Assignment 3
SC627

Bhavini Jeloka
18D100007

I. Introduction

In this assignment, we implement an algorithm for balancing a network of robots. This falls under the category of multi agent control and coordination as the action of one bot affects the others.

II. Balancing Robots - Algorithm Implementation

In this section, we describe in detail our control strategy to ensure that the bots are balanced and the multi agent system is in equilibrium.

A. Algorithm Initiation

We define a class that contains the position, velocities (angular and linear) and orientation of the robot via the odometer. It also receives the odometer data from its left and right neighbours. After the file is launched, we begin taking readings and this is further used in deciding the next step of motion.

B. Control Input

Since all the bots are along the x axis, we basically penalise a robot if it gets too close to its left or right neighbours. For this, we implement a simple proportional controller on the error or deviation. To that extent, we develop the following control law for robot i with left neighbour $i-1$ and right neighbour $i+1$:

$$u = k * (x_{i-1} - x_i) + k * (x_{i+1} - x_i) \tag{1}$$

This control is commanded to the velocity. We see that, in the case of equilibrium where i is in at the midpoint of its left and right neighbours ($x_{i-1} - x_i = -(x_{i+1} - x_i)$), the commanded velocity becomes zero - thus yielding the desired results. We use $k = 1$ for faster convergence. We hard code the velocity in the y direction to be zero. This velocity is then published after being converted into the format used by ROS (helper code already has that function).

C. Termination

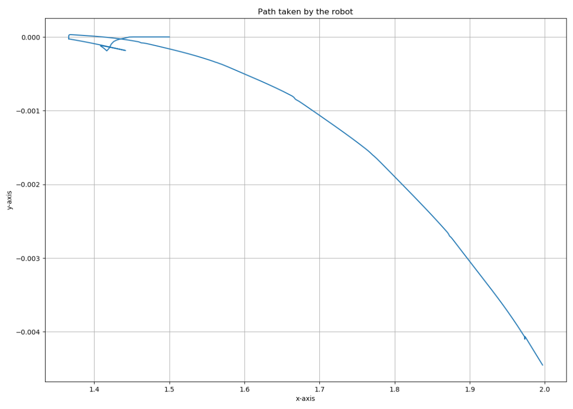
We terminate the algorithm once the velocities are close to zero and the robots begin to stabilize. To be precise, our exact stopping condition is when the absolute velocities (bot in consideration, along with the left and right neighbours) are within a $\delta = 0.001$ ball around 0.

III. Numerical Results

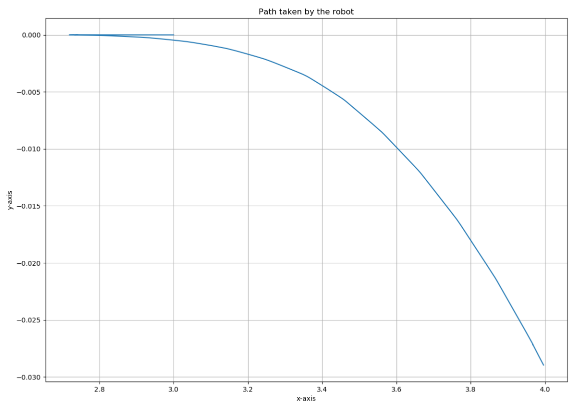
We test our algorithm in a 2D environment with 8 robots in a straight line. Some key features of the environment are given below along with plots and visualisations of the results obtained.

1) Position of Robot 1 (Stationary): [0,0]
2) Position of Robot 8 (Stationary): [14,0]
3) Maximum Velocity of the Robots: 15m/s

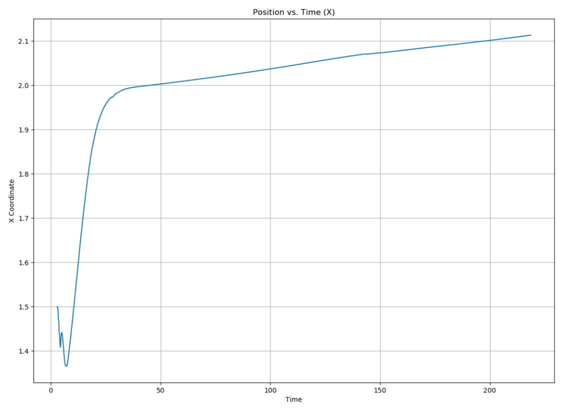
We see that the robots are equally spaced and are 2 metres apart (with a maximum error of 0.1m). Thus, our strategy has successfully worked within the desired error bounds.



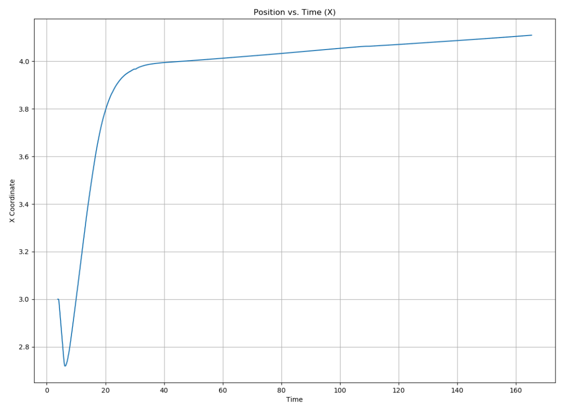
(a) Robot 2: Path



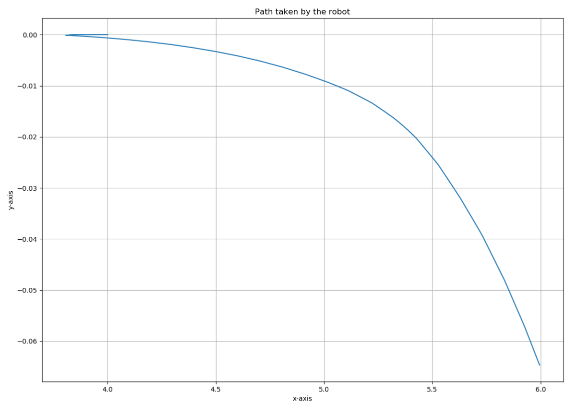
(a) Robot 3: Path



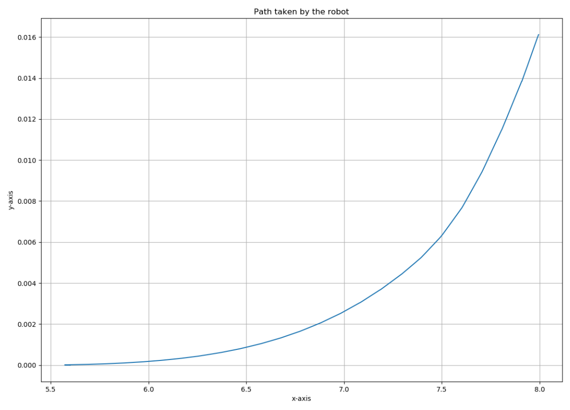
(b) Robot 2: x-Coordinate



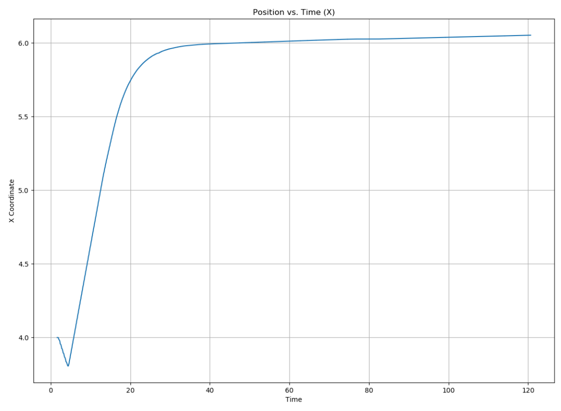
(b) Robot 3: x-Coordinate



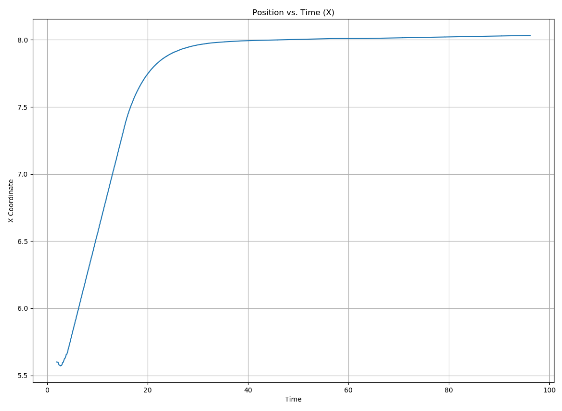
(a) Robot 4: Path



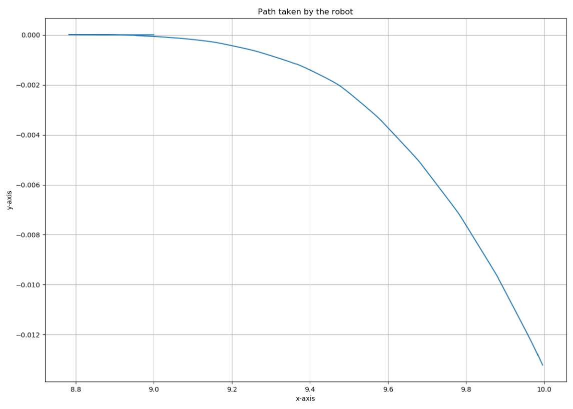
(a) Robot 5: Path



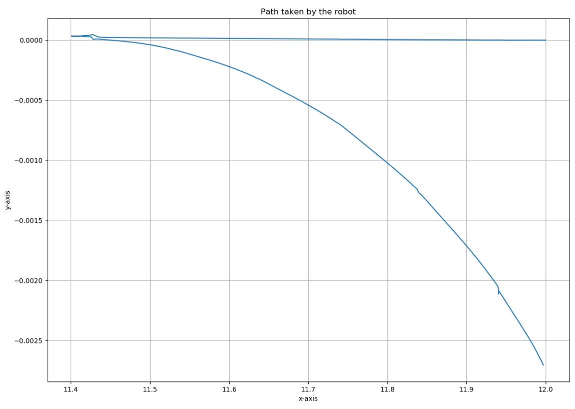
(b) Robot 4: x-Coordinate



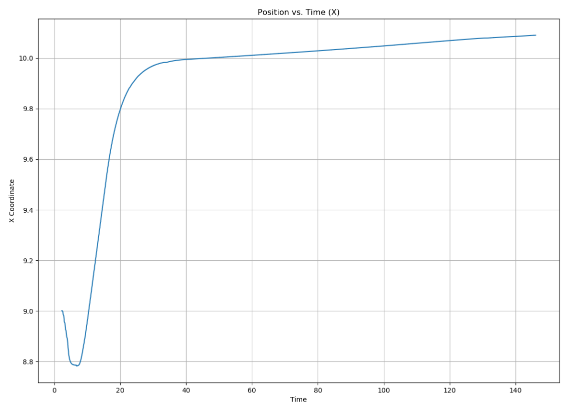
(b) Robot 5: x-Coordinate



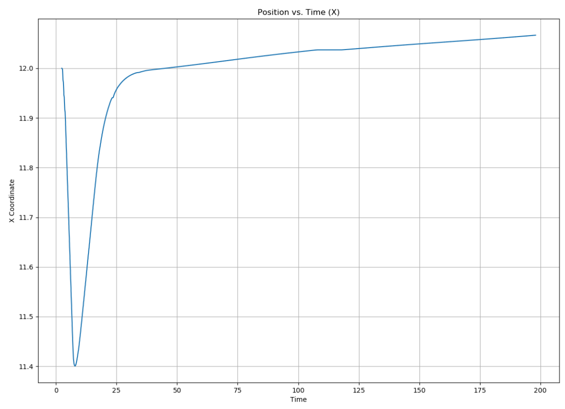
(a) Robot 6: Path



(a) Robot 7: Path



(b) Robot 6: x-Coordinate



(b) Robot 7: x-Coordinate