of inertia, kg*m^2
m = 87;  %mass, kg
h = 1;   %height, m
a = 0.5;   b = 1.0;
U = 5;  %constant forward velocity, m/s

% Matlab's-Calculated State Space from the system transfer function
num = -m*h/b*[a*U U^2];
den = [(I+m*h^2) 0 -m*g*h];
sys = tf(num,den);
%fprintf('Controller Companion State Space:\n')
[A,B,C,D] = tf2ss(num,den);
System_Poles = eig(A);

% Controllability and Observability
P = ctrb(A,B);  % controllability matrix
rank(P);
N = ctrb(A',C');  %observability matrix
rank(N);

% Convert to Observer Companion Form
Ao=A';  Bo=C';  Co=B';
A=Ao;   B=Bo;   C=Co;
sys = ss(A,B,C,D);

% Design Regulator Gain Matrix, K, w/ pole placement
r = 4.25;      %radius of 'Butterworth' pattern
disp('Closed-loop pole placement:')
% V = roots([1/r^2  sqrt(2)/r  1])  %desired closed loop poles
V = [-4.54 + 2.32i; -4.54 - 2.32i]
K = acker(A,B,V);
ACL_1 = A-B*K;   %closed loop dynamics
sysCL_1 = ss(ACL_1, zeros(2,1), C, D);   %closed-loop state space

% Regulator-Controlled System: Response to IC's
[theta,t,X] = initial(sysCL_1, x_in',t);
theta_deg = theta*radtodeg;
delta = -K*X';   %input steer angle
delta_deg = delta*radtodeg;
OS = min(theta_deg);  %overshoot
tp = t(find(theta_deg==OS));  %peak time
ts = stime(t,theta_deg);    %settling time
delta_max = max(abs(delta_deg));  %largest absolute steer input
fprintf('OS= %4.2f deg,  tp= %3.1f sec,  ts= %3.1f sec,  max input= %4.1f deg\n',OS,tp,ts,delta_max)
figure(2)
subplot(2,1,1),plot(t,theta_deg,'b'),title('Controller Response to Initial Conditions'),xlabel('Time (s)'),...
   ylabel('\theta (deg)'),grid
subplot(2,1,2),plot(t,delta_deg,'r'),title('Steer Angle Input, \delta'),xlabel('Time (s)'),ylabel('\delta (deg)'),grid
```

MATLAB Code for LQR Design

```
clear all
close all
clc
format compact

% Parameters & Constants
degtorad = pi/180; radtodeg = 180/pi;
g = 9.81;    %local acceleration due to gravity, m/s^2
I = 3.28;    %principle moment of inertia, kg*m^2
m = 87;      %mass, kg
h = 1;       %height, m
a = 0.5;     %rear wheel to cg projection, m
b = 1.0;     %wheel base, m
vr = 5;      %constant forward velocity, m/s

%Build Observer Canonical State Space
fprintf('Observer Companion State Space:\n')
A = [[0              1];
    [m*g*h/(I+m*h^2) 0]]
B = [[-m*h*a*vr/b/(I+m*h^2)];
    [-m*h*vr^2/b/(I+m*h^2)]]
C = [1 0]
D = [0]

% Design Regulator Gain Matrix, K, w/ LQR
R = [1 0
    0 1e-2];                    %state-weighting matrix
Lambda = [0.07];                %control cost matrix
[K,S,E] = lqr(A,B,R,Lambda);
K
ACL = A-B*K                     %closed loop dynamics
disp('LQR Closed-loop poles:')
CL_System_Poles = E             %closed loop poles
sysCL = ss(ACL, zeros(2,1), C, D);     %closed-loop state space

% Optimally-Controlled System: Response to IC's
t = (0:0.01:4);   x_in = [5*degtorad 0*degtorad];
[theta,t,X] = initial(sysCL, x_in',t);
theta_deg = theta*radtodeg;
delta = -K*X';    %input steer angle
delta_deg = delta*radtodeg;
OS = min(theta_deg);  % error overshoot
tp = t(find(theta_deg==OS));  % peak time
delta_max = max(abs(delta_deg));  %largest absolute steer input
theta_final=theta_deg(length(theta_deg))
% Calculate the %2 of 1 deg settling time
% Reverse order of theta and t matrices
for i=0:(length(theta_deg)-1)
   theta_flip(i+1)=theta_deg(length(theta_deg)-i);
   t_flip(i+1)=t(length(t)-i);
end
theta_flip=theta_flip';
t_flip=t_flip';
```

```
% Find the first value larger than the criteria
for i=1:length(theta_flip)
   if abs(theta_flip(i))>=0.02
      index=i;
      break
   end
end
ts=t_flip(index) %display the settling time
fprintf('LQR overshoot= %4.2f deg, peak time= %3.1f sec, max input= %4.1f deg\n',OS,tp,delta_max)
figure(2)
subplot(2,1,1),plot(t,theta_deg,'b'),title('Optimal Controller Response to Initial Conditions'),xlabel('Time (s)'),...
   ylabel('\theta (deg)'),grid
subplot(2,1,2),plot(t,delta_deg,'r'),title('Steer Angle Input, \delta'),xlabel('Time (s)'),ylabel('\delta (deg)'),grid
```