AA 274: Principles of Robotic Autonomy
Problem Set 1

Name: Ashar Alam
SUID: 06265091 ashar1

January 23, 2019

Problem 1: Optimal Control

(i) We use the state vector $x = (x, y, \theta)$ and Control Vector $u = (V, \omega)$ and vector $p \in R^3$, we can find the **Hmiltonian** as:

$$H(\mathbf{x,y,u}) = \lambda + V^2 + \omega^2 + p_1 V \sin\theta + p_2 V \cos\theta + p_3\omega \tag{1}$$

Using Hamiltonian equations as shown below:

$$\dot{\mathbf{x}}^*(t) = \frac{\partial H}{\partial \mathbf{p}}(\mathbf{x}^*(t), \mathbf{u}^*(t), \mathbf{p}^*(t), t)$$
$$\dot{\mathbf{p}}^*(t) = -\frac{\partial H}{\partial \mathbf{x}}(\mathbf{x}^*(t), \mathbf{u}^*(t), \mathbf{p}^*(t), t)$$
$$\mathbf{0} = \frac{\partial H}{\partial \mathbf{u}}(\mathbf{x}^*(t), \mathbf{u}^*(t), \mathbf{p}^*(t), t)$$

Applying these conditions to our Hamiltonian we get:

$$\dot{\mathbf{x}}^* = \frac{\partial H}{\partial p} = (x^*, u^*, p^*, t) = \begin{bmatrix} V^* cos(\theta^*) \\ V^* sin(\theta^*) \\ \omega^* \end{bmatrix} \tag{2}$$

$$\dot{\mathbf{p}}^* = -\frac{\partial H}{\partial x} = (x^*, u^*, p^*, t) = \begin{bmatrix} 0 \\ 0 \\ V^*(p_1^* sin(\theta^*) - p_2^* cos(\theta^*)) \end{bmatrix} \tag{3}$$

$$\mathbf{0} = -\frac{\partial H}{\partial u} = (x^*, u^*, p^*, t) = \begin{bmatrix} 2V^* + p_1^* cos\theta^* + p_2^* sin\theta^* \\ 2\omega^* + p_3^* \end{bmatrix} \tag{4}$$

Solving Equation (2), (3) and (4), we get

$$V^* = -0.5 * (p_2^* sin\theta^* + p_1^* cos\theta^*), \quad \omega^* = -\frac{-p_3^*}{2} \tag{5}$$

Taking into consideration our boundary conditions, we may re-scale t_f as $r = \frac{t}{t_f} \in [0, 1]$ with new vector $\mathbf{z} = (x, p, r)$ to form new BVP as:
for $\quad$ *i = 1,2,3*

$$\frac{d\dot{\mathbf{z}}_{\mathbf{i}}^*(\tau)}{d\tau} = r^*(\tau) \begin{bmatrix} V^*(\tau) cos(\theta^*(\tau)) \\ V^*(\tau) sin(\theta^*(\tau)) \\ \omega^*(\tau) \end{bmatrix} \tag{6}$$

for i = 4,5,6

$$\frac{d\dot{\mathbf{z}}_{\mathbf{i}}^{*}(\tau)}{d\tau} = r^{*}(\tau)\begin{bmatrix} 0 \\ 0 \\ V^{*}(\tau)(p_1^{*}sin(\theta^{*}(\tau)) - p_2^{*}cos(\theta^{*}(\tau))) \end{bmatrix} \tag{7}$$

$$\frac{d\dot{\mathbf{z}}_{\mathbf{7}}^{*}(\tau)}{d\tau} = \frac{dr}{d\tau} = 0 \tag{8}$$

Also, the earlier boundary conditions:

$$\mathbf{x}(0) = \begin{bmatrix} 0 \\ 0 \\ -\frac{\pi}{2} \end{bmatrix}, \quad \mathbf{x}(t_f) = \begin{bmatrix} 5 \\ 5 \\ -\frac{\pi}{2} \end{bmatrix} becomes \tag{9}$$

$$\mathbf{x}^{*}(0) = \begin{bmatrix} 0 \\ 0 \\ -\frac{\pi}{2} \end{bmatrix}, \quad \mathbf{x}^{*}(1) = \begin{bmatrix} 5 \\ 5 \\ -\frac{\pi}{2} \end{bmatrix} \tag{10}$$

We further use the boundary conditions for fixed position and free time

t_f	free	δt_f arbitrary
$\mathbf{x}(t_f)$	fixed	$\delta \mathbf{x}_f = 0$

BC

$$\mathbf{x}^{*}(t_0) = \mathbf{x}_0$$
$$\mathbf{x}^{*}(t_f) = \mathbf{x}_f$$
$$H(\mathbf{x}^{*}(t_f), \mathbf{u}^{*}(t_f), \mathbf{p}^{*}(t_f), t_f) + \frac{\partial h}{\partial t}(\mathbf{x}^{*}(t_f), t_f) = 0$$

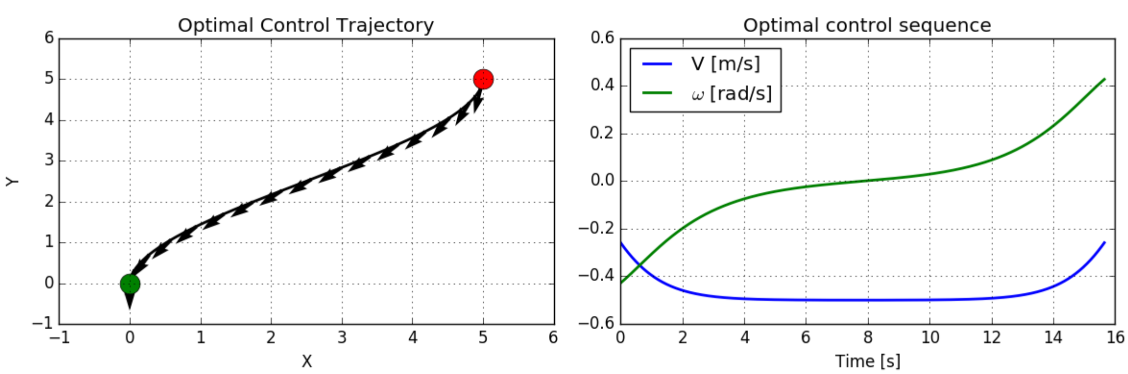
to get

$$H(\mathbf{x}^{*}(1), \mathbf{u}^{*}(1), \mathbf{p}^{*}(1)) = 0$$

Thus, we have obtained a 2P-BVP problem with 7 ODE's and 7 unknown variables in $\mathbf{z_i}$

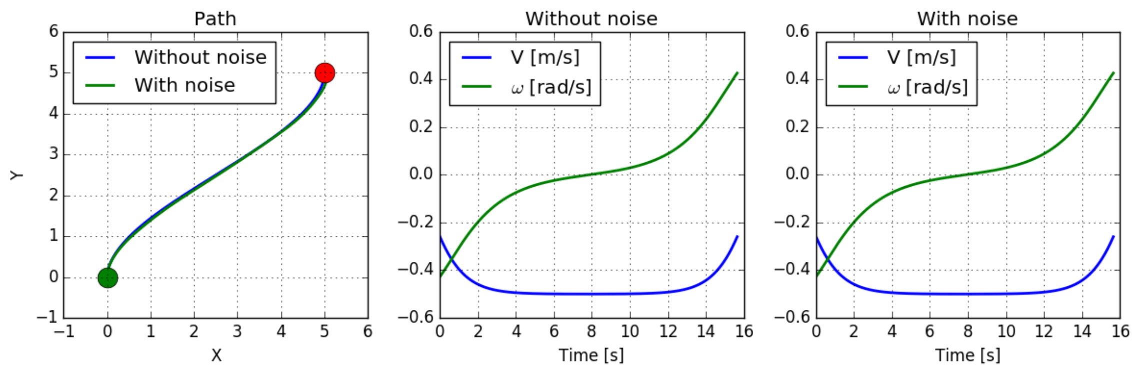
(ii) Code is included in the zip file.

(iii) We obtained the following plots using the python P1_optimal_control.py command:



(iv) Intuitively, largest λ will result in moving the car quickly to make sure t_f is small and Velocity is maximum. This is because λ is a penalty for optimization of t_f. So, larger λ results in smaller t_f.

(v) I used $\lambda = 0.25$ and initial guess as [1.0, 1.0, 1.0, -$\pi/2$, -1.0, -1.0, -1.0, 20.0]. We obtained the following plots using the python sim_traj.py –data optimal_control –dist 1 –ctrl open command:



Both the plots with and without noise almost coincide.

Problem 2: Differential Flatness

(i) We can arrange the basis functions and obtain them in the following form

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & t_f & t_f^2 & t_f^3 \\ 0 & 1 & 2t_f & 3t_f^2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} x(0) \\ \dot{x}(0) \\ x(t_f) \\ \dot{x}(t_f) \end{bmatrix} \tag{11}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & t_f & t_f^2 & t_f^3 \\ 0 & 1 & 2t_f & 3t_f^2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} y(0) \\ \dot{y}(0) \\ y(t_f) \\ \dot{y}(t_f) \end{bmatrix} \tag{12}$$

Using initial conditions given below:

$$\begin{bmatrix} x(0) \\ y(0) \\ \dot{x}(0) \\ \dot{x}(0) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -0.5 \end{bmatrix} \quad \begin{bmatrix} x(t_f) \\ y(t_f) \\ \dot{x}(t_f) \\ \dot{x}(t_f) \end{bmatrix} = \begin{bmatrix} 5 \\ 5 \\ 0 \\ 0.5 \end{bmatrix} \tag{13}$$

Equations (11) and (12) will result in the required linear equations for x and y in terms of x_i's and y_i's

(ii) If we set $V(t_f) = 0$, then $det(J) = 0$. Thus, J will become non-invertible as $det(J) = V$ and we will no longer be able to solve the equations in $\ddot{x}$ and $\ddot{y}$

(iii) Code is included in the zip file.

(iv) We know

$$\theta = tan^{-1}\frac{\dot{y}}{\dot{x}} \quad V = \sqrt{\dot{x}^2 + \dot{y}^2} \tag{14}$$

We can also write V as

$$V = \frac{\dot{x}}{cos\theta} or V = \frac{\dot{y}}{sin\theta} \tag{15}$$

From J as defined in the problem we have

$$\ddot{x} = acos\theta - \omega V sin\theta \tag{16}$$

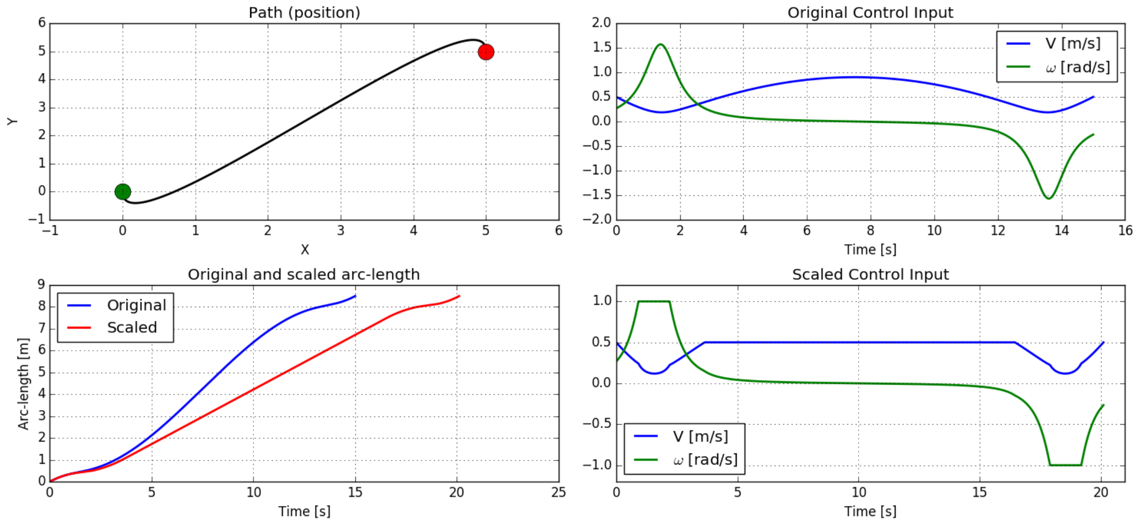
$$\ddot{y} = asin\theta - \omega V cos\theta \tag{17}$$

Solving, Equation (16) (17), we get

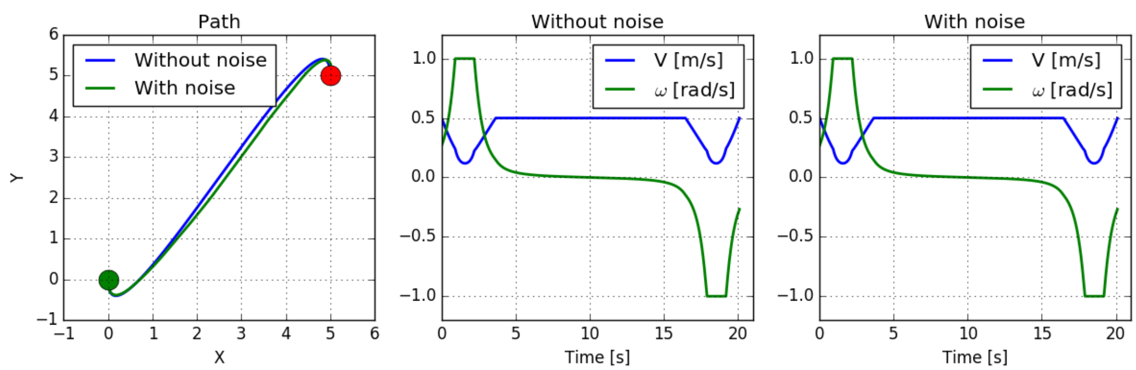
$$\omega = \frac{\dot{x}\ddot{y} - \dot{y}\ddot{x}}{V^2} \tag{18}$$

We implement the differential trajectory code using sign function in Python.

(v) We obtained the following plots using the differential_flatness.png command:



(vi) We obtained the following plots using the python sim_traj.py –data differential_flatness –dist 1 –ctrl open command:



Problem 3: Closed Loop Control I

(i) Code is included in the zip file.

(ii)
- **Forward parking**

$$x_0 = 5, y_0 = 3, \theta_0 = \pi/2, t_{end} = 20$$

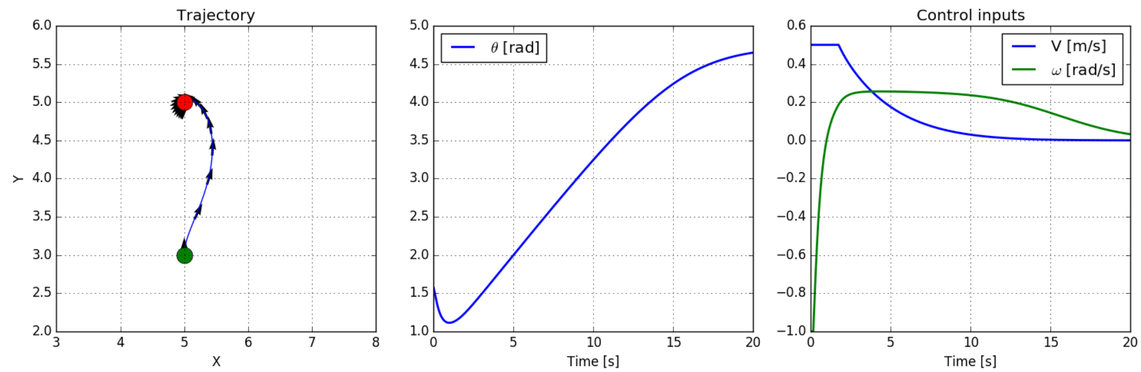
- **Reverse parking**

$$x_0 = 5, y_0 = 3, \theta_0 = -\pi/2, t_{end} = 20$$

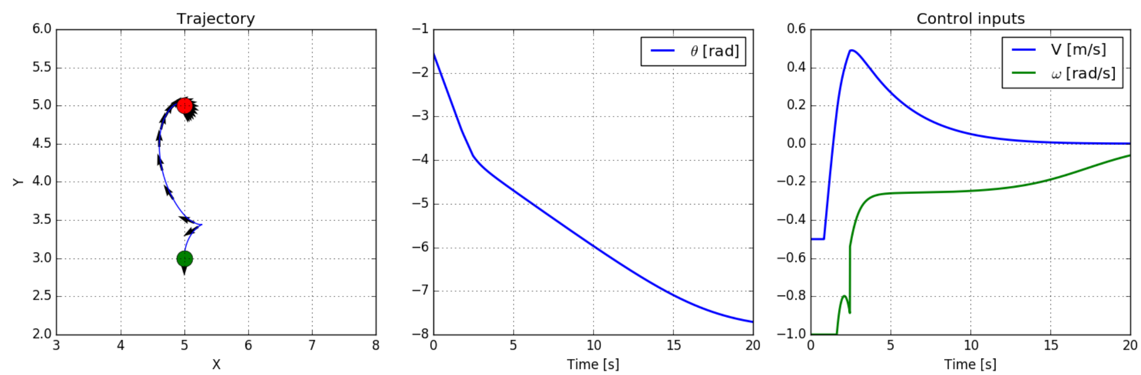
- **Parallel parking**

$$x_0 = 7, y_0 = 3, \theta_0 = \pi/2, t_{end} = 20$$

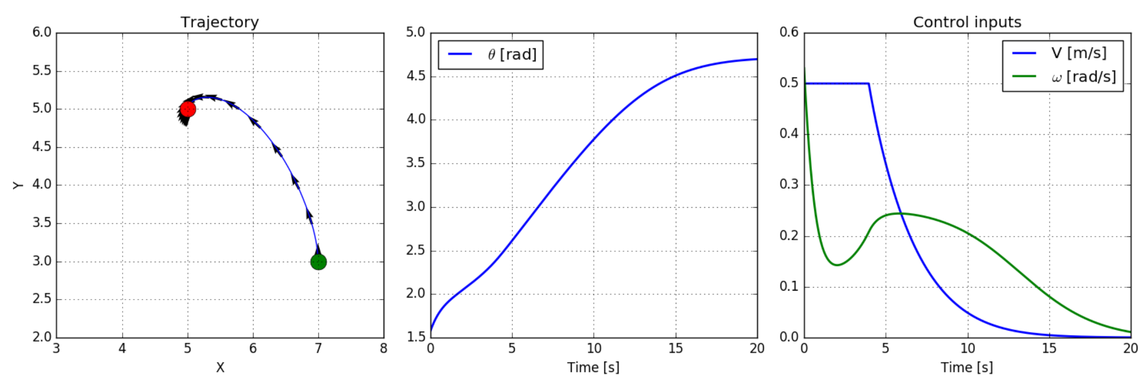
(iii) Plots



(i) Forward Parking



(ii) Reverse Parking



(iii) Parallel Parking

Problem 4: Closed Loop Control II

(i) From Problem 2 and Equation(16) and Equation (17) we have

$$u_1 = \ddot{x} = acos\theta - \omega V sin\theta \tag{19}$$

$$u_2 = \ddot{y} = asin\theta - \omega V cos\theta \tag{20}$$

Solving (19) and (20), we get a and ω in terms of u, V and θ

$$a = u_2 sin(\theta) + u_1 cos(\theta) = \dot{(V)} \tag{21}$$

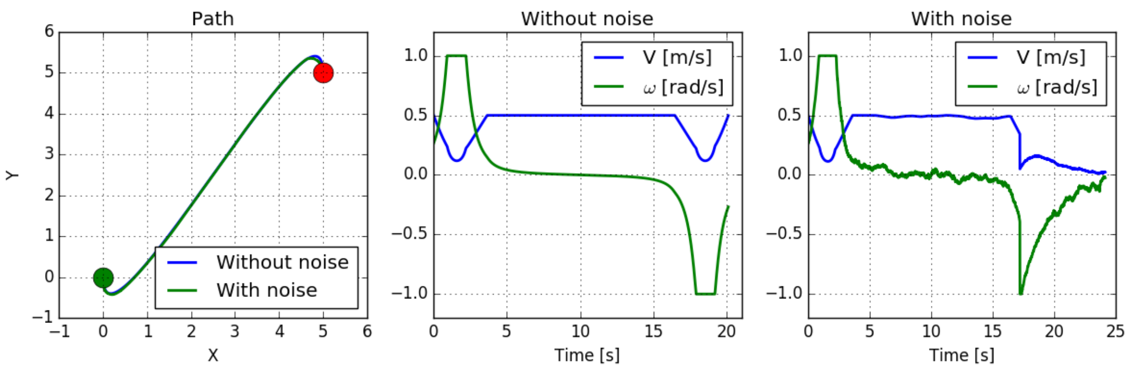
and

$$\omega = \frac{u_2 cos(\theta) - u_1 sin(\theta)}{V} \tag{22}$$

Implementing this in code while taking care of singularity i.e. when V = 0, we can obtain the desired results.

(ii) Code is included in the zip file.

(iii) We obtained the following plots using the python sim_traj.py –data differential_flatness –dist 1 –ctrl closed command:



Problem 5: ROS

(i) The rosbag is saved in the file named **name.bag**

(ii) The command to play a rosbag with the given name is: rosbag play **bagname.bag**

(iii) We must run the command: **chmod a+x nodename.py**.
This command makes the scripts executable. The flag a is used to give the privilege to all users. The flag x is used to make the file executable. Thus, the composite flag a+x gives all users (either in the file's group or other user and obviously the super user) the ability to run the file as executable.

(iv) The command: **rostopic list** is used to display the list of active topics (information about multiple topics).
To get information about the current topic we may use the command: **rostopic info topic-name**

(v) The rosbag is saved in the file named **gazebo.bag**

.
.
Comments:
My approach for recording a ROSbag

- Launch ROSmaster with roscore
- Launch your publisher
- Record your Rosbag

My approach for playing a ROSbag

- Launch ROSmaster with roscore
- Launch your rosbag
- rostopic echo your bag topic to see the contents printed