ME455 HW4 - Zhengyang Kris Weng Submission

05/16/2025

1.

1. (20 pts) The solution of (5) can be calculated through the following ODEs:

$$p(t)^\top B(t) + b_v(t)^\top = 0 \tag{7}$$
$$\dot{p}(t) = -A(t)^\top p(t) - a_z(t) \tag{8}$$
$$\dot{z}(t) = A(t)z(t) + B(t)v(t), \tag{9}$$

with the initial and terminal conditions being:

$$z(0) = 0, \quad p(T) = p_1. \tag{10}$$

These ODEs can be re-organized into the following two-point boundary value problem, which does not involve $v(t)$ at all:

$$\begin{bmatrix} \dot{z}(t) \\ \dot{p}(t) \end{bmatrix} = \underbrace{\begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}}_{M} \begin{bmatrix} z(t) \\ p(t) \end{bmatrix} + \begin{bmatrix} m_1 \\ m_2 \end{bmatrix}, \quad \begin{bmatrix} z(0) \\ p(T) \end{bmatrix} = \begin{bmatrix} 0 \\ p_1 \end{bmatrix}. \tag{11}$$

What should the $a_z(t)$ and $b_v(t)$ be? Note that they are different from the terms $a_x(t)$ and $b_u(t)$ above. What should the block matrix M look like? What should the vectors m_1 and m_2 look like? (Hint: the block matrix M and vectors m_1 and m_2 should not include $v(t)$ at all.) Lastly, how to calculate $v(t)$ once you have solved the above two point boundary value problem?

Turn in: The expressions for $a_z(t)$, $b_v(t)$, the block matrix M, the vectors m_1 and m_2, and $v(t)$ (assuming $p(t)$ and $z(t)$ are solved).

Figure 1: 1

$$a_z(t) = l_x(t)^T + Q_z z(t) = (D_1 l(x(t)^{[k]}, u(t)^{[k]}))^T + Q_z z(t)$$
$$b_u(t) = (D_2 l(x(t)^{[k]}, u(t)^{[k]}))^T + R_v v(t)$$

$$M = \begin{bmatrix} A(t) & -B(t)R_v(t)^{-1}B(t)^T \\ -Q_z(t) & -A(t)^T \end{bmatrix}$$

From the matrix above:

$$m_1(t) = -B(t)R_v(t)^{-1} l_u(t)^T = -B(t)R_v(t)^{-1}(D_2 l(x(t)^{[k]}, u(t)^{[k]}))^T$$

$$m_2(t) = -l_x(t)^T = -(D_1 l(x(t)^{[k]}, u(t)^{[k]}))^T$$

$$v(t) = -R_v(t)^{-1}(B(t)^T p(t) + l_u(t)^T)$$

2.

2. (20 pts) Solve the following 2D optimization problem for the variable $x = [x_1, x_2]$:

$$\begin{aligned} x^* &= \arg\min_x f(x) \\ &= \arg\min_x \; 0.26 \cdot (x_1^2 + x_2^2) - 0.46 \cdot x_1 x_2 \end{aligned} \tag{14}$$

using gradient descent with Armijo line search. The line search process in each iteration is summarized in the pseudocode below. Note that, in practice, the parameter α should be small (between 10^{-4} to 10^{-2}) and the parameter β should be between 0.2 to 0.8. Use the initial guess of the variable $x = [-4, -2]$, use the following parameters $\gamma_0 = 1, \alpha = 10^{-4}, \beta = 0.5$, run for 100 iterations in total.

Turn in: A plot showing the trajectory of the iterations over the contour of the objective function, see the example figure above.

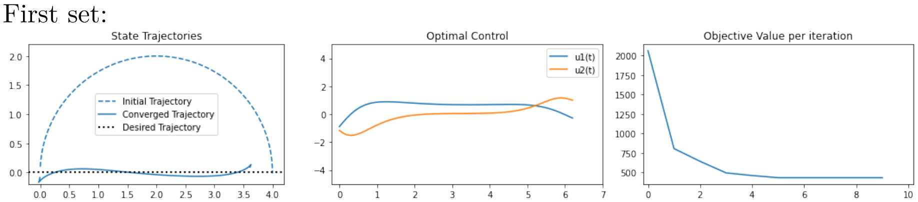
Figure 2: 2

See code implementation in `hw4.ipynb`

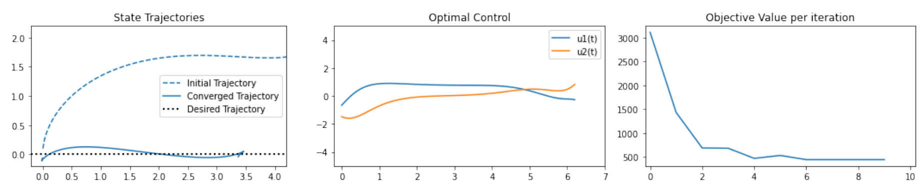
3.

See code implementation in `hw4.ipynb`

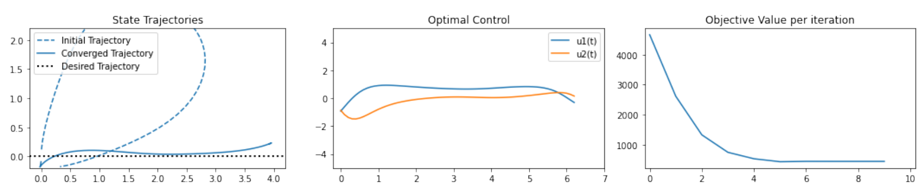
First set:



Second set:



Third set:



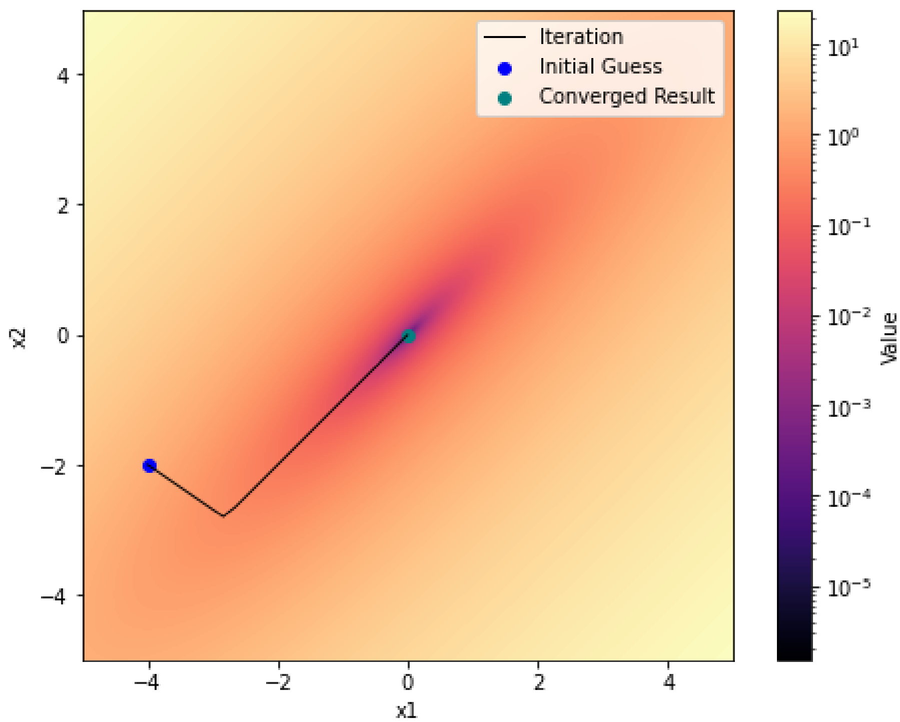


Figure 3: Q2

3. (60 pts) Apply iLQR to the differential drive vehicle for a length of time $T = 2\pi sec$ to track the desired trajectory $(x_d(t), y_d(t), \theta_d(t)) = (\frac{4}{2\pi}t, 0, \pi/2)$ subject to the dynamics:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta)u_1 \\ \sin(\theta)u_1 \\ u_2 \end{bmatrix}, \ (x(0), y(0), \theta(0)) = (0, 0, \pi/2). \tag{13}$$

Note that the desired trajectory corresponds to an infeasible trajectory for parallel parking. A Python template for iLQR can be found here: `https://drive.google.com/file/d/1Br8DArJtnEZXjZok2aWh7PMVoTuRq1hc/view?usp=sharing`, you should try different parameters and initial control trajectories to see their effect on the optimal trajectory.

Figure 4: 3