Homework 4: Optimal Control

Due: 11:59PM, May 14, 2025

Software

Python packages *NumPy*, *SciPy* and *Matplotlib* are sufficient for the calculation used this assignment. *SciPy* contains the function "solve_bvp", which can be used to solve the two-point boundary value problem in this homework. The computation of this assignment could take a while to finish on your computer, but it should not take more than *a few minutes* per iLQR iteration. Feel free to use Jupyter Notebook (or Google Colab) as the programming environment.

Preliminaries

Given the dynamics of a system $\dot{x}(t) = f(t, x(t), u(t))$ and the initial state $x(0) = x_0$, we want to calculate the optimal control signal $u(t)$ for the system to optimally track a desired trajectory $x_d(t)$ within the time horizon $[0, T]$. This optimal control problem is defined as:

$$\begin{aligned} u(t)^* &= \arg\min_{u(t)} J(u(t)) \\ &= \arg\min_{u(t)} \int_0^T \left[\underbrace{(x(t) - x_d(t))^\top Q_x (x(t) - x_d(t)) + u(t)^\top R_u u(t)}_{l(x(t), u(t))} \right] dt \\ &\qquad + \underbrace{(x(T) - x_d(T))^\top P_1 (x(T) - x_d(T))}_{m(x(T))} \end{aligned} \tag{3}$$

$$s.t. \quad x(t) = x_0 + \int_0^t f(x(\tau), u(\tau)) d\tau. \tag{4}$$

We can solve this problem using iLQR. In the k-th iteration, given the current estimation of the optimal control $u(t)^{[k]}$ and the corresponding system trajectory $x(t)^{[k]}$, we need to calculate the optimal descent direction, denoted as $v(t)^{[k]}$, through another optimal control problem below:

$$\begin{aligned} v(t)^{[k]} = \arg\min_{v(t)} & \underbrace{\int_0^T \underbrace{D_1 l(x(t)^{[k]}, u(t)^{[k]})}_{a_x(t)} \cdot z(t) + \underbrace{D_2 l(x(t)^{[k]}, u(t)^{[k]})}_{b_u(t)} \cdot v(t) dt + \underbrace{Dm(x(T)^{[k]})}_{p_1} \cdot z(T)}_{DJ(u(t)) \cdot v(t)} \\ & + \int_0^T z(t)^\top Q_z z(t) + v(t)^\top R_v v(t) dt, \end{aligned} \tag{5}$$

where $z(t)$ and $v(t)$ are governed by the following linear dynamics:

$$z(t) = \underbrace{z_0}_{z_0 = 0} + \int_0^t \underbrace{D_1 f(x(\tau)^{[k]}, u(\tau)^{[k]})}_{A(\tau)} \cdot z(\tau) + \underbrace{D_2 f(x(\tau)^{[k]}, u(\tau)^{[k]})}_{B(\tau)} \cdot v(\tau) d\tau. \tag{6}$$

Once the optimal descent direction is calculated, we can use it to update the control for the next iteration using Armijo line search.

Problems

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).

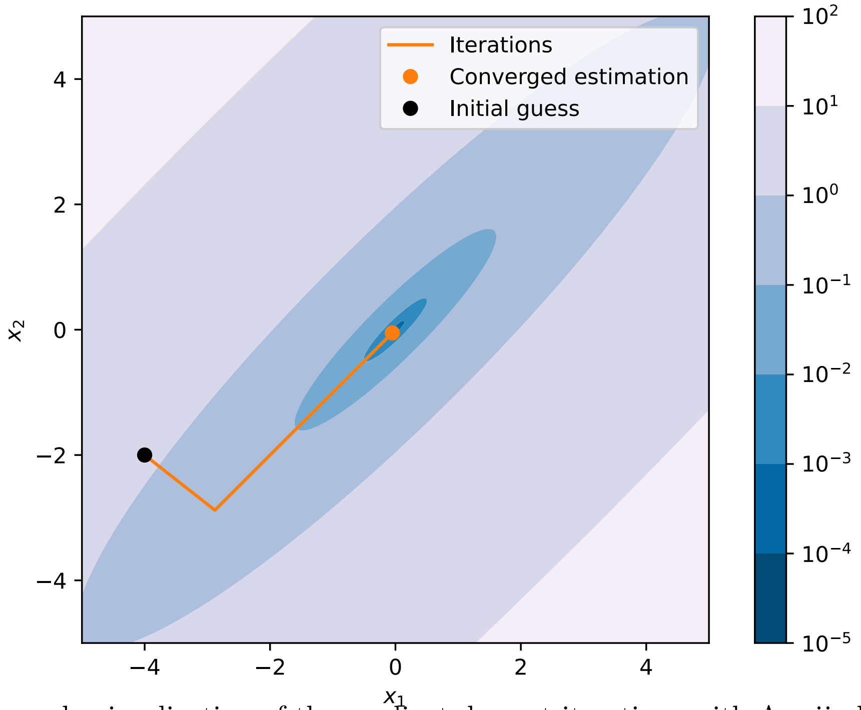

Figure 6: Example visualization of the gradient descent iterations with Armijo line search.

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

$$\begin{aligned} x^* &= \arg\min_x f(x) \\ &= \arg\min_x \; 0.26 \cdot (x_1^2 + x_2^2) - 0.46 \cdot x_1 x_2 \end{aligned} \tag{12}$$

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.

Algorithm 1 Armijo line search

```
1: procedure ARMIJO($x^{[k]}, \gamma_0, \alpha, \beta$)
2:     $\gamma \leftarrow \gamma_0$                         ▷ $\gamma$ is the step size, $\gamma_0$ is the initial step size.
3:     $z^{[k]} \leftarrow -\nabla J(x^{[k]})$              ▷ $z^{[k]}$ is the descent direction for this iteration.
4:     while $J(x^{[k]} + \gamma z^{[k]}) > J(x^{[k]}) + \alpha\gamma\nabla J(x^{[k]})^\top z^{[k]}$ do    ▷ Check the Armijo condition
5:         $\gamma \leftarrow \beta\gamma$        ▷ If the Armijo condition is not met, scale down the step size by $\beta$.
6:     end while
7:     return $x^{[k]} + \gamma z^{[k]}$
8: end procedure
```

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.

Turn in: Select three sets of different initial control trajectories and objective parameters, with one of the initial control trajectories being $[1, -0.5]$ for the whole horizon. For each set of parameters, choose a convergence criterion and run the algorithm until convergence. For each set of parameters, generate a plot with the following content (see the example below): (1) The initial and converged system trajectory; (2) The optimal control signals; (3) Iterations of the objective function value. Submit three such plots and include the parameters you use.

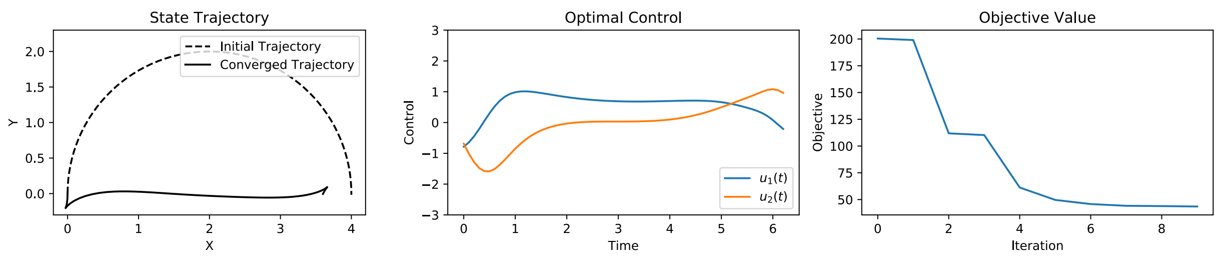

Figure 7: Example visualization for iLQR.