AA 274A: Principles of Robot Autonomy I
Problem Set 1

Name: Alban Broze
SUID: abroze

10/11/2022

Problem 1: Trajectory Generation via Differential Flatness

(i) The set of linear equations to express the initial and final conditions can be described as a matrix equation: $Ax = b$ where x is the vector containing the coefficients x_i, y_i for $i = 1, ..., 4$; A is the matrix containing information about the functions $\psi_i(t)$ for $i = 1, ..., 4$ at times $t = 0$ and $t = t_F$; and b is the vector containing the corresponding initial and final conditions at times $t = 0$ and $t = t_F$, respectively. In this problem, the $Ax = b$ system is the following:

$$\begin{bmatrix} \psi_1(0) & \psi_2(0) & \psi_3(0) & \psi_4(0) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \psi_1(0) & \psi_2(0) & \psi_3(0) & \psi_4(0) \\ \dot{\psi}_1(0) & \dot{\psi}_2(0) & \dot{\psi}_3(0) & \dot{\psi}_4(0) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \dot{\psi}_1(0) & \dot{\psi}_2(0) & \dot{\psi}_3(0) & \dot{\psi}_4(0) \\ \psi_1(t_F) & \psi_2(t_F) & \psi_3(t_F) & \psi_4(t_F) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \psi_1(t_F) & \psi_2(t_F) & \psi_3(t_F) & \psi_4(t_F) \\ \dot{\psi}_1(t_F) & \dot{\psi}_2(t_F) & \dot{\psi}_3(t_F) & \dot{\psi}_4(t_F) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \dot{\psi}_1(t_F) & \dot{\psi}_2(t_F) & \dot{\psi}_3(t_F) & \dot{\psi}_4(t_F) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} x(0) \\ y(0) \\ \dot{x}(0) = V(0)\cos(\theta(0)) \\ \dot{y}(0) = V(0)\sin(\theta(0)) \\ x(t_F) \\ y(t_F) \\ \dot{x}(t_F) = V(t_F)\cos(\theta(t_F)) \\ \dot{y}(t_F) = V(t_F)\sin(\theta(t_F)) \end{bmatrix}$$

Which simplifies to the following after filling in the values:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 15 & (15)^2 & (15)^3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 15 & (15)^2 & (15)^3 \\ 0 & 1 & 2*(15) & 3*(15)^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2*(15) & 3*(15)^2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \frac{1}{2} * \cos(\frac{-\pi}{2}) = 0 \\ \frac{1}{2} * \sin(\frac{-\pi}{2}) = \frac{-1}{2} \\ 5 \\ 5 \\ \frac{1}{2} * \cos(\frac{-\pi}{2}) = 0 \\ \frac{1}{2} * \sin(\frac{-\pi}{2}) = \frac{-1}{2} \end{bmatrix}$$

(ii) If we set $V(t_f) = 0$, $det(J) = 0$ since $det(J) = V$. If $det(J) = 0$, the matrix J is non-invertible and thus we can't find a and ω. Therefore, we can't set $V(t_f) = 0$. Reminder, J is given in the homework as:

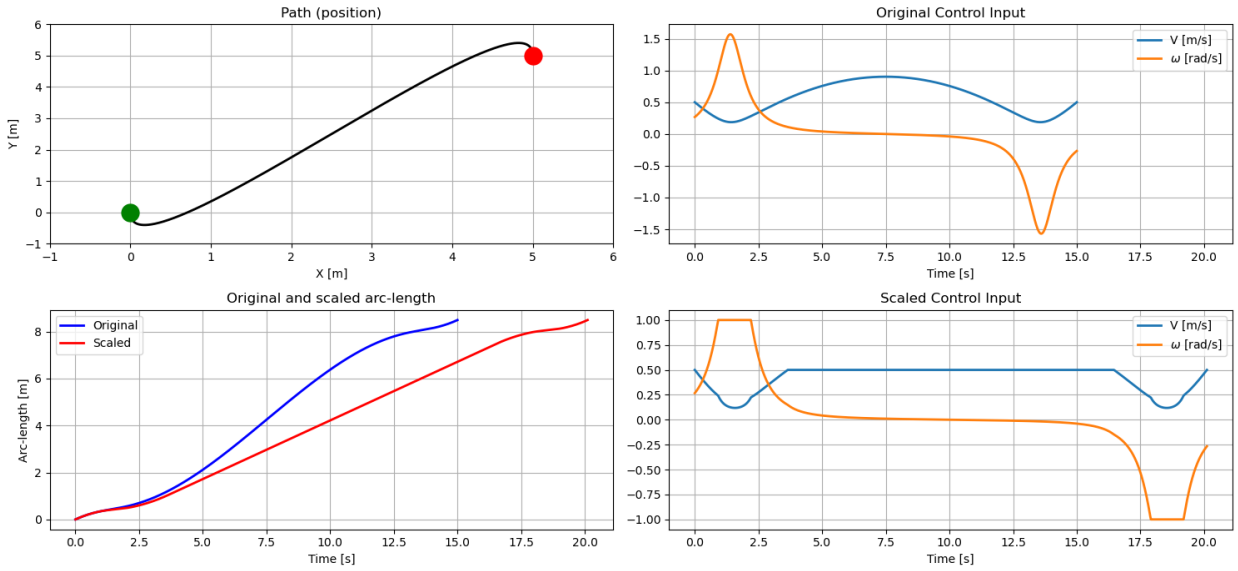
$$J := \begin{bmatrix} \cos(\theta) & -V(t)\sin(\theta) \\ \sin(\theta) & V(t)\cos(\theta) \end{bmatrix}$$

Therefore, $V(t)$ can't be zero at any time.

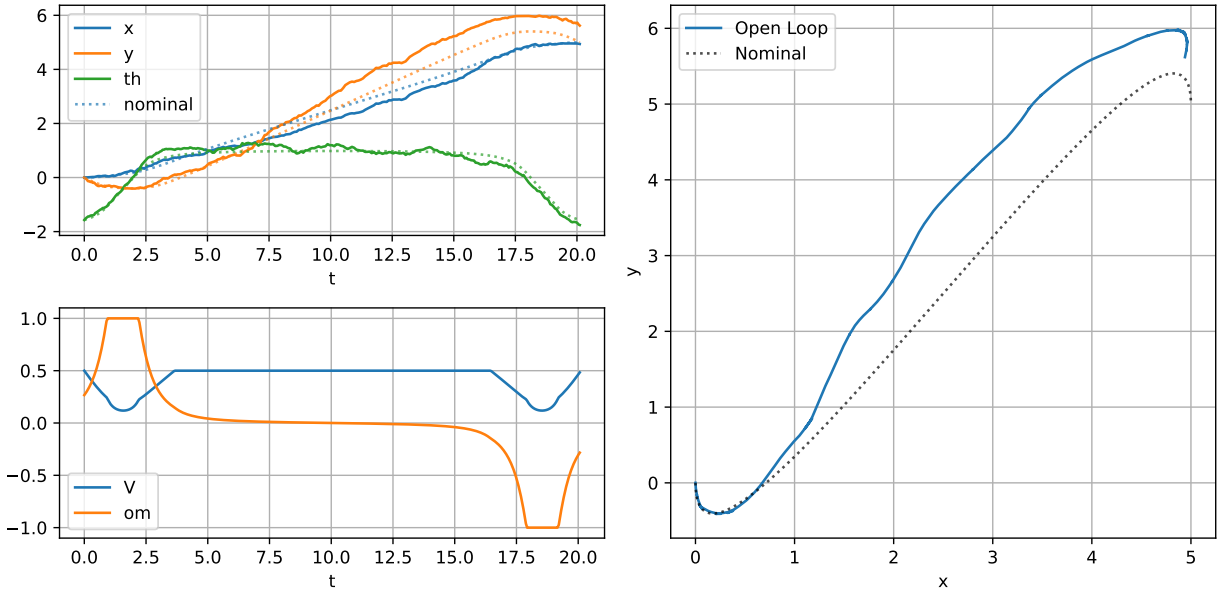
(iii) See code

(iv) See code

(v) Differential flatness:



(vi) Open-loop simulation:

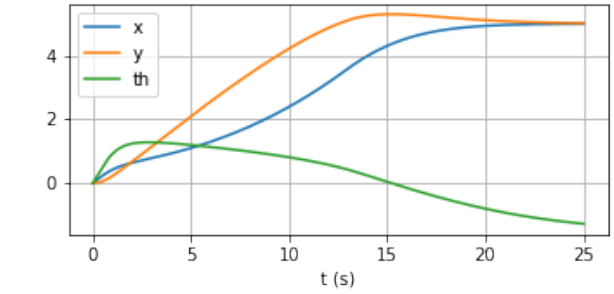


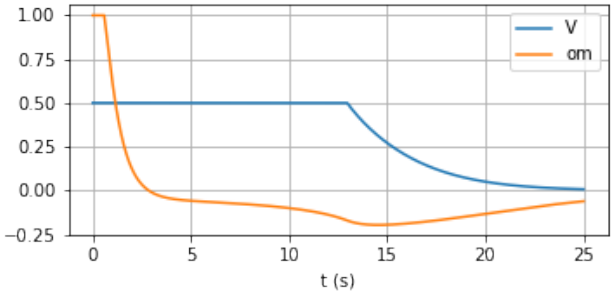
Problem 2: Pose Stabilization

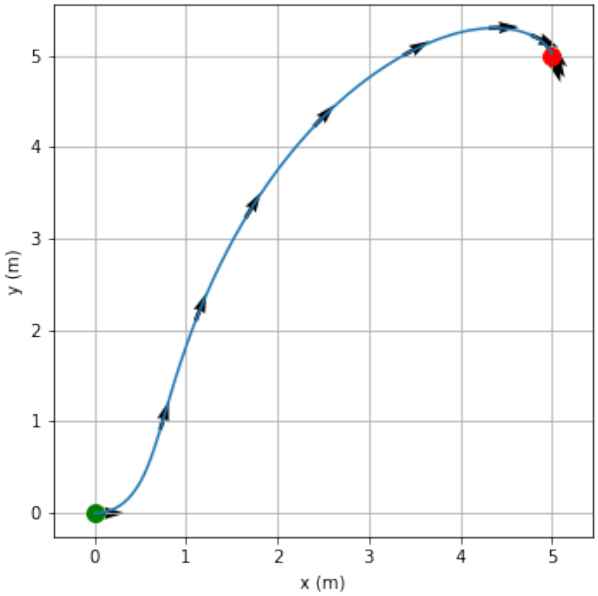
(i) See code

(ii) See code

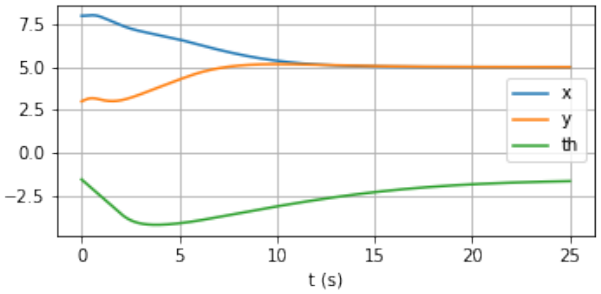
(iii) Forward parking:

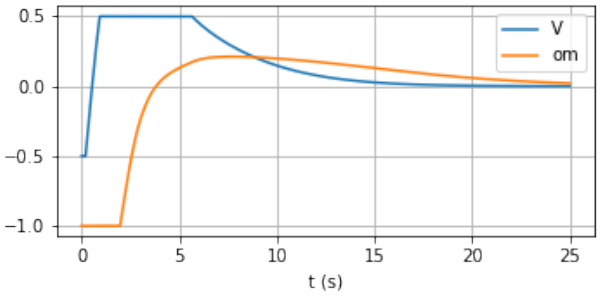


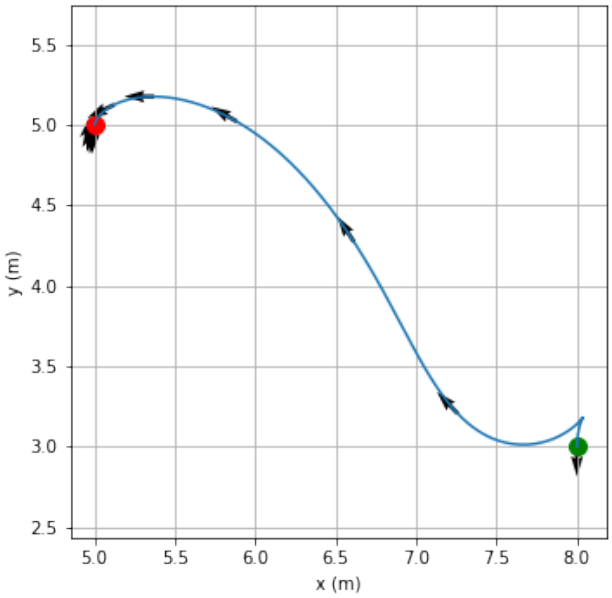




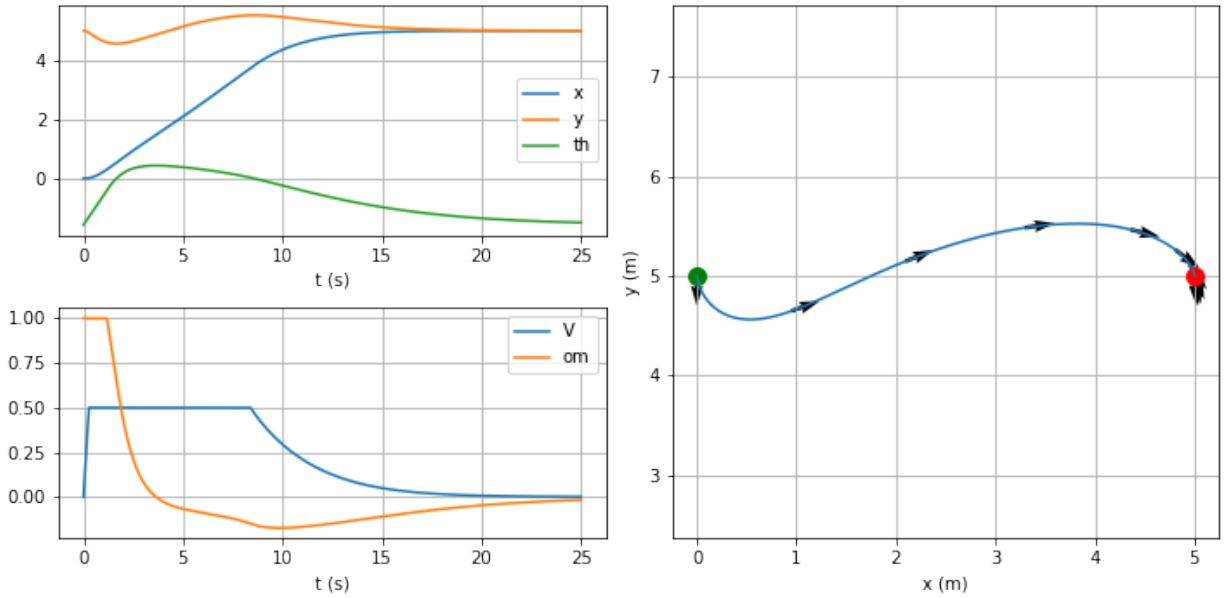
Reverse parking:







Parallel parking:



Problem 3: Trajectory Tracking

(i) The system of equations for computing the true control inputs (V, ω) in terms of the virtual controls (u_1, u_2) can be described as a system of matrices as follows:

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -V(t)\sin(\theta) \\ \sin(\theta) & V(t)\cos(\theta) \end{bmatrix} \begin{bmatrix} \dot{V}(t) \\ \omega \end{bmatrix}$$

This matrix form can be rewritten as two equations:

$$u_1 = \cos(\theta)\dot{V}(t) - V(t)\sin(\theta)\omega$$

$$u_2 = \sin(\theta)\dot{V}(t) + V(t)\cos(\theta)\omega$$

Solving for $\dot{V}(t)$ and ω yields the following equations in terms of (u_1, u_2):

$$\dot{V}(t) = \frac{u_1 + V(t)\sin(\theta)\omega}{\cos(\theta)}$$

$$\omega = \frac{u_2 - \sin(\theta)\dot{V}(t)}{V(t)\cos(\theta)}$$

By substituting the equation for ω into the equation for $\dot{V}(t)$ and simplifying, we get the following expression for $\dot{V}(t)$:

$$\dot{V}(t) = u_1\cos(\theta) + u_2\sin(\theta)$$

To return $V(t)$, we need to integrate $\dot{V}(t)$. The Euler integration method can be used for integration. The following equation describes the Euler method:

$$V[t+1] = V[t] + \dot{V}[t] * \Delta t$$

Combining the last two equations yields an equation for $V(t)$ in terms of (u_1, u_2) at each time step:

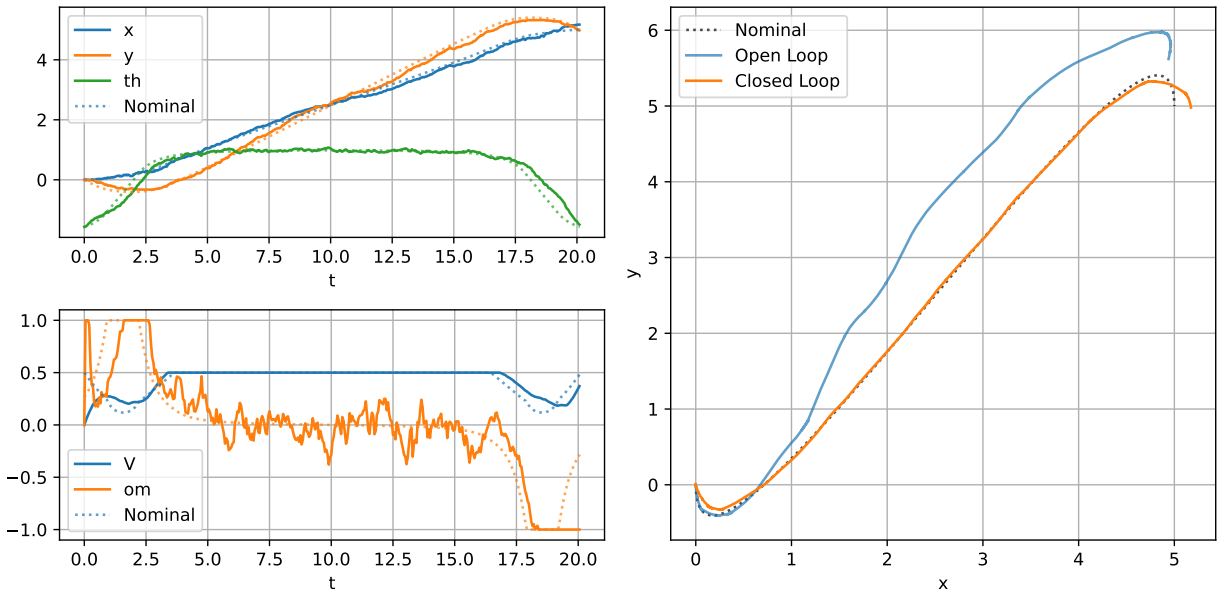
$$V[t+1] = V[t] + (u_1 \cos(\theta) + u_2 \sin(\theta)) * \Delta t$$

Recall the equation for ω:

$$\omega = \frac{u_2 - \sin(\theta)(u_1 \cos(\theta) + u_2 \sin(\theta))}{V(t)\cos(\theta)}$$

(ii) See code

(iii) Closed-loop simulation:



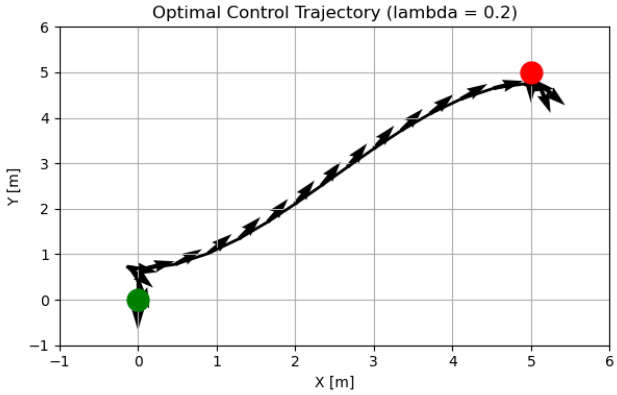
Extra Problem: Optimal Control and Trajectory Optimization

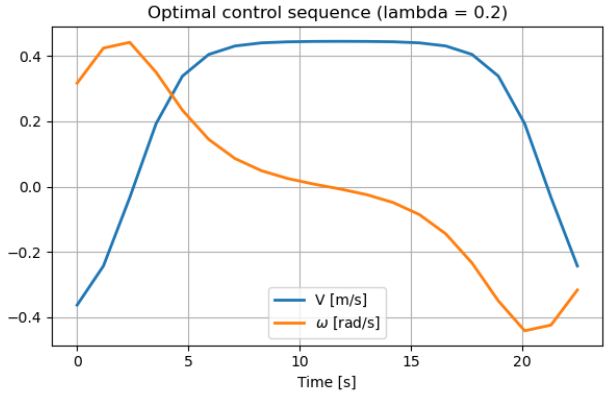
(i) The optimal control problem can be transcribed into a finite dimensional constrained optimization problem of the following form:

$$\begin{aligned}
\min_{(x_i,u_i)} \quad & \sum_{i=0}^{N} \Delta t_i(\lambda + V_i[t_i]^2 + \omega_i[t_i]^2) \\
\text{s.t.} \quad & |V_i[t_i]| \leq 0.5 \quad \text{m/s} \quad \text{for} \quad i = 0, 1, ..., N \\
& |\omega_i[t_i]| \leq 1.0 \quad \text{rad/s} \quad \text{for} \quad i = 0, 1, ..., N \\
& \mathbf{x}_0[t_0] = [x_0, y_0, \theta_0] = [0, 0, -\pi/2] \\
& \mathbf{x}_F[t_N] = [x_F, y_F, \theta_F] = [5, 5, -\pi/2] \\
& \mathbf{x}_{i+1}[t_{i+1}] = \mathbf{x}_i[t_i] + \Delta t \dot{\mathbf{x}}_i[t_i] \\
& \Delta t = t_N / N
\end{aligned} \tag{1}$$

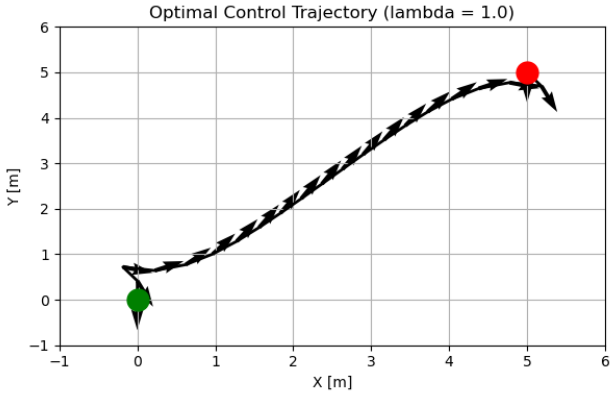
(ii) See code

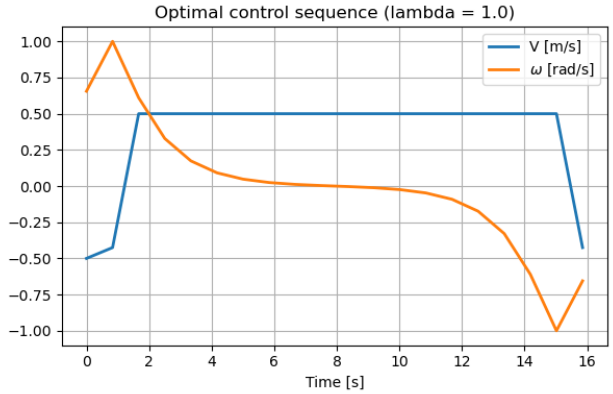
(iii) Optimal control with $\lambda = 0.2$



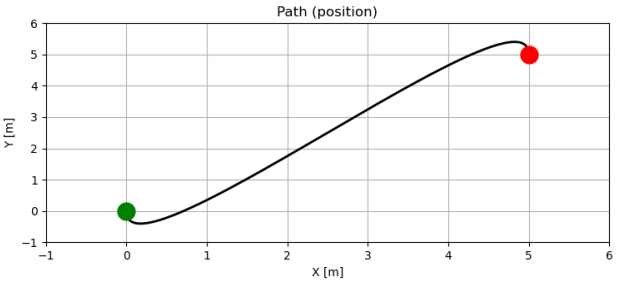


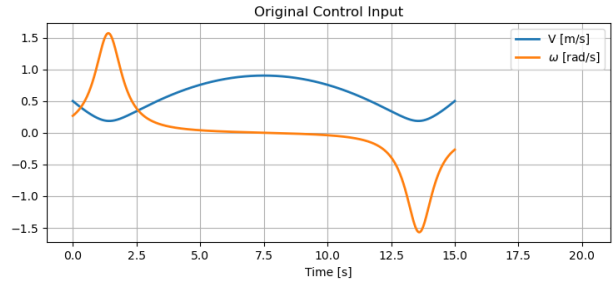
Optimal control with $\lambda = 1.0$

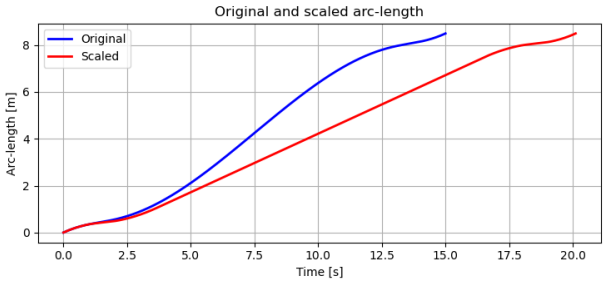


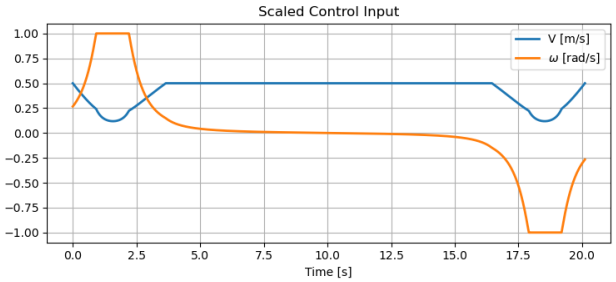


Recall the trajectory obtained through the differential flatness approach in Problem 1:









In terms of **trajectory**, the results obtained through the differential flatness approach in Problem 1 look similar to the results obtained via optimal control, except at the beginning of the trajectory, where a little difference can be noticed. As a matter of fact, in the optimal control problem (for both λ's), the robot directly spins around itself, tries one direction (opposite compared to the goal state) and then iterates back to the trajectory in direction of the goal. In the differential flatness approach,

this turn-around is more smooth, but eventually takes around the same amount of time.

In terms of **linear velocity** V and **angular velocity** ω, some differences as well as some similarities can be noticed.

Differences:

(a) The velocity profile for the differential flatness approach is different than the one obtained via the optimal control trajectory for $\lambda = 0.2$. The first one is flatter (V keeps the same value 0.5 m/s for a much longer period), while the latter has a more curved shape and never reaches the upper-bound value of 0.5 m/s.

(b) In the differential flatness approach, the value of ω remains constant and equal to 0 for a long period of time, meaning that the robot is on the good trajectory to reach its objective for a large fraction of its trajectory. This is not the case in the optimal control trajectories, in which the robot never really has a constant angular velocity. This could be seen as if the robot was heading towards one (slightly inaccurate) direction for a long time and slowly adjusting its orientation to eventually reach its goal.

(c) In the differential flatness approach, as opposed to the optimal control method, no negative velocities are recorded. The robot goes straight to its objective, as mentioned earlier above.

Similarities:

(a) In both the differential approach and the optimal control trajectory for $\lambda = 1.0$, the robot maintains a constant velocity (maximum velocity) for a large fraction of its trajectory.

(b) In both the differential approach and the optimal control trajectory for all λ's, the angular velocity profile ω kind of follows the same shape, i.e. being positive at the beginning of the trajectory, crossing zero around half of the trajectory, and eventually becoming negative towards the end of the trajectory to reach the goals state.

(iv) In terms of trajectory, they are pretty much the same for both values of λ. However, significant differences can be noticed for the linear velocity profile, the angular velocity profile and the time it takes to achieve the goal state. Let's break down the comparison into these three components:

Linear velocity profile (V):
The linear velocity profile is smoother for $\lambda = 0.2$ than for $\lambda = 1.0$. Choosing a smaller value for could thus help in achieving smoother actuation, which is preferred.

Angular velocity profile (ω):
As for the linear velocity profile, the angular velocity profile is smoother for $\lambda = 0.2$ than for $\lambda = 1.0$. Choosing a smaller value for λ could thus help in achieving smoother actuation, which is preferred.

Time to achieve goal state ($\mathbf{t}_F$):
The robot takes more time to achieve the goal state for $\lambda = 0.2$ than for $\lambda = 1.0$. In one case, it takes about 16.5 seconds, while it takes around 23.5 seconds in the other case.

In the end, the different values of lambda allow to make engineering design choices between a smoother trajectory and smoother actuation but a slower robot, or a more agressive trajectory and actuation but a faster arrival at the final goal state.