Assignment - 5

Anurag Sahu (2018121004)
Apoorva Srivastava (20191702014)

Question 1

Theory:

Trajectory Planning Overview : The trajectory planning problem consists in finding a relationship between two elements belonging to different domains: time and space. Accordingly, the trajectory is usually expressed as a parametric function of the time which provides at each instant the corresponding desired position.

Polynomial Trajectories : In the simplest case, a motion is defined by assigning the initial and final time instant t0 and t1, and conditions on position, velocity and acceleration at t0 and t1. From a mathematical point of view, the problem is then to find a function

$$q = q(t) \quad t \in [t0,\ t1]$$

such that the given conditions are satisfied. This problem can be easily solved by considering a polynomial function

$$q(t) = a + bt + ct^2 + dt^3 + \ldots + mt^m$$

where the m+ 1 coefficients ai are determined so that the initial and final constraints are satisfied. The degree n of the polynomial depends on the number of conditions to be satisfied and on the desired "smoothness" of the resulting motion. Since the number of boundary conditions is usually even, the degree n of the polynomial function is odd, i.e. three, five, seven, and so on.

In general, besides initial and final conditions on the trajectory, other conditions could be specified concerning its time derivatives (velocity, acceleration, jerk, ...) at generic instants

tj ∈ [t0, t1]. In other words, one could be in-terested in determining a polynomial function q(t)whose k-th time derivative assumes a specific value q(k)(tj) at a given instant tj. Mathematically, these conditions can be specified as a matrix form

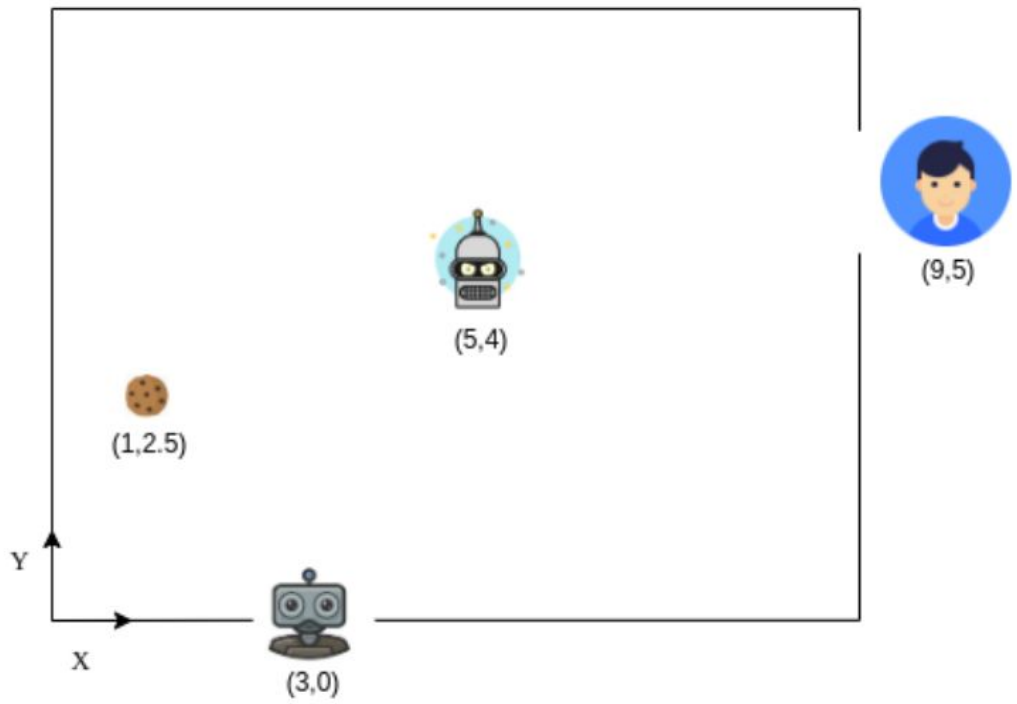
$$\mathbf{M\ a = b}$$

Where M is a known (n+1)×(n+ 1) matrix, b collects the given (n+1)conditions to be satisfied, and $a = [a0, a1, ..., an]^T$ is the vector of the unknown parameters to be computed. In principle this equation can be solved simply as

$$a = M^{-1}b$$

Instead of directly taking the Inverse of M matrix we can also take the pseudo inverse of the matrix M.

Given :



Robot's Initial Position : (3,0) at time t = 0

Robot's Initial Velocity : (0,0) at time t = 0
Robot should pass the point : (1,2.5) at some time instant ti assumed to be ti = 2
Robot's velocity as passing point : (0,0) this is assumed at some time instant ti assumed to be ti = 2
Robot's Final position : (9,5) at time t = 5
Robot's Velocity at final point : (0,0) at time t = 5

Analysis:

Given the above constraints we obtained following set of matrices:

$$\mathbf{M\ a = b_x}$$

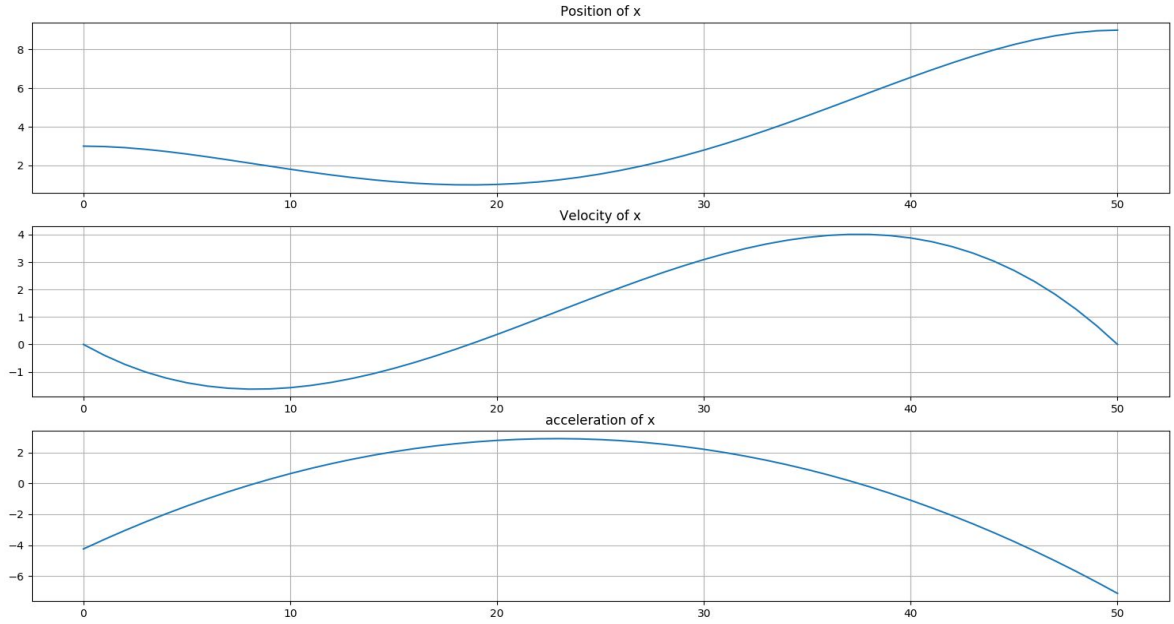
$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1.923 & 3.697 & 7.11 & 13.67 \\ 1 & 5 & 25 & 125 & 625 \\ 0 & 1 & 10 & 75 & 500 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} 3 \\ 0 \\ 1 \\ 9 \\ 0 \end{bmatrix}$$

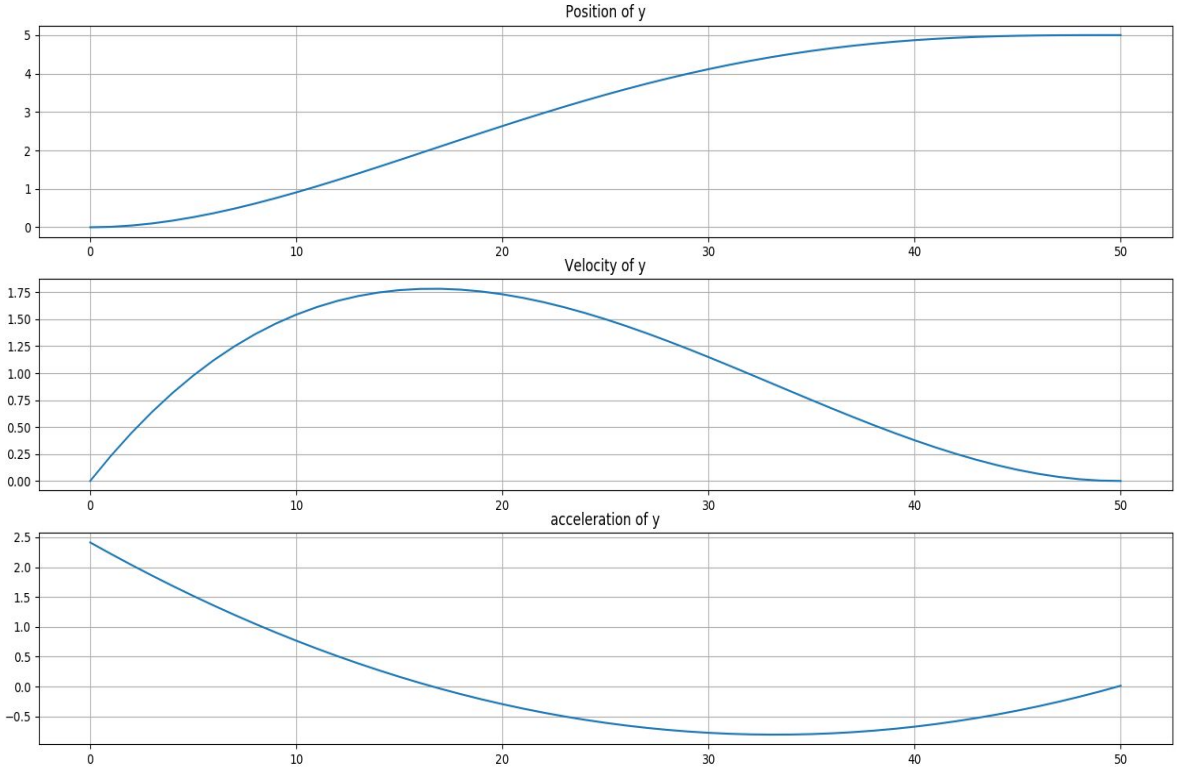
$$\mathbf{M\ a = b_y}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1.923 & 3.697 & 7.11 & 13.67 \\ 1 & 5 & 25 & 125 & 625 \\ 0 & 1 & 10 & 75 & 500 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 2 \\ 5 \\ 5 \end{bmatrix}$$

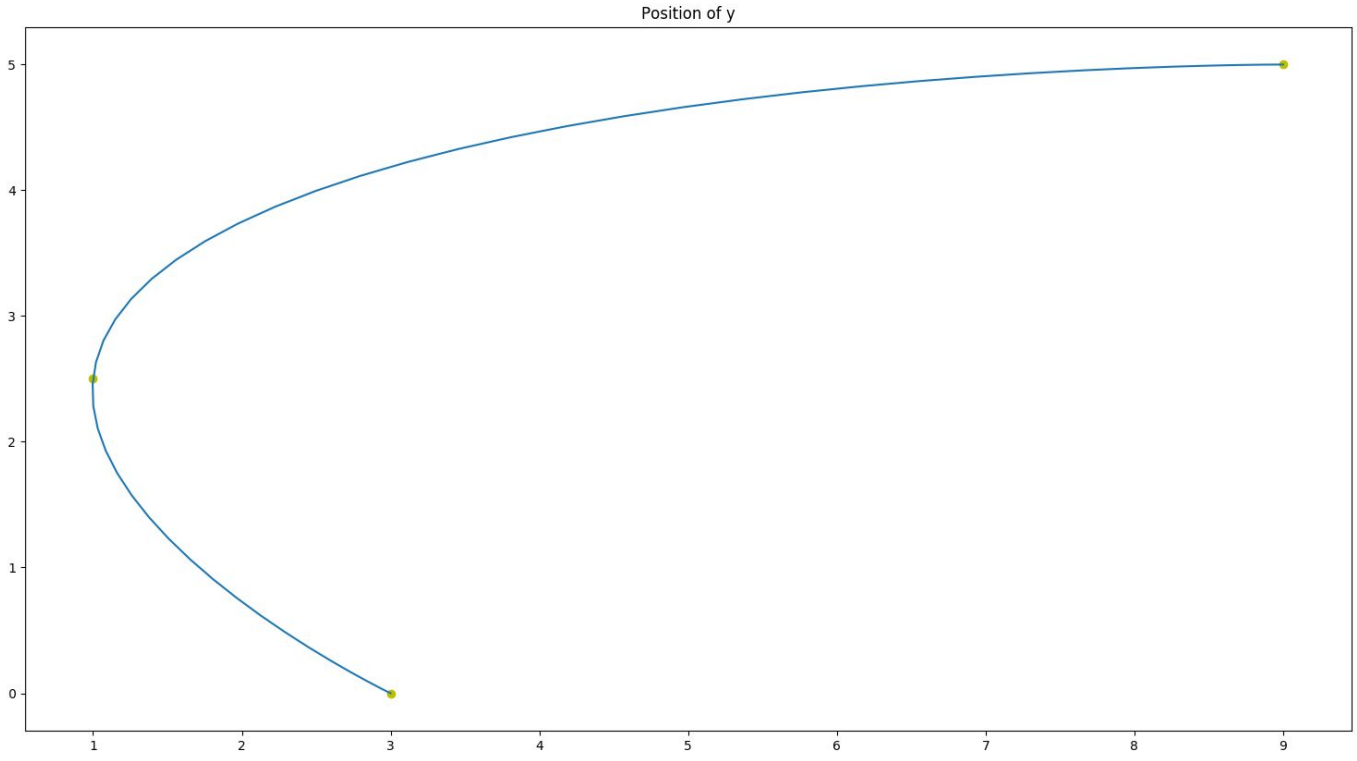
Code Link: Link to Code on Google drive, Link to code on Github.

Code Output : This is the code output of the First part of the Question which gives the Position, Velocity and accelerations of x and y components of robots motio





This is the final Trajectory Followed by the Robot.



Question 2:

Theory : In this part of the Question we use the same theory as we used for the previous question just that we change the polynomial, Where the polynomial is:

$$f(t) = \sum_{i=0}^{n} {}^{n}C_i \tau^i (1-\tau)^{n-i} p_i$$

Where

$$\tau = \left(\frac{t-t_o}{t_f-t_o}\right).$$

Analysis:
Given the above constraints we obtained following set of matrices:

$$\mathbf{M\ a = b_x}$$

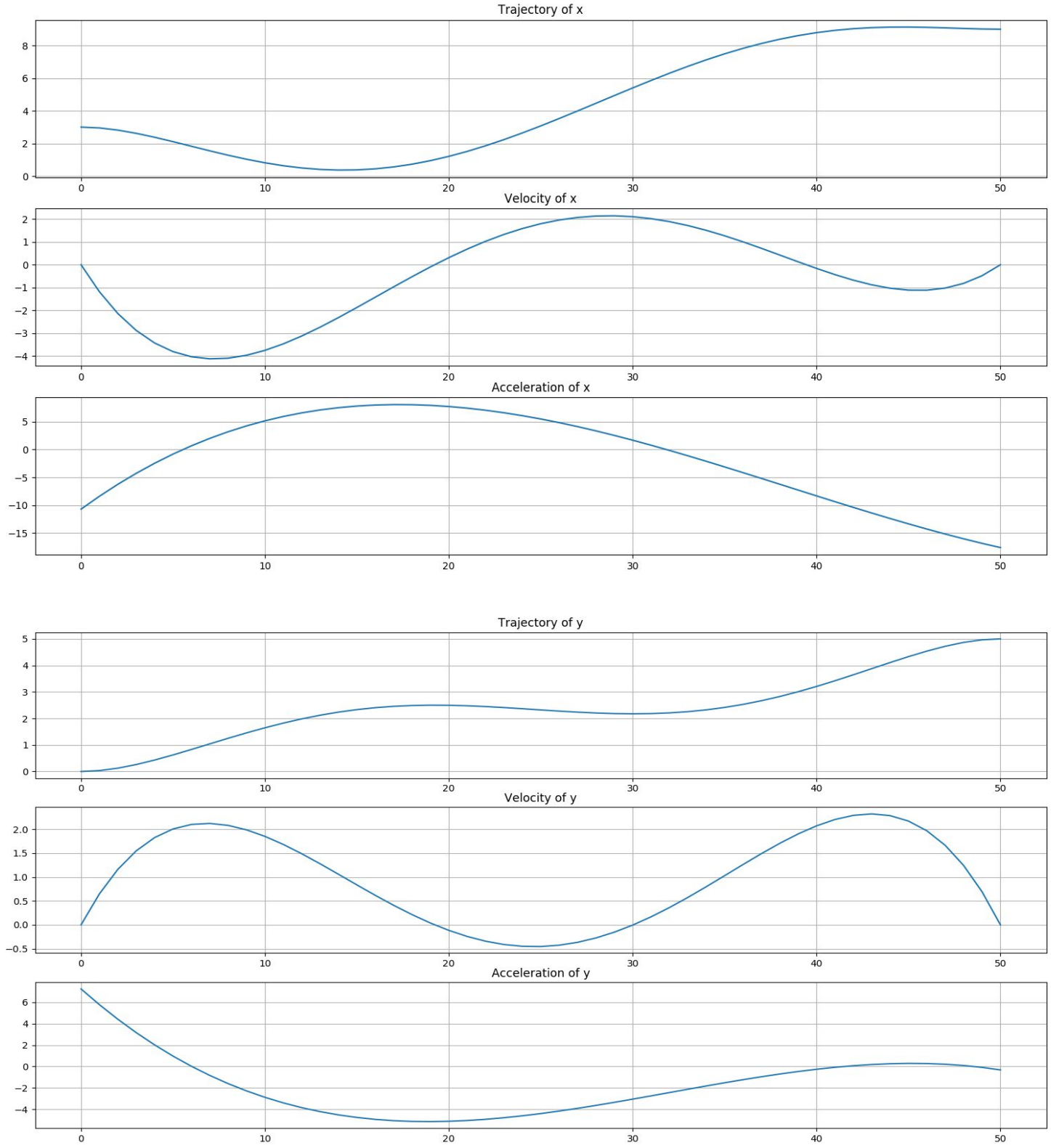
$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 \\ 0.08 & 0.27 & 0.34 & 0.21 & 0.06 & 0.0081 \\ -0.97 & -0.21 & 0.02 & 0.19 & 0.11 & 0.02 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} 3 \\ 0 \\ 1 \\ 0 \\ 9 \\ 0 \end{bmatrix}$$

$$\mathbf{M\ a = b_y}$$

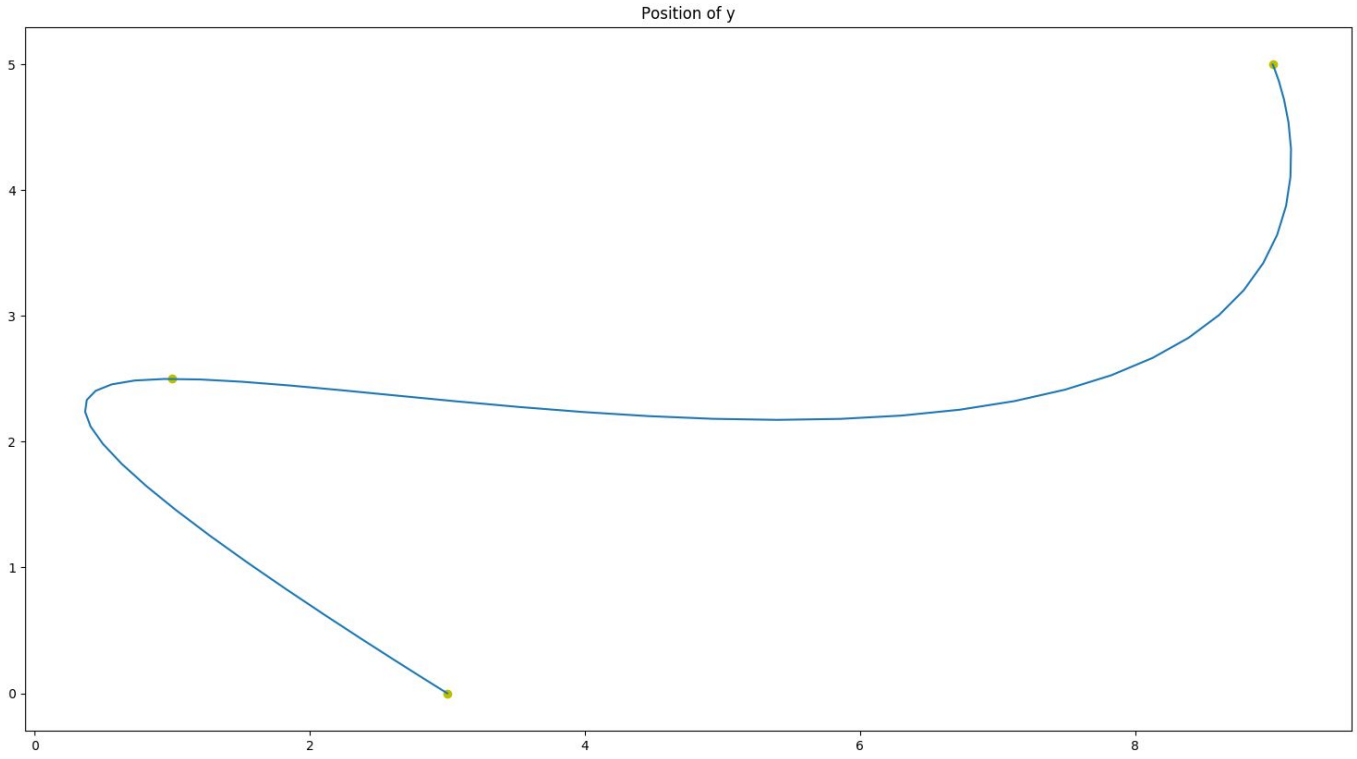
$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 \\ 0.08 & 0.27 & 0.34 & 0.21 & 0.06 & 0.0081 \\ -0.97 & -0.21 & 0.02 & 0.19 & 0.11 & 0.02 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 2 \\ 5 \\ 0 \\ 5 \end{bmatrix}$$

Code Link: Link to Code on Google drive, Link to code on Github.

Code Output:
This is the code output of the First part of the Question which gives the Position, Velocity and accelerations of x and y components of robots motion.


Trajectory of x
Velocity of x
Acceleration of x
Trajectory of y
Velocity of y
Acceleration of y

This is the final Trajectory Followed by the Robot.



<u>Question 3:</u>

There can be multiple ways in which the trajectory can be planned to avoid the evil robot appearing at (5,4).Some of them are mentioned below:

1)Cell Decomposition Method

2)Artificial Potential Method

3)Cubic Trajectory to plan the path using known points to reach the goal

1)<u>Cell Decomposition Method</u>

According to the cell decomposition methods, the free space of the robot is subdivided into several regions, called cells, in such a way that a path between any two configurations lying in the same cell is straightforward to generate.

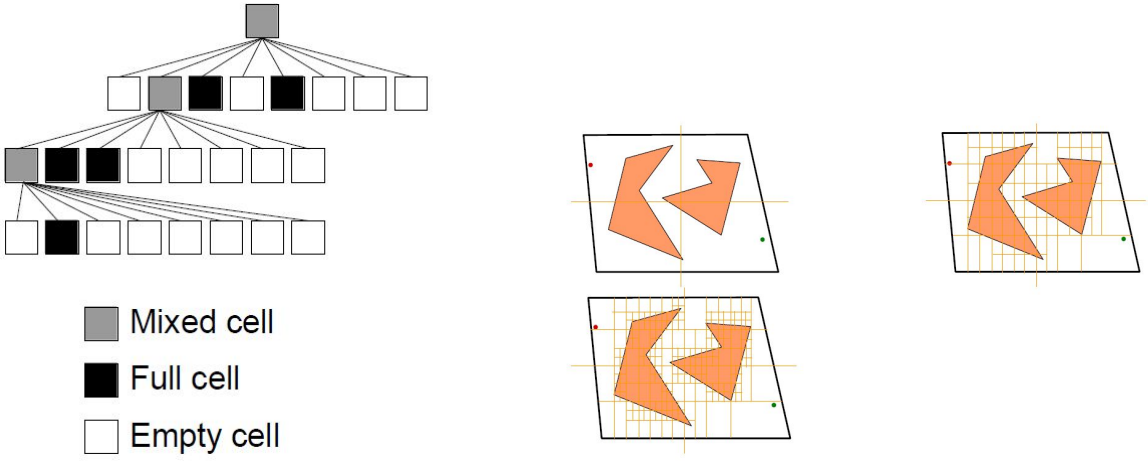
Algorithm:

- Whole space (assumed 2-dimensional) is divided into four cells.
- the algorithm checks if each cell is completely empty, completely full or

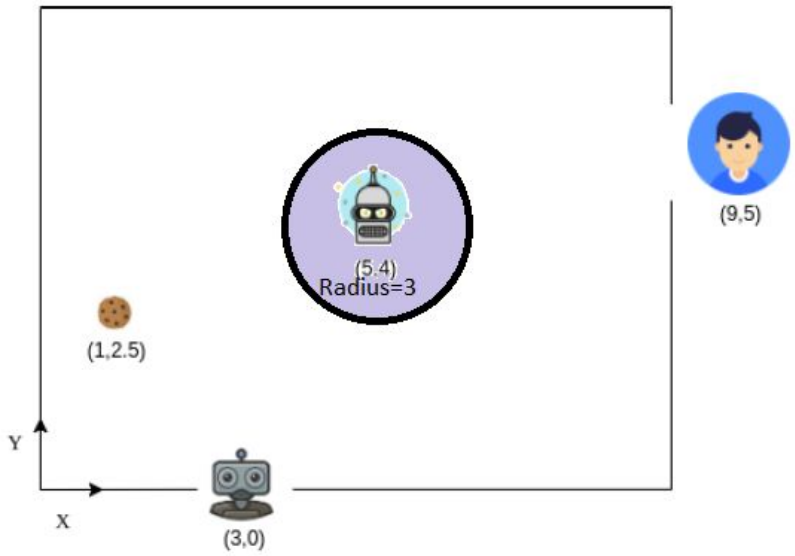
mixed (such words obviously refer to the occupancy by the obstacles)

- Each mixed cell is in turn divided into four subcells.
- Same process is repeated again and again for every mixed cell until the space is divided among the empty and full cells only.
- The path is then planned towards the goal via the adjacent empty cells.

The Algorithm can be visualized in the figure below-



This algorithm can be applied to the given scenario if we assume a circle of 3 units(i.e.radius of evil robot+radius of our robot) around the Evil Robot and decompose the space into cells by above method, we can easily plan the trajectory to reach the goal.



2)Artificial Potential Method

The basic idea is to consider the robot in the configuration space as a moving point subject to a potential field generated by the goal configuration and the obstacles in the whole space: namely, the target configuration produces an attractive potential, while the obstacles generate a repulsive potential. The sum of these two contributions is the total potential, which can be seen as an artificial force applied to the robot, aimed at approaching the goal and avoiding the obstacles. Thus, given any configuration during the robot motion, the next configuration can be determined by the direction of the artificial force to which the robot is subjected. This normally represents the most promising direction of motion in terms of free path.

Here we can plan the path dynamically by allocating repulsive potential to the Obstacle and attractive potential to the Goal and Robot will itself be dragged towards the goal under the influence of artificial force avoiding any contact with the Evil Robot.

3)Cubic Trajectory to plan the path using known points to reach the goal

If we are provided with the position and velocity at 2 points,we have in all 4 constraints so we can take a cubic polynomial to plan the trajectory between those 2 points as

$$q(t) = a_0 + a_1(t-t_0) + a_2(t-t_0)^2 + a_3(t-t_0)^3 \quad ; t_0 <= t <= t_1$$

Now, this Cubic Trajectory can be exploited to form a trajectory between n points and dividing the whole path into n-1 sections then finding Cubic Trajectory for each section.

In defining a trajectory through a set of points q0, . . . , qn, not always the velocities in the intermediate points are specified. In these cases, suitable values for the intermediate velocities may be determined with heuristic rules such as

$$v = \begin{cases} v0; & \text{(assigned)} \\ 0; & \text{if sign}(dk) = \text{sign}(dk+1) \\ .5\,(dk + dk+1); & \text{if sign}(dk) = \text{sign}(dk+1) \\ Vn; & \text{(assigned)} \end{cases}$$

where $d_k = (q_k - q_{k-1})/(t_k - t_{k-1})$ is the slope of the line segment between the instants t_{k-1} and t_k

sign(·) is the sign function.

Here we can take multiple points in the space provided and predict the velocity of the robot at that point using the rule given above. Then by finding trajectory between each of the assumed points we can reach the goal avoiding the Evil Robot.