Controller Design for a Steerable Self Stabilizing Bicycle

Master of Engineering Project

5/19/2014

Diego Olvera (do98)

Table of Contents

The following work was done in the Robotics and Biomechanics Lab at Cornell University under the supervision of Professor Andy Ruina.

Summary

This report describes the approach undertaken to create a controller for a self-stabilizing bicycle. The model for the bicycle chosen is based on a simplified version of the linearized equations of motion cited in footnote 1. Two controllers were then created using this model: one for bicycle stability, and another for bicycle stability and steering reference tracking. These controllers were then implemented in simulation by using the ode45 Matlab solver and they were shown to meet the specifications.

Motivation

The goal of this project is to create a controller for a bicycle that self-stabilizes. This controller should receive a reference steering angle and then make the bicycle follow this reference while staying upright.

In the future it is desired to implement the controller on a real bicycle, which will be able to self-stabilize and navigate using position data. This controller will also be used to create a steer-by-wire bicycle in which the rider turns a handlebar that is essentially a joystick. The microcontroller implementing the controller will then turn the steering wheel as needed, thus causing the bicycle to turn stably. Such a bicycle will be able to be ridden by people who do not know how to ride a bicycle or who cannot stabilize it themselves.

Dynamics Model

Linearized Equations of Motion

The Dynamics model of a bicycle used here are based on the linearized equation of motion described in a paper co-written by professor Andy Ruina[1]. These equations are as follows:

$$[1] \qquad \begin{bmatrix}\ddot{\phi}\\ \ddot{\delta}\end{bmatrix} = M^{-1}(-C\begin{bmatrix}\dot{\phi}\\ \dot{\delta}\end{bmatrix} - K\begin{bmatrix}\phi\\ \delta\end{bmatrix} + \begin{bmatrix}T_\phi\\ T_\delta\end{bmatrix})$$

$$[2] \qquad \dot{\psi} = \frac{v\delta + c\dot{\delta}}{w}\cos(\lambda)$$

$$[3] \qquad \dot{x} = v\cos(\psi)$$

$$[4] \qquad \dot{y} = v\sin(\psi)$$

where:

$\phi = roll\ angle$ $\psi = yaw\ angle\ (heading)$

$\delta = steer\ angle$ $x = x\ position\ of\ back\ wheel\ with\ respect\ to\ a\ fixed frame$

$T_\phi = roll\ torque$ $y = y\ position\ of\ back\ wheel\ with\ respect\ to\ a\ fixed frame$

$T_\delta = steer\ torque$ $v = forward\ velocity\ of\ bicycle\ (assumed\ constant)$

$w = wheelbase$ $\lambda = steer\ axis\ tilt$

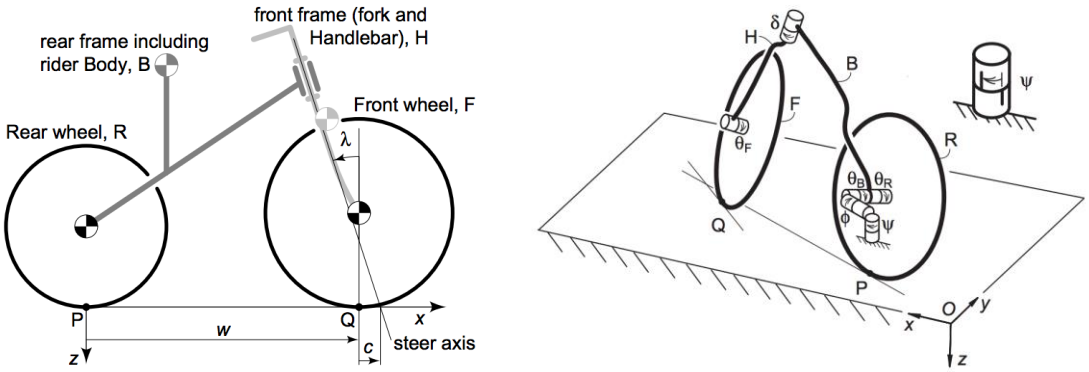


Figure 1: Bicycle Coordinate Axes [Figures from paper cited in 1]

It should be noted that this model assumes a constant forward velocity. In practice this would be ensured by another controller acting on a hub motor attached to one of the wheels guaranteeing a constant rotation rate of the wheel.

The matrices M, C and K are obtained from the parameters of the bicycle being used. (See appendix A for the values used to model the bicycle which will be used for this specific project.

---

[1] Linearized dynamics equations for the balance and steer of a bicycle *J.P. Meijaard, Jim M. Papadopoulos, Andy Ruina, A.L. Schwab*

As will be seen later, many of these parameters will be ignored except for some geometric parameters and the overall mass of the bicycle)

For more detail on how to obtain these matrices please see the cited paper. A Matlab script was written which obtains these matrices for a bicycle with any parameters of mass, inertia, and dimension given. The matrices from this script were compared the paper's benchmark values, thus confirming their validity.

Equations 2, 3, and 4 above are mostly for simulation purposes. Equation 1, which describes the dynamics of the bicycle's roll angle and steer angle, is the one which will inform how the controller for the bicycle will be created.

Point Mass Bicycle

The linearized equations of motion above are a good start, but they can be difficult to use for controls purposes because these equations deal with the steering torque, a state that is difficult to control and measure in a real bicycle. Upon a suggestion by professor Ruina, the following simplifications to the model were made:

- The front frame has no inertia
- A vertical head angle ($\lambda = 0$)
- No trail ($c = 0$)
- Rear wheel has no inertia
- Rear frame is a point mass
- Front wheel has no mass or inertia

These simplifications essentially turn the bicycle into a point mass, thus eliminating the individual inertias of the bicycle's components and leaving only a large steerable, point mass "hinge" satisfying the rolling constraint.

Because these simplifications remove the interactions between the steering torque and inertia, the second equation ($\ddot{\delta}$) becomes meaningless and can be removed. This leaves the following equation:

$$[5] \qquad \ddot{\phi} = -MiK_{11}\phi - MiK_{12}\delta - MiC_{12}\dot{\delta} + Mi_{11}T_{\phi}$$

where MiK_{11} is the element in the first row and first column of the matrix produced by left multiplying the inverse of the M matrix times the K matrix in the linearized equations.

Equation 5 is what will be used for controls formulation.

Validity of the point mass model

It may not be immediately obvious that this simplification yields a valid model. To test this, a controller was created for stabilizing the bicycle and it was implemented using the linearized equations and the simplified equations. The results are compared in Figure 2.

As can be seen, the dynamics vary slightly with the largest differences being in the overshoot of each of the states.

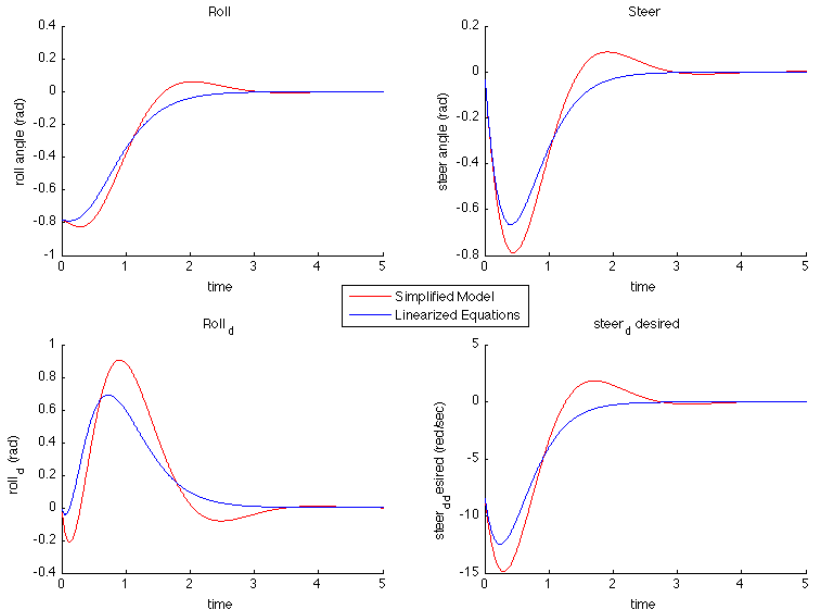


Figure 2: Comparison of the Linearized Equations and the Point Mass Bicycle Equations

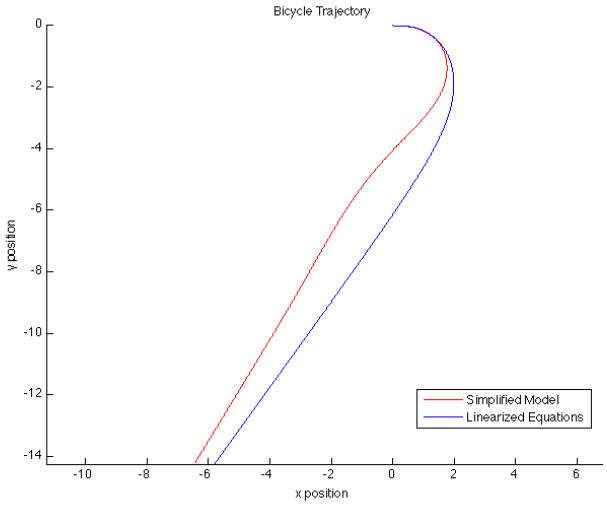


Figure 3: Comparison of the Linearized Equations and the Point Mass Bicycle Equations

A B and C form

The controls strategy used here will involve assuming that a servo-motor mounted to the steering wheel can provide any steering angular velocity desired. In this way the $\dot{\delta}$ variable in equation 5 can be treated as a control variable as well as a state. This leads to the following dynamics formulation:

$$[6] \qquad \begin{bmatrix} \dot{\phi} \\ \dot{\delta} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -MiK_{11} & -MiK_{12} & 0 \end{bmatrix} \begin{bmatrix} \phi \\ \delta \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ -MiC_{12} \end{bmatrix} \dot{\delta} + \begin{bmatrix} 0 \\ 0 \\ Mi_{11} \end{bmatrix} T_{\phi}$$

or

$$[7] \qquad \dot{z} = Az + Bu + D\,T_{dist}$$

The outputs of interest are the roll and steer angles so the output variable is:

$$y = Cz$$

where:

$$C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Control Design For Self Stabilization

Now that a model is obtained, a controller can be designed to stabilize the bicycle.

Specifications

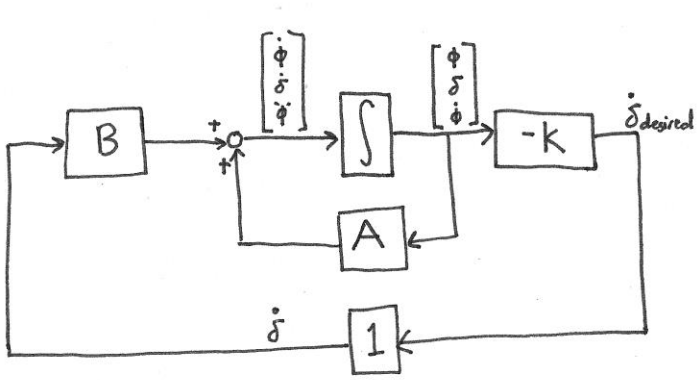
In order to guide the control design process, a set of desired specifications are listed for the overall system dynamics. Firstly, the bicycle is required to have a 5% settling time smaller than 1 second to a step input, ensuring a fast response. The steering angular velocity is also desired to be kept below 7 π radians per second to ensure that the servomotor can provide this actuation. This limit can be imposed on a step response of size $\frac{\pi}{4}$ radians to represent a worst-case scenario when the bicycle is operational.

Specs

- 1 second 5% settling time
- steering angular velocity < 7π as response to a step input of $\pi/4$

Block Diagram

Knowing the system dynamics of the bicycle and the specifications, the overall system can be put into block diagram form:



It is desired to find a K matrix that will stabilize the bicycle and meet the specifications.

Controller Design Using CVX

CVX is a convex optimization toolbox for Matlab, and it is what will be used here to create controllers and observers that meet the required specifications. This technique for controller design states the creation of a closed loop system as an optimization problem, which must satisfy a set of linear matrix inequalities. Please refer to Appendix B for details on this process.

CVX outputs the following controller when given the mentioned specs and a point mass bicycle model with parameters listed in Appendix A.

$$[8] \qquad K = [-9.4964 \quad 5.9447 \quad -2.8767]$$

Closed Loop System Simulation

Using the controller listed in equation 8, a simulation is made of the closed loop system with the initial conditions of the bike as shown below:

$$\phi_0 = \frac{-\pi}{4}$$

$$\delta_0 = 0$$

$$\dot{\phi}_0 = 0$$

The results are shown in figures 4 to 6.

For more details on how the animation was achieved, please see Appendix C.

As can be seen, the system meets the settling time requirement of 1 second and the steering angular velocity does not exceed 7π in magnitude. The states of the bicycle also stay within reasonable values, the steering angle does not exceed $\pi/2$ and the roll angle does not fluctuate on its way to the upright position.

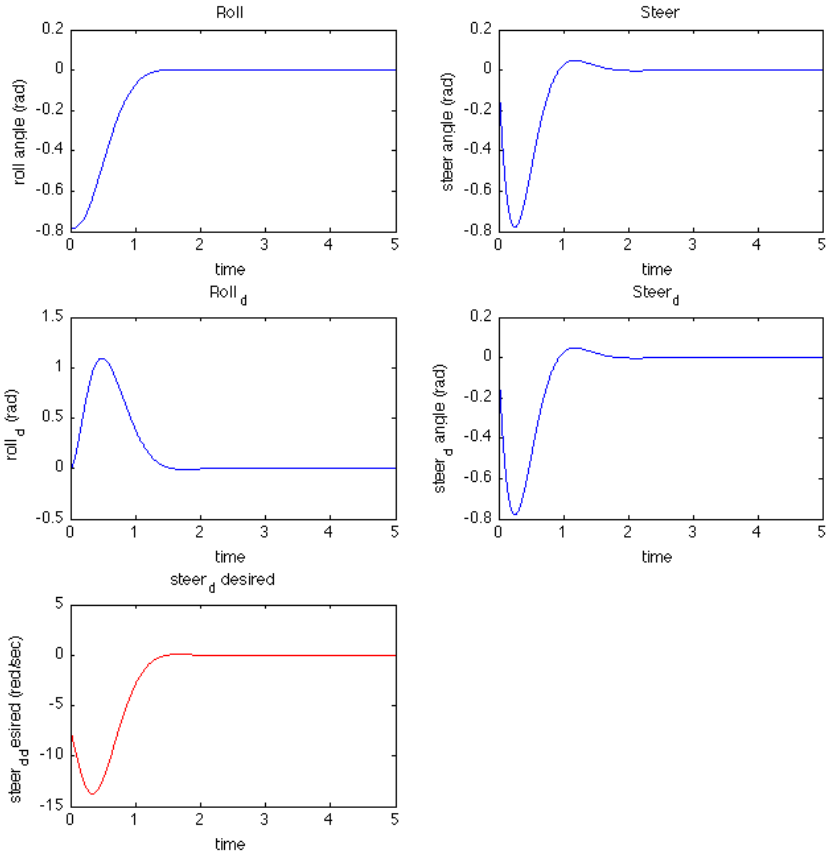


Figure 4: ODE Simulation of Closed Loop System

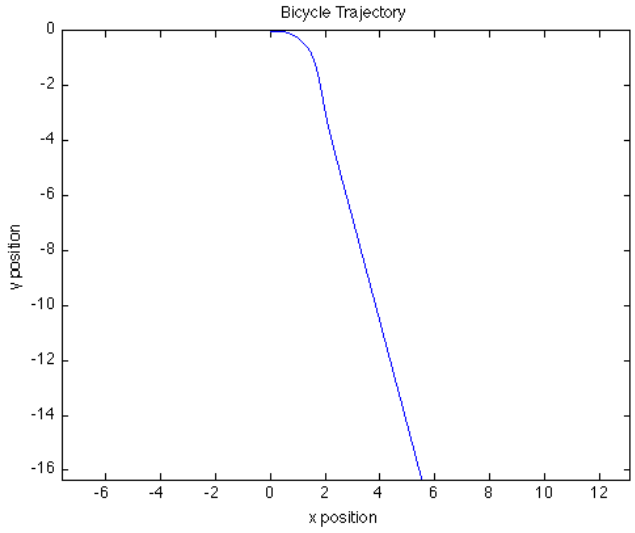


Figure 5: Bicycle Trajectory

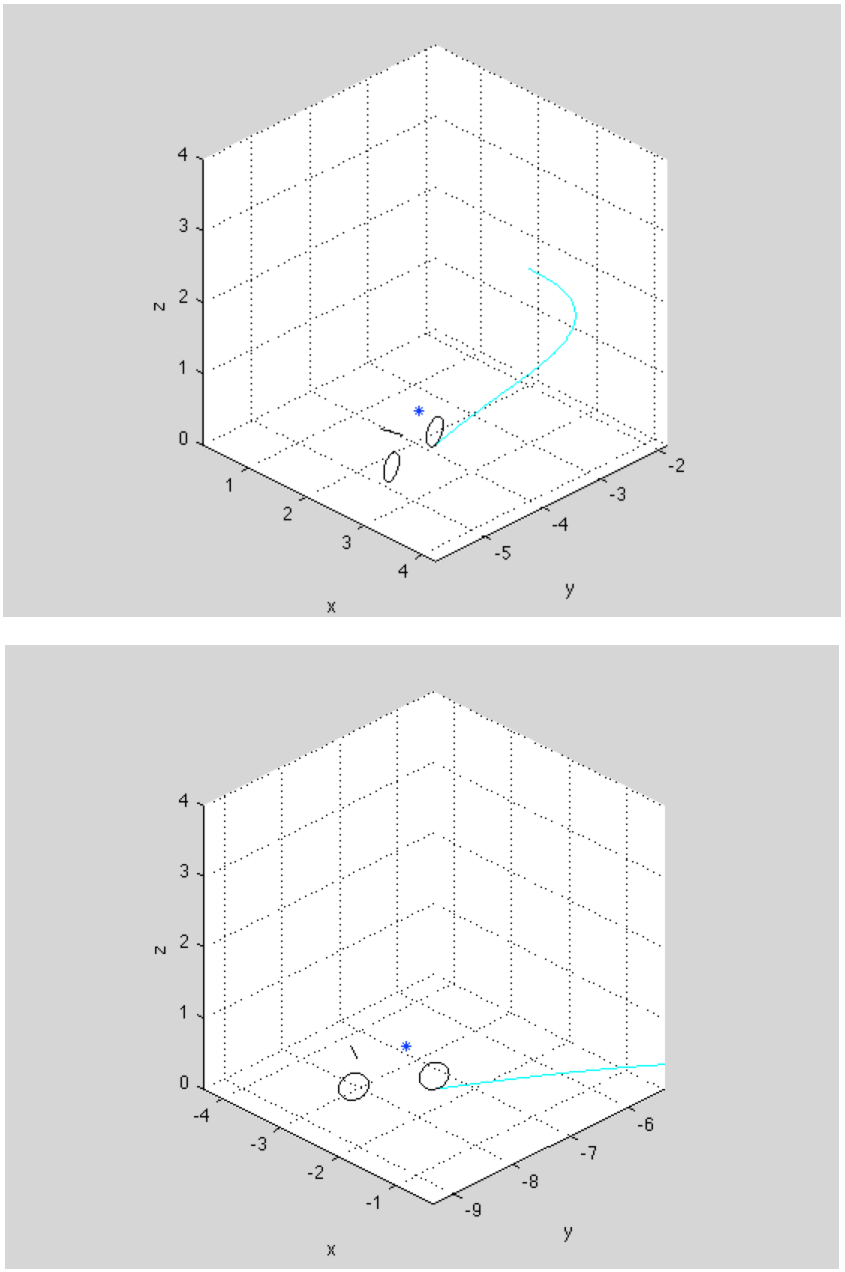


Figure 6: Animation of ODE simulation

Observer Based Controller With Reference Tracking

The ultimate goal of this project is to create a bicycle which can track a steering reference and self-stabilize. While the controller mentioned in the previous section is adequate for stabilization, it does not track a steering reference and does not provide integral control to guarantee zero steady state error. In order to do this, we can refine the specifications mentioned previously:

Specs

- 1 second 5% settling time
- steering angular velocity < 7π as response to a step input of $\pi/4$
- zero steady state steering angle reference tracking

The approach here will be to create new states which will be the integral of the reference error, and a controller will be designed for reference tracking. Because these new states are not easy to sense in a practical setting, an observer will also be designed to provide full state knowledge to the controller. The observer will be made to converge to the real states much faster than the controller dynamics to ensure the separation principle holds. Finally, this system will be simulated using an ode45 solver.

Figure 7 shows a block diagram of the overall system approach. The system in the dotted box is the system observer.

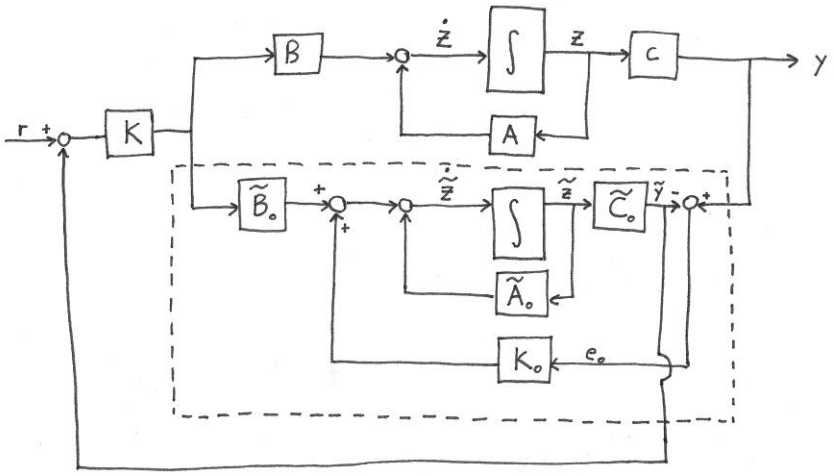


Figure 7: Observer Controller For Reference Tracking Block Diagram

Augmented System

In order to meet the mentioned specifications, we can re-formulate our dynamics system to include two new states:

$$e_{ref} = \int_0^t r - C\,z$$

where

$$r = \begin{bmatrix} \phi_{ref} \\ \delta_{ref} \end{bmatrix} = \begin{bmatrix} 0 \\ \delta_{ref} \end{bmatrix}$$

is the reference signal, and e_{ref} is the integral of the error between the current states and the reference states. The new augmented system takes the form:

$$\begin{bmatrix} \dot{z} \\ \dot{e}_{ref} \end{bmatrix} = \begin{bmatrix} A & 0 \\ -C & 0 \end{bmatrix} \begin{bmatrix} z \\ e_{ref} \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u + \begin{bmatrix} 0 \\ I \end{bmatrix} r$$

or:

$$\begin{bmatrix} \dot{z} \\ \dot{e}_{ref} \end{bmatrix} = \dot{\tilde{z}} = \tilde{A}\tilde{z} + \tilde{B}u + B_{ref}r$$

Controller Design Using CVX for Reference Tracking

Using these new $\tilde{A}$ and $\tilde{B}$ matrices, the procedure described in in Appendix B can be used to create a controller which uses all of these states to track the steering reference and maintain stability.

CVX outputs the following controller. As mentioned before, it is using the point mass bicycle model with values in Appendix A.

$$K = [-22.7218 \quad 9.1335 \quad -2.5151 \quad 47.6035 \quad -29.9128]$$

Observer Design Using CVX

In order to obtain all of the states needed for the feedback controller above, an observer must be created to approximate the real states of the system. This problem essentially reduces to creating an observer feedback matrix K_o which is guaranteed to converge to the real states of the system.

$$e_{obs} = [A - K_o C]e_{obs}$$

This observer can be designed using CVX and the process for doing this is detailed in Appendix B. The only thing required of this observer is that it obtains a 5% settling time of less than 0.1 seconds so that it does not interfere with the control dynamics.

CVX outputs the following observer gain:

$$K_o = \begin{bmatrix} 0.1559 & -0.0007 \\ 0.0004 & 0.0443 \\ 6.1965 & -0.0401 \end{bmatrix} * 10^3$$

however, because we have two unobservable states in our observer, the K_o matrix must be modified slightly for the matrix dimensions to match.

$$K_o = \begin{bmatrix} 0.1559 & -0.0007 \\ 0.0004 & 0.0443 \\ 6.1965 & -0.0401 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} * 10^3$$

Simulation of Observer and Controller

In order to simulate the response of the observer controller, the whole system can be put into the following form:

$$\begin{bmatrix} \dot{z} \\ \dot{\tilde{z}} \end{bmatrix} = \begin{bmatrix} A & -BK \\ K_o C & (\tilde{A} - \tilde{B}K - K_o \tilde{C}) \end{bmatrix} \begin{bmatrix} z \\ \tilde{z} \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix} r$$

and simulated using the Matlab ode45 solver.

For this simulation the following reference signals and initial conditions are used:

$$z_0 = \begin{bmatrix} \phi_0 \\ \dot{\phi}_0 \\ \delta_0 \end{bmatrix} = \begin{bmatrix} \frac{\pi}{8} \\ 0 \\ 0 \end{bmatrix} \qquad \tilde{z}_0 = \begin{bmatrix} \phi_{obs} \\ \dot{\phi}_{obs} \\ \delta_{obs} \\ e_{\phi ref} \\ e_{\delta ref} \end{bmatrix} = \begin{bmatrix} 0 \\ 0.1 \\ 0.2 \\ 0 \\ 0 \end{bmatrix}$$

Figures 8 to 11 show the system response. As can be seen, the observer converges to the real states within a tenth of a second, the bicycle stabilizes in less than 1 second and at steady state, the bicycle travels in a circle, as expected.

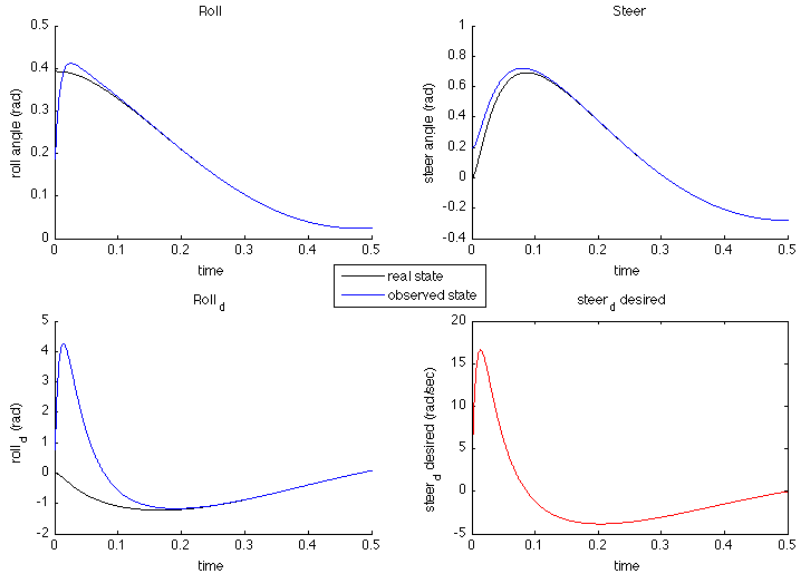


Figure 8: Observer Transient Behavior (on the Timescale of Milliseconds)

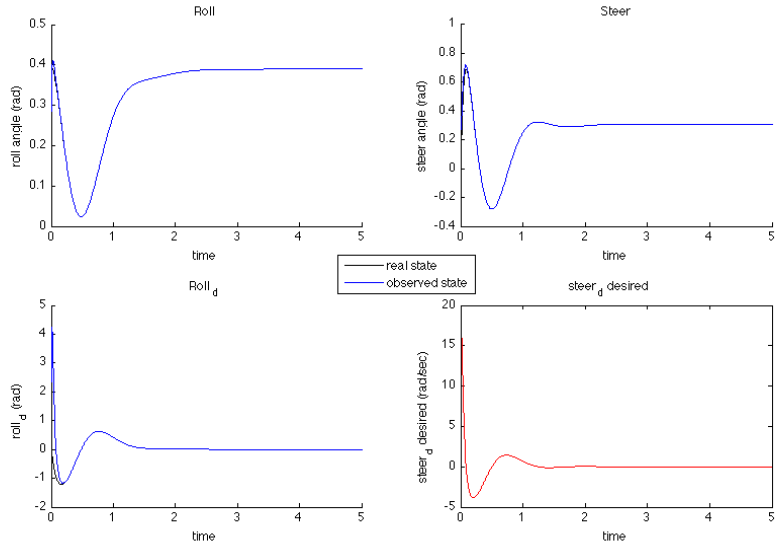


Figure 9: Controller Transient Behavior

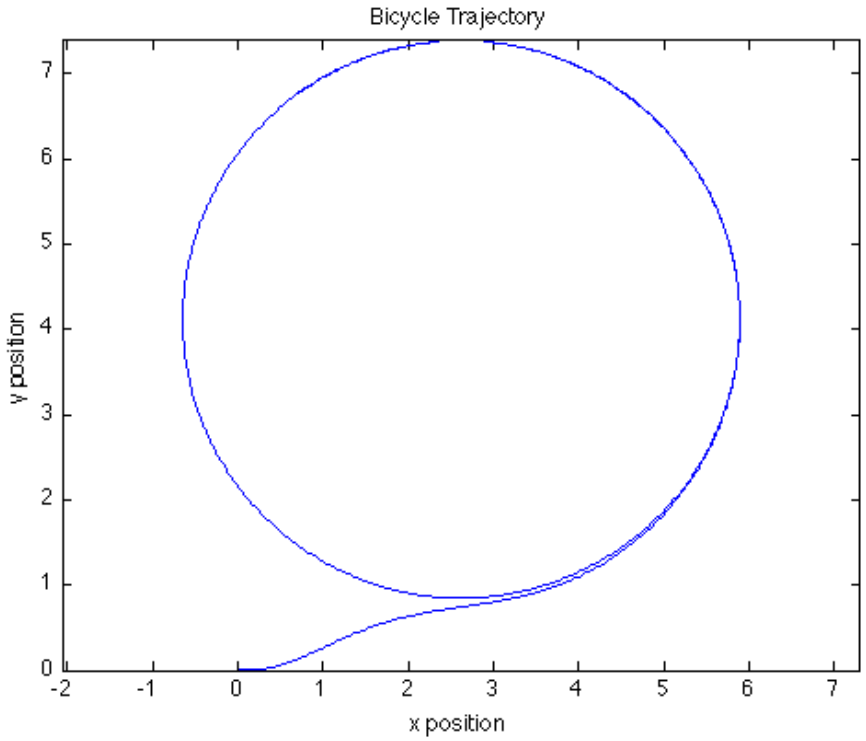


Figure 10: Bicycle Trajectory (10 second simulation)

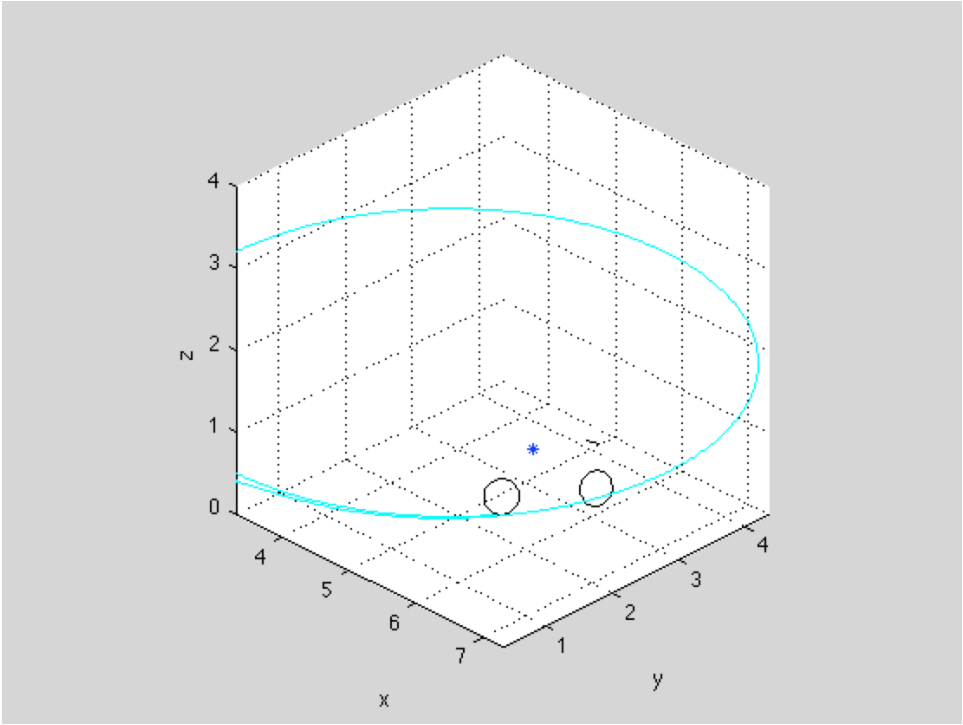


Figure 11: Reference Tracking Simulation

Comparing the Controls Models

Two controllers were described in this report and were demonstrated to meet the specifications required. However, these two models differ in ways the deserve mentioning.

The controller for self-stabilization, while only stabilizing the bicycle without reference tracking, is a very easy controller to implement on a real microcontroller. If all of the states can be provided by sensors, then all that is needed to implement the control law is a one line operation:

$$u = -Kz$$

The second controller is a bit more complicated to implement. Because it involves an observer, an integrator is needed to predict the states of the bicycle. However, this controller is much more complete and allows for steering reference tracking witch is our goal.

Conclusion

This report has detailed the approach taken towards creating a self-stabilizing bicycle. A simplified point mass model was described, and its validity was confirmed by comparing it to the linearized equations of motion for a bicycle. Then, this point mass model was used to create two different kinds of controllers: one for stability and another for stability and steering reference tracking. The controllers were then simulated and shown to meet the required specifications.

Appendix A:
Bicycle Model Parameters

These parameters are based on measurements of the bicycle, which is desired to stabilize. Some of these parameters were not changed from the benchmark values listed in the paper cited in footnote 1 because no effort was made yet to measure these values from the real bicycle.

	Parameter	Symbol	Value
	Wheel base	w	1.02 m
	Trail	c	0.08 m
	Steer Axis Tilt	λ	π/10
	Gravity	g	9.81 m/s^2
	Forward Speed	v	3.57 m/s
Rear Wheel	Wheel Radius	r_R	0.1905 m
	Wheel Mass	m_R	6.5 kg
	Mass Moments of Inertia	(I_{Rxx},I_{Ryy})	$(0.0603,0.12)*m_R/2$ kg*m^2
Rear Body and Frame Assembly	Position Center of Mass	(X_B,z_B)	(0.3,-0.9) m
	Mass	m_B	15.65 kg
	Mass Moments of Inertia	$\begin{bmatrix} I_{Bxx} & 0 & I_{Bxz} \\ 0 & I_{Byy} & 0 \\ I_{Bxz} & 0 & I_{Bzz} \end{bmatrix}$	$\begin{bmatrix} 9.2 & 0 & 2.4 \\ 0 & 11 & 0 \\ 2.4 & 0 & 2.8 \end{bmatrix}*m_B/85$
Front Handlebar and Fork Assembly	Position Center of Mass	(X_H,z_H)	0.9 m
	Mass	m_H	4 kg
	Mass Moments of Inertia	$\begin{bmatrix} I_{Hxx} & 0 & I_{Hxz} \\ 0 & I_{Hyy} & 0 \\ I_{Hxz} & 0 & I_{Hzz} \end{bmatrix}$	$\begin{bmatrix} 0.05892 & 0 & -0.00756 \\ 0 & 0.06 & 0 \\ -0.00756 & 0 & 0.00708 \end{bmatrix}$
Front Wheel	Radius	r_F	0.1905 m
	Mass	m_F	1.81 kg
	Moments of Inertia	(I_{Fxx},I_{Fyy})	$(0.1405,0.28)*m_F/3$

Appendix B:
Controller and Observer Design Using CVX

Convex Optimization

CVX[2] is a convex optimization toolbox for Matlab. This toolbox can be used for the design of control algorithms by stating the creation of the controller K as an optimization problem in which the closed loop system is required to meet any set of given specs.

Controller Design Using CVX

For this section, a dynamics model of the following form will be assumed:

$$\dot{z} = Az + Bu$$

subject to a control law:

$$u = -Kz$$

Solving for K is stated as a convex optimization by four linear matrix inequalities. One of these linear matrix inequalities guarantees Lyapunov stability of the closed loop system, LMIs two and three, along with the first, place bounds on the control effort used for a given set of initial conditions, and the fourth places limits on the settling time of the closed loop system.

Linear Matrix Inequality 1 – Guarantee Lyapunov Stability

This first LMI aims to ensure that the closed loop system will be a Lyapunov stable system that settles to our desired stability point. That is, it will ensure that our system will always head in the direction of the desired final states. To do this, a value function is defined as such:

$$V = z^T P z$$

where z is a vector of our states and P is a cost matrix to be determined. The derivative of this function is:

$$\dot{V} = \dot{z}^T P z + z^T P \dot{z}$$

$$\dot{V} = z^T (A_{cl}^T P + P A_{cl}) z$$

where:

$$A_{cl} = A - BK$$

In order to ensure Lyapunov stability the following properties are required:

$$V > 0$$

$$\dot{V} < 0$$

[2] http://cvxr.com/cvx/ CVX Matlab Convex Optimization *Michael C. Grant, Stephen P. Boyd*

so that the value function to always heads to zero. In order for this to be the case, P must be a positive definite matrix, and $(A_{cl}^T P + P A_{cl})$ must be a negative definite matrix.

$$P > 0$$

$$(A_{cl}^T P + P A_{cl}) < 0$$

This can be guaranteed by creating P and K in the right way. However, P and K appear nonlinearly in the above inequalities. In order to do this we can restate our problem as one inequality:

$$\begin{bmatrix} -(A_{cl}^T P + P A_{cl}) & 0 \\ 0 & P \end{bmatrix} > 0$$

and apply a congruence transformation:

$$\begin{bmatrix} P^{-1} & 0 \\ 0 & P^{-1} \end{bmatrix} \begin{bmatrix} -(A_{cl}^T P + P A_{cl}) & 0 \\ 0 & P \end{bmatrix} \begin{bmatrix} P^{-1} & 0 \\ 0 & P^{-1} \end{bmatrix}^T > 0$$

$$\begin{bmatrix} -(P^{-1}A^T - P^{-1}K^T B^T + AP^{-1} - BKP^{-1}) & 0 \\ 0 & (P^{-1})^T \end{bmatrix} > 0$$

Now, as can be seen, the terms P^{-1} and KP^{-1} appear linearly. Applying a change of coordinates:

$$J = P^{-1}K^T$$

$$Y = P^{-1}$$

the expression becomes:

[9] $$\begin{bmatrix} -(YA^T - JB^T + AY - BJ^T) & 0 \\ 0 & Y^T \end{bmatrix} > 0$$

Using CVX we can solve for a J and Y that satisfy this LMI and thus find K and P:

$$P = Y^{-1}$$

$$K = (PJ)^T$$

Linear Matrix Inequalities 2 and 3 – Minimize Control Effort

Because the value "V" from the Lyapunov stability definition is a non-physical value, we can use it to create bounds on the control effort u. This can be done by way of the following inequality:

$$u^2 < V(z) < V(z0) < \gamma^2$$

where u is the control effort, $V(z0)$ is the first value of the system in time, and γ is some spec we choose.

LMI 1

The inequality

$$V(z) < V(z0)$$

is already guaranteed by Lyapunov stability, but the first and third each require an LMI.

LMI 2

The inequality

$$V(z0) < \gamma^2$$

Ensures that the first cost of the Lyapunov function is lower than the square of our control specification.

This can be rewritten as:

$$0 > -\gamma^2 + z_0^T P z_0$$

applying a Schur complement:

$$\begin{bmatrix} -\gamma^2 & z_0^T \\ z_0 & -P^{-1} \end{bmatrix} < 0$$

and using the same change of variables as before

$$[10] \quad \begin{bmatrix} -\gamma^2 & z_0^T \\ z_0 & -Y \end{bmatrix} < 0$$

Now that this inequality is linear with respect to Y it can be put into the CVX solver. The value γ and the initial conditions matrix z_0 will be specs we input as required max control effort for a given initial condition.

LMI 3

The third inequality

$$u^2 < V(z)$$

can be rewritten as:

$$z^T K^T K z < z^T P z$$

$$K^T K < P$$

Applying a congruence transformation:

$$P^{-1} K^T K (P^{-1})^T < (P^{-1})^T$$

and changing variables once again:

$$J J^T - Y < 0$$

And, applying a Schur Complement:

$$[11] \quad \begin{bmatrix} Y & J \\ J^T & I \end{bmatrix} > 0$$

This LMI now ensures the control effort inequality.

Linear Matrix Inequality 4– Ensure Settling Time Spec

In order to ensure that our closed loop system has a settling time of less than t_s, the system must have poles to the left of:

$$\sigma = \mu w_n = \frac{3}{t_s}$$

in the complex plane, where t_s is the 5% settling time spec. This approach is typically used to check how well a closed loop system will respond to a step input by looking at where the closed loop poles are, however we can turn this into an LMI spec to require that our closed loop system meet this spec. This is done by way of the following inequality:

$$P(A_{cl} + \sigma I) + (A_{cl} + \sigma I)^T P < 0$$

which can be rewritten as:

$$PA_b - PBK + \sigma P + A_b^T P - K^T B^T P + \sigma P < 0$$

Applying a congruence transformation

$$P^{-1}[PA_b - PBK + \sigma P + A_b^T P - K^T B^T P + \sigma P]P^{-1^T} < 0$$

$$-(A_b P^{-1^T} - BKP^{-1^T} + P^{-1}A_b^T - P^{-1}K^T B^T + 2\sigma P^{-1^T}) > 0$$

and a change of variables:

$$[12] \quad -(A_b Y - BJ^T + YA_b^T - JB^T + 2\sigma Y) > 0$$

Equation 11 is now an LMI that will ensure that the settling time spec is met.

Using CVX

Equations 9 to 12 are now ready to be put into the CVX solver to create a P and K matrix.

Figure 12 shows the LMIs in the convex optimization solver. As can be seen, the value "control_max_sqrd" is input as a variable to minimize and the settling time spec "ts" is strictly specified. This is because any controller created must make a trade-off between these two specs. My making one of the two specs a requirement and another an optimization parameter, we can control what CVX optimizes for.

```
%% CVX

ts=2;

epsilon=0.0000001;

cvx_begin sdp quiet
variable Y(n,n)
variable J(n,m)
variable control_max_sqrd

[-(Y*Ab'-J*Bb'+Ab *Y'-Bb*J'),zeros([n,n]);...
 zeros([n,n]),Y'] >epsilon*eye(2*n)  %Guarantee Lyapunov stability

[-(control_max_sqrd), x0';...
  x0,-Y]<-epsilon*eye(n+1)              %Guarantee first cost is
                                      %lower than (10^2)

-(Ab*Y'-Bb*J'+Y*Ab'-J*Bb'+(2*(3/ts)*Y))>epsilon*eye(n)   %Guarantee 5%
                                                         %settling time
minimize(control_max_sqrd)

[Y',J;...
J',eye(m)]>epsilon*eye(n+m)  %Guarantee that all costs are less than
                               %the initial cost

cvx_end

P=Y\eye(n)

K=J'*P

gamma0=x0'*P*x0;    %Initial cost
```

Figure 12: Matlab code specifying LMIs to CVX

Observer Design Using CVX

The design of an observer can be stated as the creation of a closed loop system of the following form:

$$\dot{e} = [A - K_o C]e$$

$$e = z - z_{obs}$$

Where K_o is the observer gain matrix desired, e is the error between the real states and the observed states, and A and C are the system matrices.

Observer Lyapunov Stability

In order to ensure Lyapunov stability of the observer, a procedure similar to the one stated for the controller can be used to create an LMI for this spec:

$$P > 0$$

$$(A_{cl}^T P + P A_{cl}) < 0$$

Writing these as one inequality

$$\begin{bmatrix} -(A_{cl}^T P + P A_{cl}) & 0 \\ 0 & P \end{bmatrix} > 0$$

$$\begin{bmatrix} -(A^T P - C^T K_o^T P + PA - PK_o C) & 0 \\ 0 & P \end{bmatrix} > 0$$

Using a change of variables:

$$J = PK_o$$

$$[13] \quad \begin{bmatrix} -(A^T P - C^T J^T + PA - JC) & 0 \\ 0 & P \end{bmatrix} > 0$$

Equation 13 will ensure Lyapunov stability of the observer.

Observer Settling Time

In order for the separation principle to hold when using an observer, the observer settling time must be much smaller than the controller settling time, and so specifying a settling time for the observer is important. This can be done in a way similar to the controller settling time LMI:

$$P(A_{cl} + \sigma I) + (A_{cl} + \sigma I)^T P < 0$$

$$-(PA - PK_o C + \sigma P + A^T P - C^T K_o^T P + \sigma P) > 0$$

using a change of variables:

$$J = PK_o$$

$$[13] \quad -(PA - JC + \sigma P + A^T P - C^T J^T + \sigma P) > 0$$

Equation 13 guarantees the settling time spec for the observer closed loop system.

Appendix C:
Bicycle Simulator

The bicycle simulator created in Matlab for animating the solutions to the dynamics uses a single function repeatedly to plot the pose of the bicycle.

The function

[COG_hand,SH_hand,CPfw_hand,CPrw_hand]=DrawBikePose(x,y,z,yaw,roll,steer)

takes in the x,y and z coordinates of the bicycle's rear wheel contact point, the bicycle's yaw, roll and steer angles, and outputs the object handles of the plotted bicycle components.

This function achieves this plotting by way of a series of matrix multiplications to rotate and translate a series of points. The following dimensions and frames will be used:

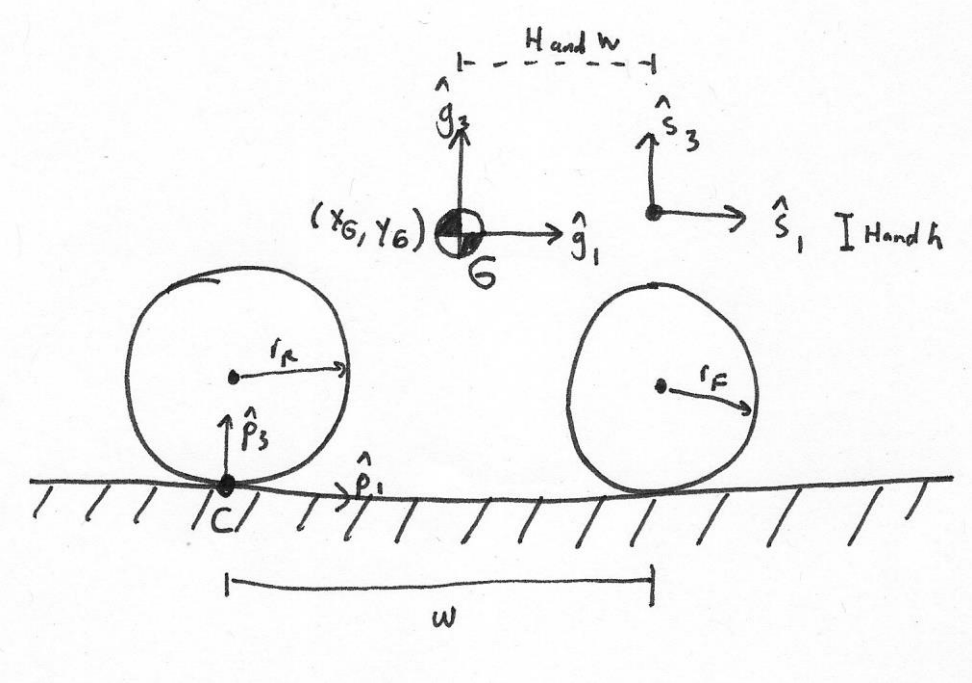
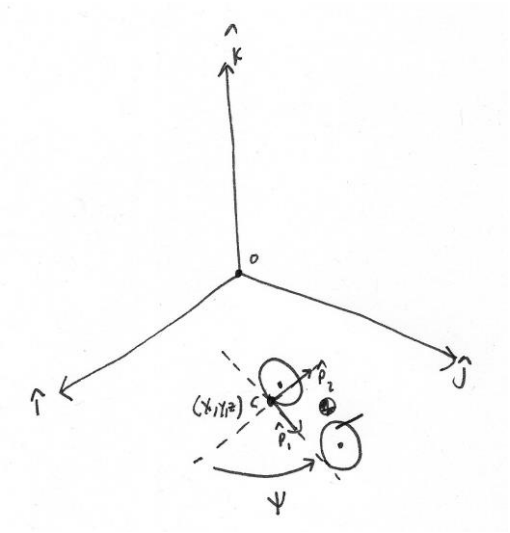


Figure 13: Bicycle Parameters and Frames

Rotation/Translation One

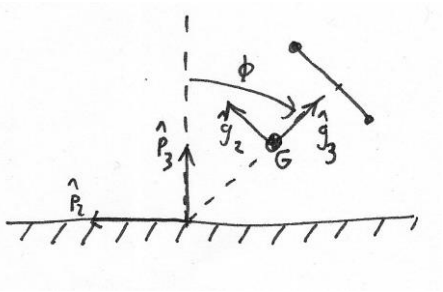
The first rotation/translation matrix describes the location of the moving frame {p1,p2,p3} located at the rear wheel contact point, with respect to the fixed { i , j , k } frame.



$$R1 = \begin{bmatrix} \cos(\psi) & -\sin(psi) & 0 & x \\ \sin(\psi) & \cos(\psi) & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation/Translation Two

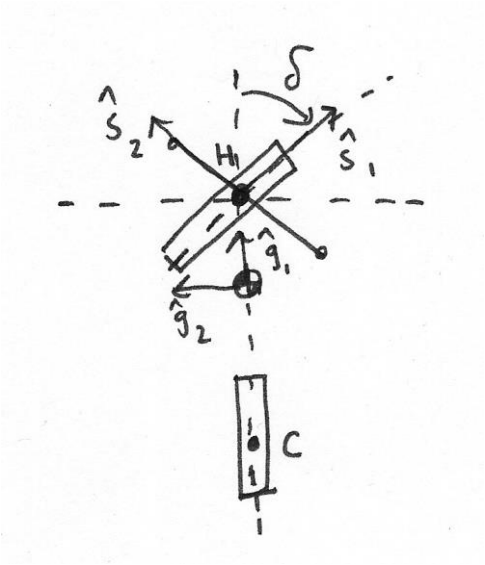
The second rotation/translation matrix describes the location and orientation of the {g1, g2 , g3} frame fixed to the center of gravity point with respect to the frame {p1,p2,p3}.



$$R2 = \begin{bmatrix} 1 & 0 & 0 & x_G \\ 0 & \cos(\phi) & -\sin(\phi) & -|y_G| * \sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) & |y_G| * \cos(\phi) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation/Translation Three

The third rotation/translation matrix describes the location and orientation of the {s1, s2 , s3} steering frame with respect to the {g1,g2,g3} frame.



$$R3 = \begin{bmatrix} \cos(\delta) & -\sin(\delta) & 0 & Handw \\ \sin(\delta) & \cos(\delta) & 0 & 0 \\ 0 & 0 & 1 & Handh \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Plotting Approach

Using these matrices described to move from frame to frame, plotting is a matter of creating points in a convenient coordinate and then multiplying those points by the coordinate transformation matrices to obtain the positions of those points expressed in the global {i,j,k} frame. Then, the plot3() command can be used.