Principles of Robot Autonomy I: Homework 2

Student Name

October 16, 2024

Problem 1

(i) **Transcription to Finite Dimensional Optimization Problem**

To transcribe this optimal control problem into a finite-dimensional constrained optimization problem, we use direct transcription with discretization. Let's discretize the time interval $[0, t_f]$ into N segments with time step Δt, where $t_i = i \cdot \Delta t$ for $i = 0, 1, \ldots, N$, and $t_N = t_f$.

Decision Variables:

The optimization variables are:

$$\mathbf{z} = [x_0, y_0, \theta_0, \ldots, x_N, y_N, \theta_N, v_0, \omega_0, \ldots, v_{N-1}, \omega_{N-1}, t_f] \tag{1}$$

This gives us $3(N+1) + 2N + 1 = 5N + 4$ decision variables.

Objective Function:

The continuous cost functional is discretized using numerical integration (e.g., trapezoidal rule or rectangle rule):

$$\min_{\mathbf{z}} \quad J = \sum_{i=0}^{N-1} \left(\alpha + v_i^2 + \omega_i^2\right) \Delta t \tag{2}$$

where $\Delta t = t_f / N$.

Constraints:

1. Initial Conditions:

$$x_0 = 0 \tag{3}$$
$$y_0 = 0 \tag{4}$$
$$\theta_0 = \pi/2 \tag{5}$$

2. Final Conditions:

$$x_N = 5 \tag{6}$$
$$y_N = 5 \tag{7}$$
$$\theta_N = \pi/2 \tag{8}$$

3. Dynamics Constraints (using forward Euler integration):

For $i = 0, 1, \ldots, N-1$:

$$x_{i+1} = x_i + v_i \cos(\theta_i) \Delta t \tag{9}$$
$$y_{i+1} = y_i + v_i \sin(\theta_i) \Delta t \tag{10}$$
$$\theta_{i+1} = \theta_i + \omega_i \Delta t \tag{11}$$

4. Collision Avoidance Constraints:

For $i = 0, 1, \ldots, N$:

$$\sqrt{(x_i - 2.5)^2 + (y_i - 2.5)^2} - 0.4 \geq 0 \tag{12}$$

where $0.4 = r_{ego} + r_{obstacle} = 0.1 + 0.3$.

Complete Optimization Problem:

$$\begin{aligned}
\min_{\mathbf{z}} \quad & \sum_{i=0}^{N-1} \left(\alpha + v_i^2 + \omega_i^2\right) \frac{t_f}{N} && (13)\\
\text{subject to} \quad & x_0 = 0, \quad y_0 = 0, \quad \theta_0 = \pi/2 && (14)\\
& x_N = 5, \quad y_N = 5, \quad \theta_N = \pi/2 && (15)\\
& x_{i+1} = x_i + v_i \cos(\theta_i) \frac{t_f}{N}, \quad i = 0, \ldots, N-1 && (16)\\
& y_{i+1} = y_i + v_i \sin(\theta_i) \frac{t_f}{N}, \quad i = 0, \ldots, N-1 && (17)\\
& \theta_{i+1} = \theta_i + \omega_i \frac{t_f}{N}, \quad i = 0, \ldots, N-1 && (18)\\
& \sqrt{(x_i - 2.5)^2 + (y_i - 2.5)^2} \geq 0.4, \quad i = 0, \ldots, N && (19)\\
& t_f > 0 && (20)
\end{aligned}$$

(ii) **Implementation Approach**

The implementation of the trajectory optimization uses the direct transcription method with `scipy.optimize.mi`
The key components are:

1. Parameterization: The trajectory is discretized into N time steps. The decision variables include the state at each time step (x_i, y_i, θ_i) and controls (v_i, ω_i), along with the final time t_f.

2. Objective Function: The cost is computed as:

$$J = \sum_{i=0}^{N-1} \left(\alpha + v_i^2 + \omega_i^2\right) \Delta t \tag{21}$$

3. Constraint Handling: Constraints are implemented as:

- Equality constraints for boundary conditions (initial and final states)
- Equality constraints for dynamics (discretized using Euler integration)
- Inequality constraints for collision avoidance at each discretization point

4. Optimization: The `scipy.optimize.minimize` function with the SLSQP (Sequential Least Squares Programming) method is used to solve the constrained nonlinear optimization problem. This method handles both equality and inequality constraints efficiently.

5. Initial Guess: A good initial guess is crucial for convergence. Typically, a straight-line path in configuration space with constant controls provides a reasonable starting point.

The optimized trajectory successfully navigates from the start to the goal while avoiding the obstacle at $(2.5, 2.5)$ with radius 0.3.

[Include trajectory plot here from the notebook]

(iii) **Effect of Different α Values**

The parameter α in the cost function $J = \int_0^{t_f} (\alpha + v^2 + \omega^2) dt$ balances time optimality versus control effort:

Small α (e.g., $\alpha = 0.1$):

- The optimizer prioritizes minimizing control effort $(v^2 + \omega^2)$ over time
- Results in slower, smoother trajectories with smaller velocities and angular rates
- The robot takes longer to reach the goal but uses less aggressive control actions
- The trajectory tends to follow a more conservative path around the obstacle

Medium α (e.g., $\alpha = 1.0$):

- Provides a balanced trade-off between time and control effort
- The trajectory completes in moderate time with reasonable control magnitudes
- Represents a practical compromise for real robot systems

Large α (e.g., $\alpha = 10.0$):

- The optimizer strongly prioritizes minimizing time t_f
- Results in faster trajectories with larger control inputs
- The robot reaches the goal quickly but with more aggressive maneuvers
- May lead to higher velocities and sharper turns around the obstacle
- The trajectory becomes closer to minimum-time control

Summary: As α increases, the weight on time increases relative to control effort, leading to faster but more aggressive trajectories. Conversely, smaller α values produce slower, smoother paths that conserve energy. The choice of α should reflect the specific application requirements: use larger values when time is critical, and smaller values when smooth, energy-efficient motion is preferred.

Problem 2

(i)

(ii)

(iii)

Problem 3

(i)

(ii)

(iii)

(iv)

Problem 4

Note: This problem is not graded but should be completed for section preparation.

(i)

(ii)

Appendix A: Code Submission Template

PASTE CODE HERE