ENPM662 Homework 1

Brendan Neal

September 23, 2023

Contents

1 Introduction

This homework has students modeling the rear-wheel drive for a bicycle as well as deriving the kinematic equations for a 3-DOF manipulator using geometrical method.

2 Problem 1.1

2.1 Derivation of the State Equations

To derive the state space model of the bicycle, I first had to draw the situation. Depicted in Figure 1 is the bicycle model.

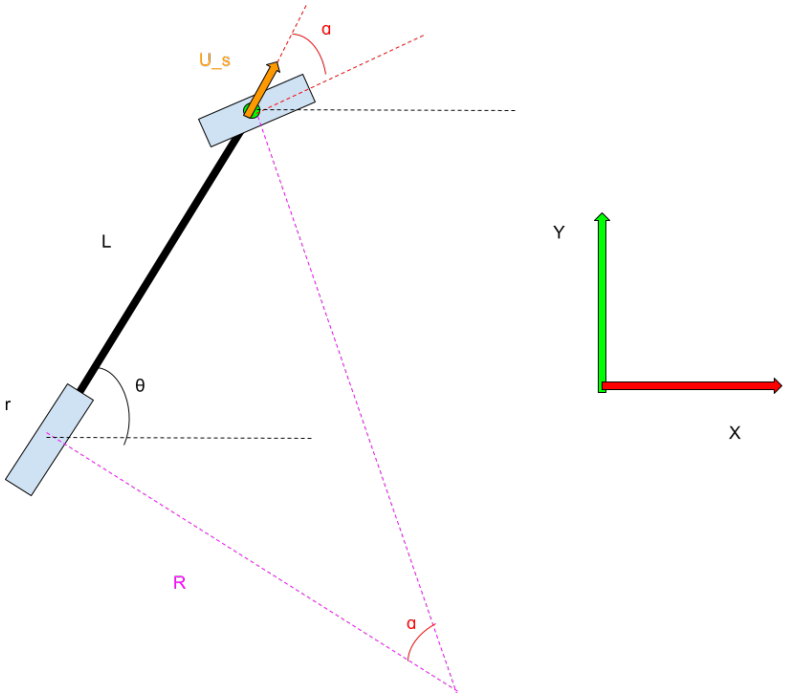

Figure 1: Bicycle Model

The table of parameters given and self-defined is given below in Table 1.

Table 1: Bicycle Model Parameters	
Parameter	Description
L	Length of Track
r	Wheel Radius
α	Steer Angle
θ	Bicycle Orientation Relative to Global Frame
R	Radius from Rear Wheel to Center of Rotation
U_s	Linear Speed of Body
ω	Angular Speed of Back Wheel

The three state equations to find are $\dot{x}$, $\dot{y}$ and $\dot{\theta}$. In the unicycle model, $\dot{x}$ and $\dot{y}$ are equal to:

$$\dot{x} = U_s * cos(\theta)$$

$$\dot{y} = U_s * sin(\theta)$$

Extending these equations to the bicycle model, which has both θ and α, the equations then change to:

$$\dot{x} = U_s * cos(\theta + \alpha)$$

$$\dot{y} = U_s * sin(\theta + \alpha)$$

because the linear velocity vector projection is affected by the steer angle, α. Please note that though the diagram shows the equation should say $\theta - \alpha$, I am using $\theta + \alpha$ with the intention of keeping the model general. This works because I am using right-hand-rule sign conventions for angles. For example, angles drawn to the right of "neutral" are negative. Thus, in this case, I am technically "adding a negative number" which is the same as subtraction.

However, U_s is not known. I need to find U_s in terms of ω. Because of no-slip and no-tilt conditions, I can use the principles of angular velocity to calculate:

$$U_s = \omega * r$$

Substituting this equation back into $\dot{x}$ and $\dot{y}$ gives:

$$\dot{x} = \omega * r * cos(\theta + \alpha)$$

$$\dot{y} = \omega * r * sin(\theta + \alpha)$$

For $\dot{\theta}$, we know the general formula is:

$$\dot{\theta} = \frac{U_s}{R}$$

I have calculated U_s previously, but R is still unknown. However, based on the diagram, L and α are known and can be used to find R. The equation is as follows:

$$R = \frac{L}{tan(\alpha)}$$

Substituting these two equations into the general form gives:

$$\frac{\omega * r}{\frac{L}{tan(\alpha)}}$$

Which simplifies to:

$$\dot{\theta} = \frac{\omega * r * tan(\alpha)}{L}$$

Finally, the last piece of information given is that $\alpha = 0.5 * sin(\pi * t)$, we can substitute this information into the three found equations for $\dot{x}$, $\dot{y}$ and $\dot{\theta}$ to yield the full velocity model:

$$\dot{x} = \omega * r * cos(\theta + 0.5 * sin(\pi * t))$$
$$\dot{y} = \omega * r * sin(\theta + 0.5 * sin(\pi * t))$$
$$\dot{\theta} = \frac{\omega * r * tan(0.5 * sin(\pi * t))}{L}$$

Knowing the velocity profile allows me to model the positional behavior of point O over time using the following equations:

$$x_{t+1} = x_t + \dot{x} * \Delta t$$
$$y_{t+1} = y_t + \dot{y} * \Delta t$$
$$\theta_{t+1} = \theta + \dot{\theta} * \Delta t$$

The application of this model in Python will be further discussed in the next section.

2.2 Results of Python Program

In order to simulate this in python, I first defined my parameters given above. Next, I created a for loop that performs numerical integration that follows the six equations defined previously. I save the state at each time interval. Finally I plot the results. The initial conditions are as follows: $X_i = 0$, $Y_i = 0$, $\theta_i = 0$, and $T = 10s$ Depicted below in Figure 2 is the X position vs. Time, Y position vs. Time, and the X vs. Y plots created by my algorithm:

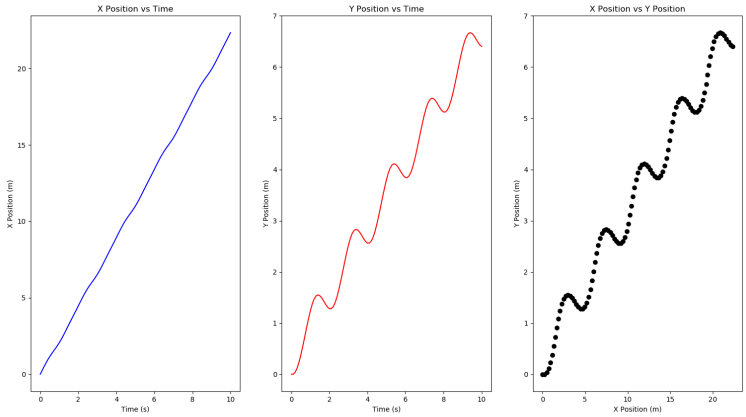

Figure 2: Bicycle State Plots

3 Problem 1.2

3.1 1.2.1: Derive the Forward Kinematic Equations

To derive the forward kinematics of the manipulator, I first had to draw the situation. Depicted in Figure 1 is the manipulator with relevant frames drawn.

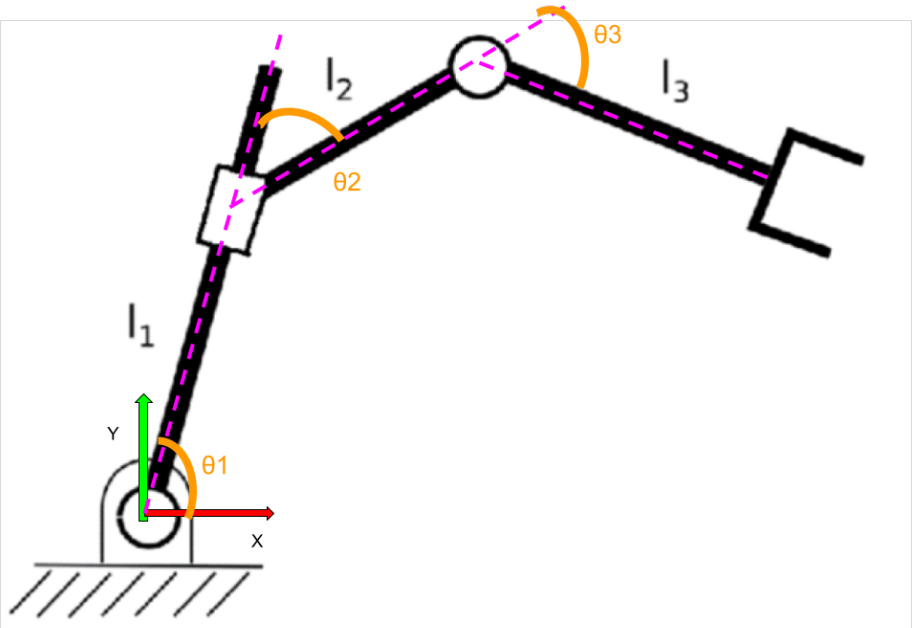

Figure 3: Manipulator Diagram

The fixed parameters in this manipulator are: L_2, L_3, and θ_2. The variable values in this manipulator are: L_1, θ_1, and θ_3. To derive the forward kinematics geometrically, I broke up the problem and worked piece by piece. Starting with L_1, the equations would be:

$$x_1 = L_1(t) * cos(\theta_1(t))$$

$$y_1 = L_1(t) * sin(\theta_1(t))$$

Moving along L_2 yields:

$$x_2 = x_1 + L_2 * cos(\theta_1(t) + \theta_2)$$

$$y_2 = y_1 + L_2 * sin(\theta_1(t) + \theta_2)$$

Moving along L_3 yields:

$$x_3 = x_2 + L_3 * cos(\theta_1(t) + \theta_2 + \theta_3(t))$$

$$y_3 = y_2 + L_3 * sin(\theta_1(t) + \theta_2 + \theta_3(t))$$

As for ϕ, the forward positional kinematics are given by:

$$\phi = \theta_1(t) + \theta_2 + \theta_3(t)$$

Combining these equations yields the full positional (x,y) forward kinematics of this robotic manipulator:

$$X = L_1(t) * cos(\theta_1(t)) + L_2 * cos(\theta_1(t) + \theta_2) + L_3 * cos(\theta_1(t) + \theta_2 + \theta_3(t))$$
$$Y = L_1(t) * sin(\theta_1(t)) + L_2 * sin(\theta_1(t) + \theta_2) + L_3 * sin(\theta_1(t) + \theta_2 + \theta_3(t))$$
$$\phi = \theta_1(t) + \theta_2 + \theta_3(t)$$

Please note that though the diagram shows the equation should include some angular subtraction, I used all addition with the intention of keeping the model general. This works because I am using right-hand-rule sign conventions for angles. For example, angles drawn to the right of "neutral" are negative. Thus, in this case, I am technically "adding a negative number" which is the same as subtraction.

In order to find the velocity forward kinematics of this robotic manipulator, I took the derivatives of the above equations using SymPy in a Python3 script and I yielded the following equations:

$$\dot{X} = -\dot{\theta_1}(t) * L_2 * sin(\theta_1(t) + \theta_2) - L_3 * (\dot{\theta_1}(t) + \dot{\theta_3}(t)) * sin(\theta_1(t) + \theta_2 + \theta_3(t)) - L_1(t) * \dot{\theta_1}(t) * sin(\theta_1(t)) + \dot{L_1}(t) * cos(\theta_1(t))$$

$$\dot{Y} = \dot{\theta_1}(t) * L_2 * cos(\theta_1(t) + \theta_2) + L_3 * (\dot{\theta_1}(t) + \dot{\theta_3}(t)) * cos(\theta_1(t) + \theta_2 + \theta_3(t)) + L_1(t) * \dot{\theta_1}(t) * cos(\theta_1(t)) + \dot{L_1}(t) * sin(\theta_1(t))$$
$$\dot{\phi} = \dot{\theta_1}(t) + \dot{\theta_3}(t)$$

3.2 1.2.1: Derive the Inverse Kinematic Equations in Matrix Format

To derive the inverse kinematic equations, I had to transform the forward velocity kinematic equations derived earlier into matrix format. To do so, I used more of SymPy's tools. First, I used the trigexpand() method on each of the velocity equations to expand the trigonometric equations only held one variable at a time. Next, I used the expand() method on each of these expressions to perform multiplication through any parenthetical expressions. Afterward, I used the collect() method on all three equations for the three joint velocity variables ($\dot{\theta_1}(t)$, $\dot{\theta_3}(t)$, $\dot{L_1}(t)$) to isolate these variables with coefficient expressions. Then, I used the coeff() method in order to return the coefficients for these variables in all three equations. The forward velocity kinematics are given by:

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\phi}(t) \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ 1 & 1 & 0 \end{bmatrix} * \begin{bmatrix} \dot{\theta_1}(t) \\ \dot{\theta_3}(t) \\ \dot{L_1}(t) \end{bmatrix}$$

Thus, the inverse kinematics are given by:

$$\begin{bmatrix} \dot{\theta_1}(t) \\ \dot{\theta_3}(t) \\ \dot{L_1}(t) \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ 1 & 1 & 0 \end{bmatrix}^{-1} * \begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\phi}(t) \end{bmatrix}$$

Where:

$$a_1 = -L_2 * sin(\theta_2) * cos(\theta_1(t)) - L_2 * sin(\theta_1(t)) * cos(\theta_2) + L_3 * sin(\theta_2) * sin(\theta_1(t)) * sin(\theta_3(t)) - L_3 * sin(\theta_2) * cos(\theta_1(t)) * cos(\theta_3(t)) - L_3 * sin(\theta_1(t)) * cos(\theta_2) * cos(\theta_3(t)) - L_3 * sin(\theta_3(t)) * cos(\theta_2) * cos(\theta_1(t)) - L_1(t) * sin(\theta_1(t))$$

$$a_2 = L_3 * sin(\theta_2) * sin(\theta_1(t)) * sin(\theta_3(t)) - L_3 * sin(\theta_2) * cos(\theta_1(t)) * cos(\theta_3(t)) - L_3 * sin(\theta_1(t)) * cos(\theta_2) * cos(\theta_3(t)) - L_3 * sin(\theta_3(t)) * cos(\theta_2) * cos(\theta_1(t))$$

$$a_3 = cos(\theta_1(t))$$

$$a_4 = -L_2 * sin(\theta_2) * sin(\theta_1(t)) + L_2 * cos(\theta_1(t)) * cos(\theta_2) - L_3 * sin(\theta_2) * sin(\theta_1(t)) * cos(\theta_3(t)) - L_3 * sin(\theta_2) * cos(\theta_1(t)) * sin(\theta_3(t)) - L_3 * sin(\theta_1(t)) * cos(\theta_2) * sin(\theta_3(t)) + L_3 * cos(\theta_3(t)) * cos(\theta_2) * cos(\theta_1(t)) + L_1(t) * cos(\theta_1(t))$$

$$a_5 = L_3 * sin(\theta_2) * sin(\theta_1(t)) * cos(\theta_3(t)) - L_3 * sin(\theta_2) * cos(\theta_1(t)) * sin(\theta_3(t)) - L_3 * sin(\theta_1(t)) * cos(\theta_2) * sin(\theta_3(t)) + L_3 * cos(\theta_3(t)) * cos(\theta_2) * cos(\theta_1(t))$$

$$a_6 = sin(\theta_1(t))$$