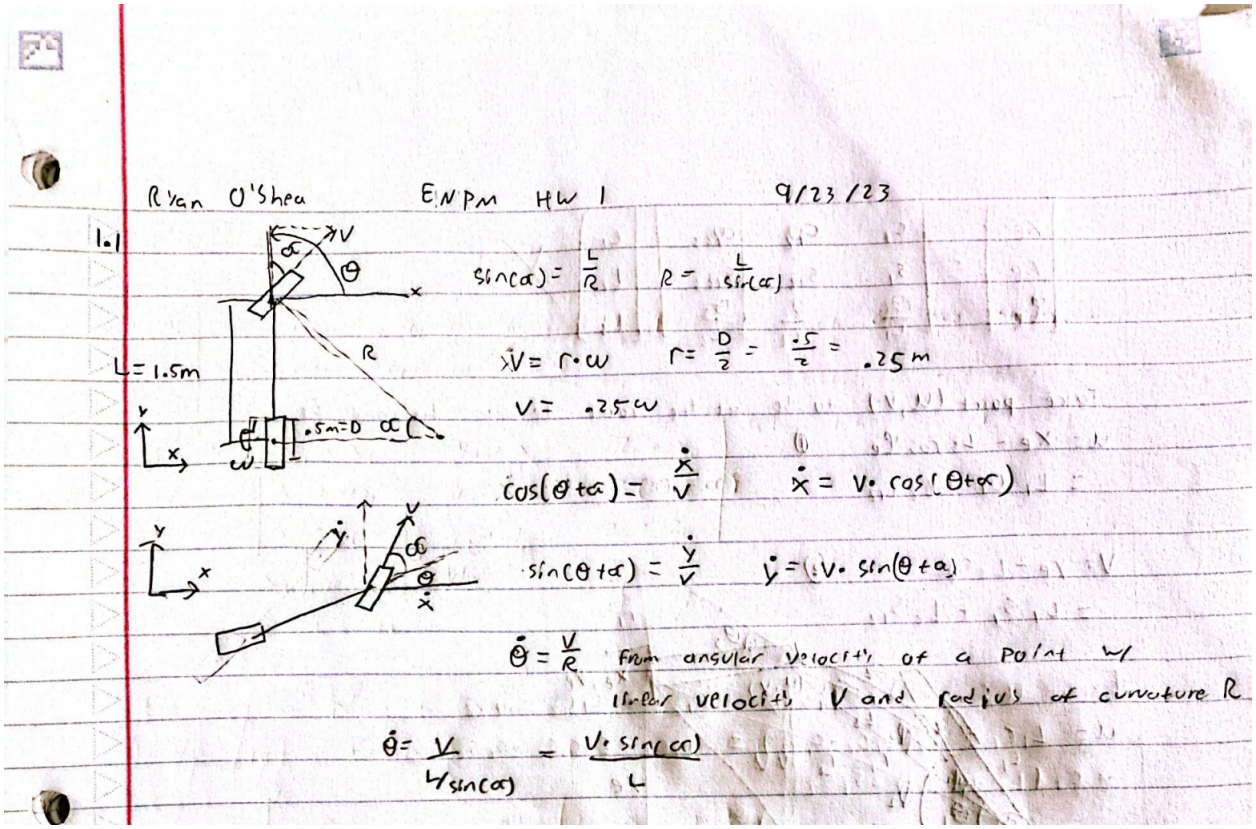
Ryan O'Shea
roshea
ENPM662

## Problem 1.1

State space derivation work:



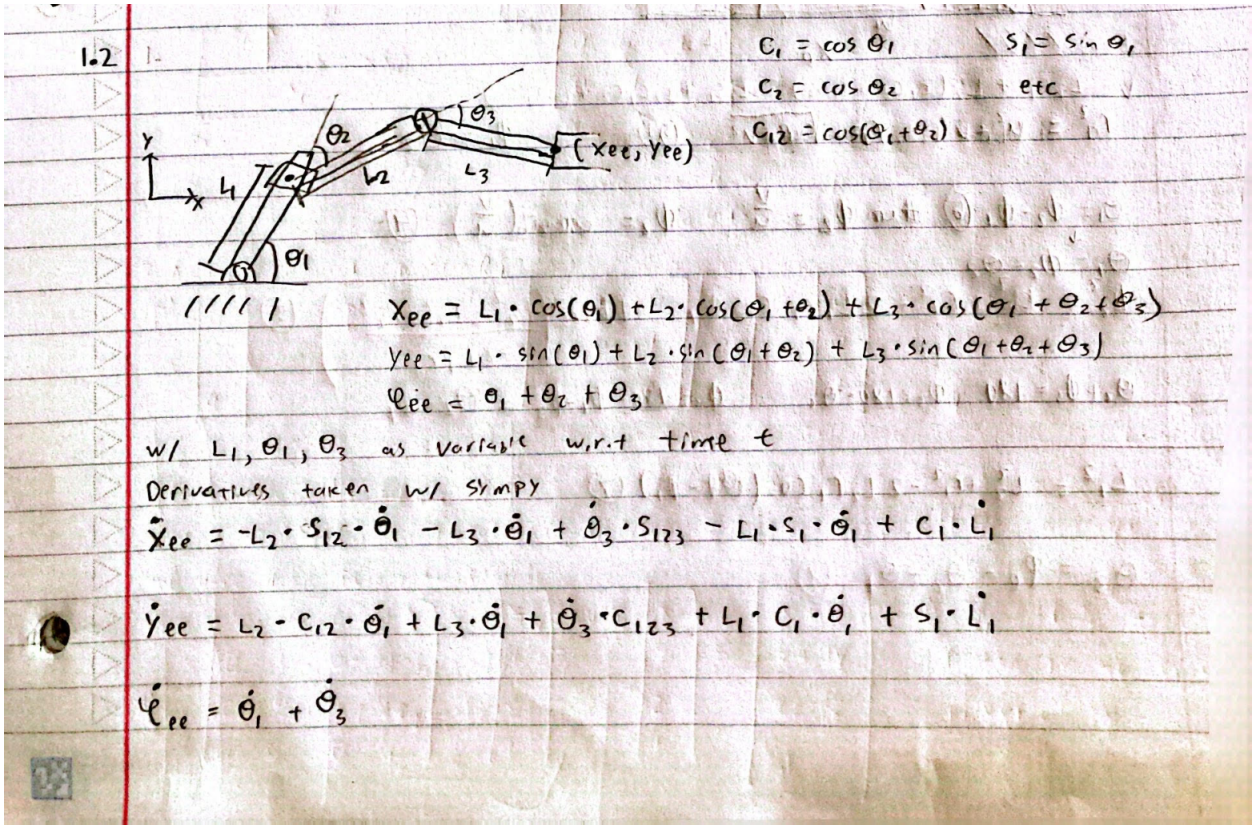

The second bicycle diagram was made to make it easier to derive the formula for x_dot. All calculations assume a positive rotation is in the counterclockwise direction.

## Problem 1.2

State space derivation:

The x and y of the end effector were calculated similar to the 2 link example we shown in class but with the addition of an extra link. For most calculations after the first I began to use the shortened naming convention for cosines and sines to save on space. The mapping scheme is in the upper right corner but for easier reference a table with examples is provided below.

| Function | Symbol |
|---|---|
| Cosine(theta_1) | C1 |
| Cosine(theta_2) | C2 |
| Cosine(theta_1 + theta_2) | C12 |
| Sine(theta_1) | S1 |
| Sine(theta_1 + theta_2 + theta_3) | S123 |




The derivatives of the forward kinematics position equations were calculated using sympy and can be found in the problem2_1.py file.

## Part 2

The matrix for the part was derived by hand using the outputs from the sympy outputs for the end effector velocity forward kinematics. The matrix was left blank with the end effector vector velocity vector on the left, the kinematics matrix in the middle, and the joint velocity vector on the right. The matrix was filled in by extracting the joint velocities from the forward kinematics velocity equations for x, y, and phi and grouping them in a

matrix. I think this could have been done in sympy but I found it much easier to do by hand. The matrix was then recreated using sympy and inverted using the inv() function. Before displaying the sympy simplify function was also used to group terms and simplify the matrix as much as possible which reduced the size significantly. The final matrix equation for calculating the joint angles from the derived matrix and the end effector velocity vector can be seen in the image below. The code use for the derivation and printouts of the matrices, end effector velocity vector, and joint angle velocity vector can be found in the problem2_2.py file.