AA 274A: Principles of Robot Autonomy I
Problem Set HW1

SUID:06734162
Name: Karthik Pythireddi
HW Group: Alberto Guiggiani, Avrum Noor, Josie Oetjen

Problem 1: Trajectory Generation via Differential Flatness

As mentioned in the problem 1, let's consider the polynomial basis functions of the below form to calculate the associated x_i,y_i coefficients. where ψ_i for i=1,....,n are the basis functions.

$$Given \quad x(t) = \sum_{i=1}^{n} x_i \psi_i(t), \quad y(t) = \sum_{i=1}^{n} y_i \psi_i(t) \tag{1}$$

The basis functions were defined as $\psi_1(t) = 1$, $\psi_2(t) = t$, $\psi_3(t) = t^2$, $\psi_4(t) = t^3$. Consider the equation (1) and try to substitute the given x(t), y(t) and $\psi(t)$ values for the corresponding i values.

$$x(t) = x_1\psi_1 t + x_2\psi_2(t) + x_3\psi_3(t) + x_4\psi_4(t) \tag{2}$$

$$y(t) = y_1\psi_1 t + y_2\psi_2(t) + y_3\psi_3(t) + y_4\psi_4(t) \tag{3}$$

Substitute the corresponding ψ values in the above equations (2) and (3) to arrive at the below set of linear equations.

$$x(t) = x_1 + x_2 t + x_3 t^2 + x_4 t^3$$

$$y(t) = y_1 + y_2 t + y_3 t^2 + y_4 t^3$$

If we substitute the given set of initial conditions x(0)=0 and y(0)=0 , in the x(t) and the y(t), then we can arrive at the x_1 and y_1 values.

$$x(0) = x_1 + x_2(0) + x_3(0)^2 + x_4(0)^3$$

$$x_1 = 0 \tag{4}$$

$$y(0) = y_1 + y_2(0) + y_3(0)^2 + y_4(0)^3$$

$$y_1 = 0 \tag{5}$$

Plugging in the new set of x_1 y_1 values in the x(t) and y(t) equations, we have them as below:

$$x(t) = x_2 t + x_3 t^2 + x_4 t^3$$

$$y(t) = y_2 t + y_3 t^2 + y_4 t^3$$

It was given that $\dot{x}(t)$= $V(t)\cos(\theta(t)$ and $\dot{y}(t)$= $V(t)\sin(\theta(t)$ and $V(0) = 0.5, \theta(0) = -\pi/2, x(t_f) = 5, y(t_f) = 5, V(t_f) = 0.5, \theta(t_f) = -\pi/2$. Differentiating the $x(t)$ and $y(t)$ with respect to t, will give us the below equations.

$$\dot{x}(t) = x_2 + 2tx_3 + 3t^2 x_4 \tag{6}$$

$$\dot{y}(t) = y_2 + 2ty_3 + 3t^2 y_4 \tag{7}$$

Plugging in the values for $\dot{x}(t)$ and $\dot{y}(t)$ along with the corresponding initial conditions in the equations (4) and (5) will provide us with the below equations.

$$\dot{x}(0) => x_2 = 0.5\cos(\theta(-\pi/2)) = 0 \tag{8}$$

$$\dot{y}(0) => y_2 = 0.5\sin(\theta(-\pi/2)) = -0.5 \tag{9}$$

Similarly plugging in the corresponding t_f=15 values will also give us the values of $\dot{x}(t)$ and $\dot{y}(t)$ at final time.

$$\dot{x}(15) = 0.5\cos(\theta(-\pi/2)) = 0$$

$$\dot{y}(15) = 0.5\sin(\theta(-\pi/2)) = -0.5$$

From the equations (4) and (5), (8) and (9), we have the corresponding coefficients as:

$$x_1 = 0, y_1 = 0, x_2 = 0, y_2 = -0.5$$

Also plugging in the $t_f = 15$ in $x(t)$, $y(t)$ $\dot{x}(t)$ and $\dot{y}(t)$ provides us with the below system of equations.

$$x(t_f = 15) => 5 = x_1 + 15x_2 + 225x_3 + 3375x_4$$

$$y(t_f = 15) => 5 = y_1 + 15y_2 + 225y_3 + 3375y_4$$

$$\dot{x}(t_f = 15) => 0 = x_2 + 30x_3 + 675x_4$$

$$\dot{y}(t_f = 15) => -0.5 = y_2 + 30y_3 + 675y_4$$

(ii) From the Problem 1, it was given that the matrix J is invertible (i.e) $det(J) \neq 0$. Also given the $det(J) = V$, if we set $V(t_f) = 0$ then it will make the values in the second column ($V(t)\cos(\theta(t)$ and $V(t)\sin(\theta(t))$ to be zero which thereby makes the $\dot{x}(t)$ and $\dot{y}(t)$ to be zero, by which we can't find the associated control inputs and orientation of the robot. That also makes the determinant of the matrix J to be zero, which is contrary to the given condition that matrix J is invertible.

(iii) Implemented the `compute_arc_length`, `rescale_V`, `compute_tau` and `rescale_om` in the `P1_differential_flatness.py` file to compute the state-trajectory.

Figure 1: Plots for the functions implemented as part of the differential flatness system

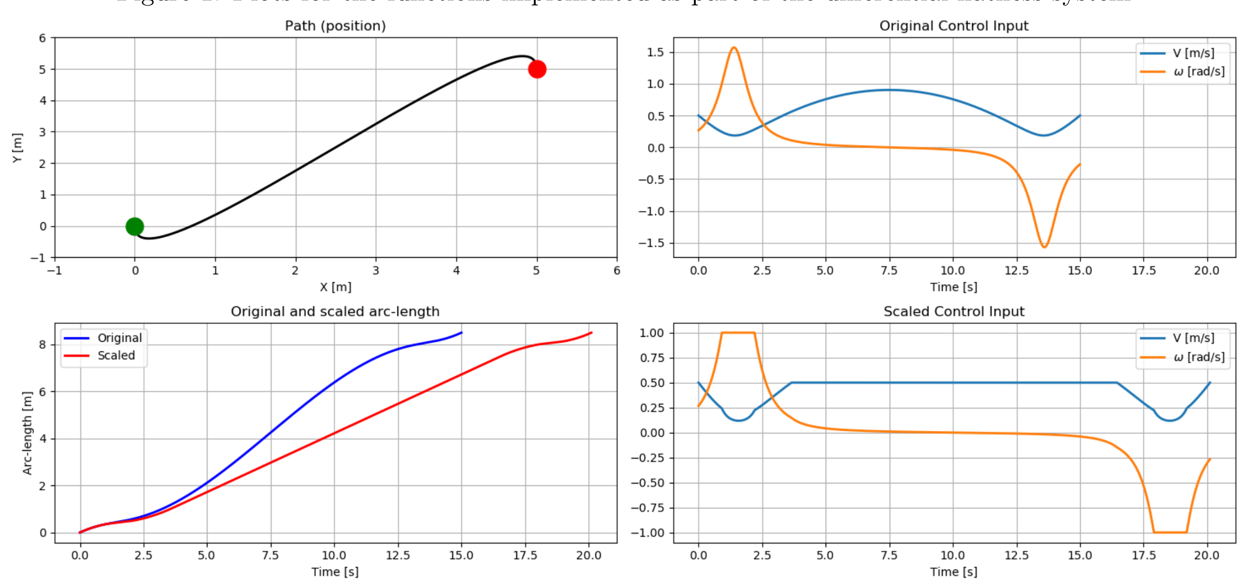
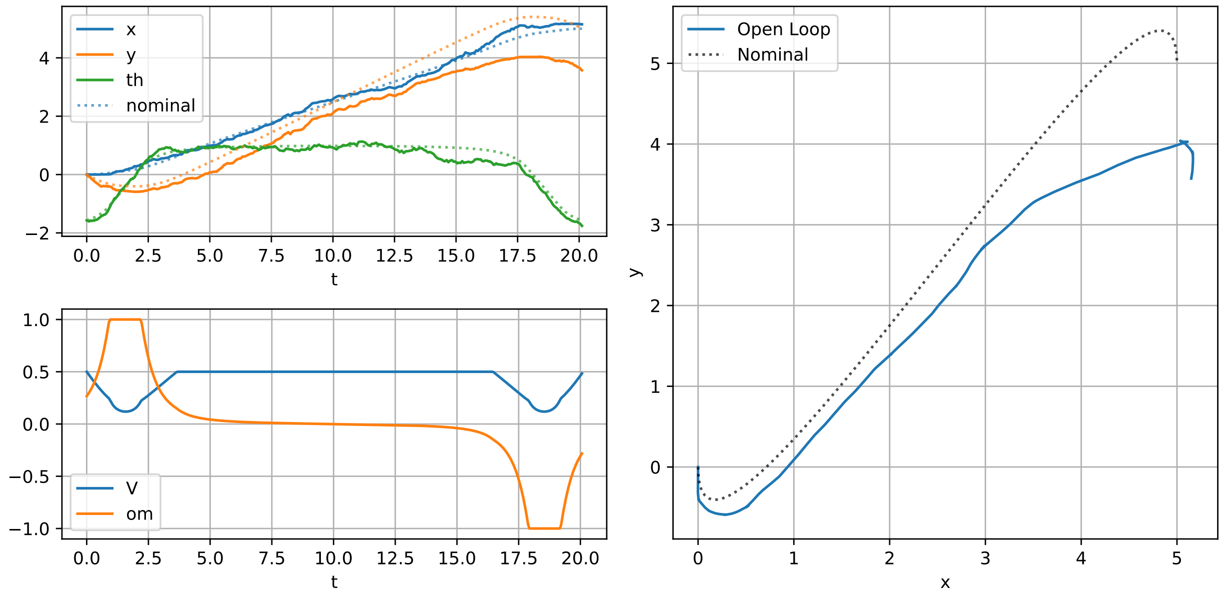


Figure 2: Plots for sim_traj_openloop.pdf with the initial conditions mentioned in the Problem 1

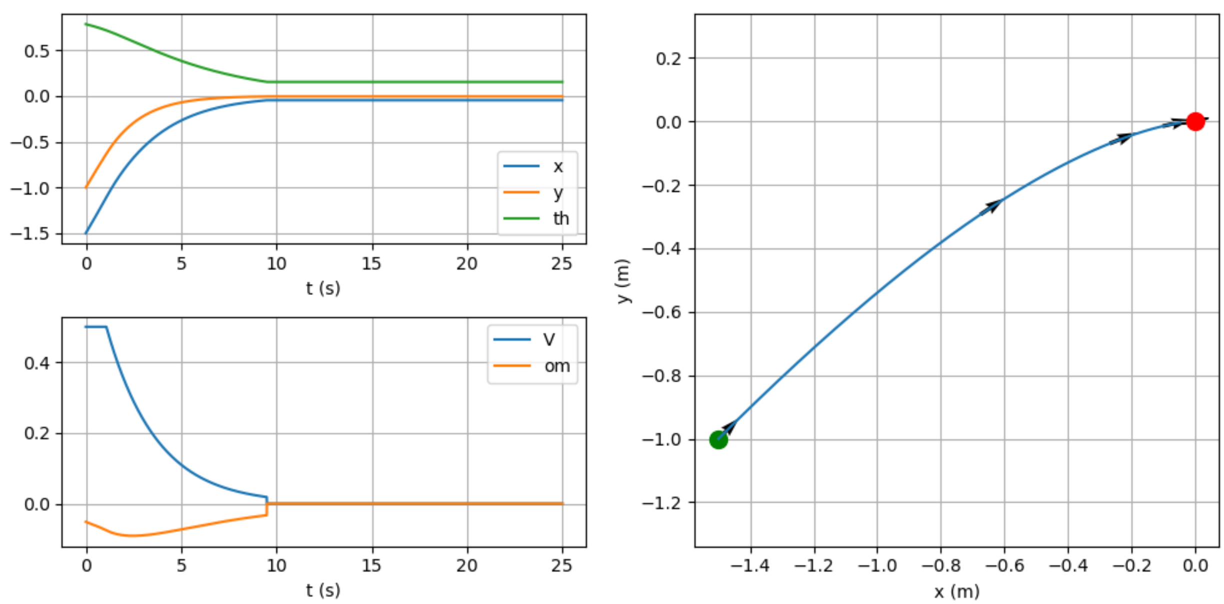


Problem 2: Pose Stabilization

(i) Implemented the compute_control in the Posecontroller class in the P2_pose_stabilization.py

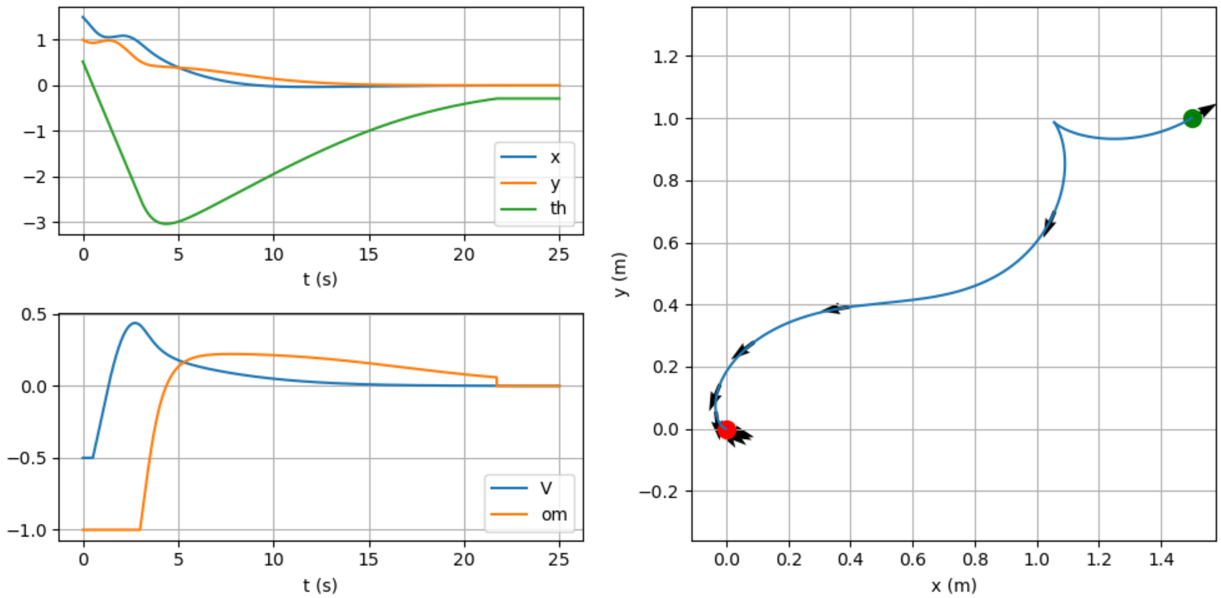
For the Forward case, the robot starts from x,y coordinates along with the orientation provided as part of the initial conditions and starts to drive forward towards the goal point.

Figure 3: Plots for the sim_parking_forward as part of the Pose Stabilization Problem



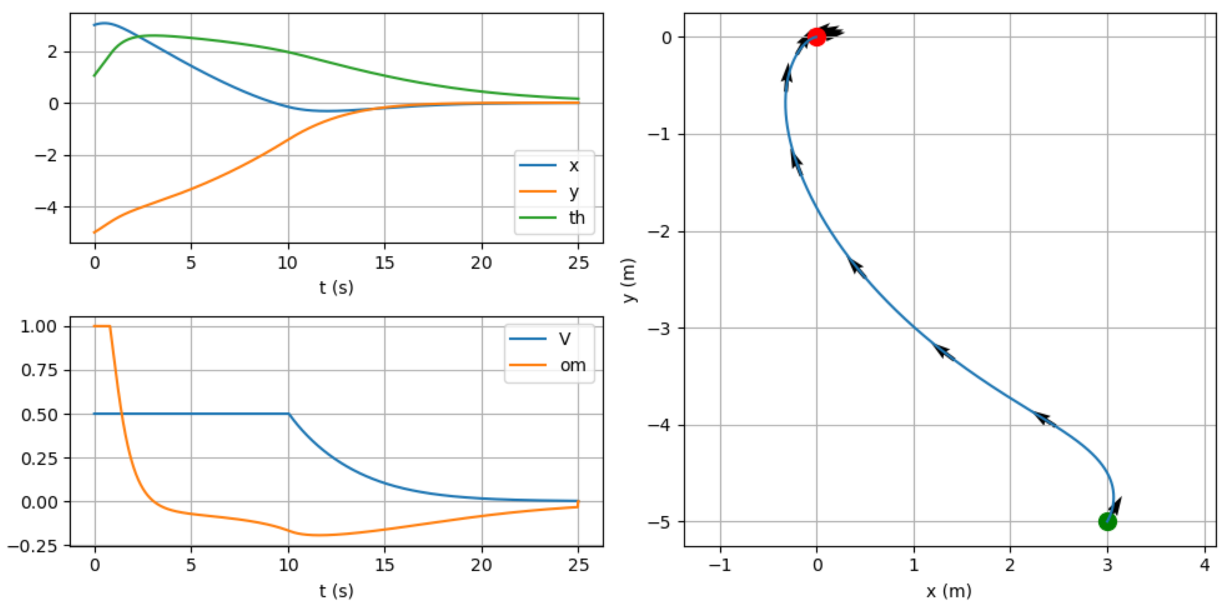
For the Reverse case, the robot starts from x,y coordinates along with the orientation provided as part of the initial conditions, re-orients itself and starts to drive towards the goal point which is behind the initial location of the robot.

Figure 4: Plots for the sim_parking_reverse as part of the Pose Stabilization Problem



For the Parallel case, the robot starts from x,y coordinates along with the orientation provided and as part of the initial conditions, re-orients itself and starts to drive towards the goal point side ways as the goal point is located on the lower left of the initial location of the robot.

Figure 5: Plots for the sim_parking_parallel as part of the Pose Stabilization Problem



Problem 3: Trajectory Tracking

As mentioned in the problem 3, we will use the differential flatness approach and will consider the $\ddot{x}(t)$ and $\ddot{y}(t)$ along with the matrix J and U.

From the problem 1, we have the matrix J as : $\begin{bmatrix} \cos(\theta(t)) & \text{-V(t)}\sin(\theta(t)) \\ \sin(\theta(t)) & \text{V(t)}\cos(\theta(t)) \end{bmatrix}$, and the system of equations being represented in the below form:

$$\begin{bmatrix} \ddot{x}(t) \\ \ddot{y}(t) \end{bmatrix} = \begin{bmatrix} \cos(\theta(t)) & \text{-V(t)}\sin(\theta(t)) \\ \sin(\theta(t)) & \text{V(t)}\cos(\theta(t)) \end{bmatrix} * \begin{bmatrix} a \\ w \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

To solve for a and w, we need to inverse the matrix J and multiply with the matrix U, and $\dot{a} = V$, $w = \dot{\theta}$

$$\begin{bmatrix} a \\ w \end{bmatrix} = J^{-1} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

If we substitute the the given u_1 and u_2 in the above matrix form, then we have the system of equations as below. Solving these will provide us with the a and w, from which we can compute the true control inputs (V, w).

$$\begin{bmatrix} a \\ w \end{bmatrix} = J^{-1} \begin{bmatrix} \ddot{x}_d + k_{px} * (x_d - x) + k_{dx}(\dot{x}_d - \dot{x}) \\ \ddot{y}_d + k_{py} * (y_d - y) + k_{dy}(\dot{y}_d - \dot{y}) \end{bmatrix}$$

By solving the V and w values using the above system of equations form, implemented the trajectory tracking problem by checking for the `V_PREV_THRES` and other constraints mentioned in the code.

Figure 6: Plots for `sim_traj_closedloop.pdf` with the initial and final positions as given in Problem 1