Assignment 02: Path Tracking for a Robot Manipulator

Elisa Alboni* - elisa.alboni@unitn.it

November 2025

1 Introduction

This assignment focuses on the problem of path tracking for a robot manipulator. The goals of this assignment are:

- Understand how to make the robot's end-effector follow a specified path in space, using optimization-based control;
- Compare path tracking and trajectory tracking for following a given path.

2 Submission procedure

You are encouraged to work on the assignments in groups of 2 people. **If you have a good reason to work alone, then you can do it, but this has to be previously validated by one of the instructors**. Groups of more than 2 people are not allowed. The mark of each assignment contributes to 10% of you final mark for the class (i.e. 3 points out of 30).

When you are done with the assignment, please submit a single compressed file (e.g., zip). **The file name should contain the ID number of the group members**, and it must contain:

- A pdf file with the answers to the questions, the **names and ID number** of the group members; you are encouraged to include plots and/or numerical values obtained through simulations to support your answers. **This pdf does not need to be long. One or two pages of text should be enough to answer the questions. You can then add other pages for plots and tables.**
- The complete orc folder containing all the python code that you have developed.

If you are working in a group (i.e., 2 people) only one of you has to submit.

Submitting the pdf file without the code is not allowed and would result in zero points. Your code should be consistent with your answers (i.e. it should be possible to produce the results that motivated your answers using the code that you submitted). If your code does not even run, then your mark will be zero, so make sure to submit a correct code.

*Optimization and Learning for Robot Control, Industrial Engineering Department, University of Trento.

3 Path Tracking for a Robot Manipulator Description

3.1 Path Tracking vs Trajectory Tracking

In trajectory tracking, the desired end-effector position is known at each time instant, so the controller can directly penalize the deviation from the reference trajectory. In contrast, in path tracking, the specific location on the path to follow at a given time is not known a priori.

We assume that the target path is specified as a function of a scalar variable $s \in [0, 1]$. For example, if the path has the shape of an infinity symbol with center c and radius r:

$$p_x(s) = c_x + r\cos(2\pi s)$$

$$p_y(s) = c_y + 0.5\, r \sin(4\pi s)$$

A naive approach to track a path would be to minimize the distance to the path, but this has two major drawbacks:

- computing the distance to an arbitrary path may be complex;
- remaining stationary on the path would be an optimal (but undesirable) solution.

3.2 Introducing a Path Parameter as Decision Variable

To address these issues, we introduce an additional trajectory variable $s(t)$, representing progress along the path. This variable evolves according to single-integrator dynamics, controlled by an auxiliary control input $w(t)$:

$$s_{k+1} = s_k + \Delta t\, w_k$$

We can then ensure the end-effector follows the path by imposing:

$$y(q) = p(s)$$

where

$$p_x(s) = c_x + r\cos(2\pi s), \quad p_y(s) = c_y + 0.5\, r \sin(4\pi s), \quad p_z(s) = c_z$$

and $y(q)$ denotes the end-effector position. To ensure consistent progress along the path, we can either maximize the terminal value of s in the cost function, or impose a terminal constraint:

$$s_N = 1$$

3.3 Modeling a Robot Manipulator

The state of the robot manipulator consists of its joint positions and velocities:

$$x = (q, \dot{q})$$

The control inputs are the joint accelerations $\ddot{q}$, which leads to the following double-integrator dynamics:

$$\ddot{q} = u$$

The torque bounds are nonlinear constraints depending on both x and u, since joint torques are obtained from the inverse dynamics:

$$\tau_{\min} \leq \underbrace{M(q)u + h(x)}_{\text{inv\_dyn}(q,\dot{q},\ddot{q})} \leq \tau_{\max}$$

3.4 Problem Formulation

The optimization problem minimizes a cost function that combines:

- a penalty on joint velocities;
- a penalty on joint accelerations;
- a penalty on the auxiliary input w.

The constraints ensure consistency with robot and path dynamics, the satisfaction of state and control limits, the progress along the path, and the tracking of the desired path.

$$
\begin{aligned}
\min_{X,U,S,W} \quad & \sum_{k=0}^{N-1} \left(w_v \|\dot{q}_k\|^2 + w_a \|u_k\|^2 + w_w \|w_k\|^2 \right) \\
\text{subject to} \quad & q_{k+1} = q_k + \Delta t \, \dot{q}_k, \quad \dot{q}_{k+1} = \dot{q}_k + \Delta t \, u_k, \quad \forall k \in [0, N-1] \\
& q^{\min} \leq q_k \leq q^{\max}, \quad -\dot{q}^{\max} \leq \dot{q}_k \leq \dot{q}^{\max}, \quad \forall k \in [1, N] \\
& \tau_{\min} \leq \text{inv\_dyn}(q_k, \dot{q}_k, u_k) \leq \tau_{\max}, \quad \forall k \in [0, N-1] \\
& s_{k+1} = s_k + \Delta t \, w_k, \quad \forall k \in [0, N-1] \\
& y(q_k) = p(s_k), \quad \forall k \in [1, N] \\
& x_0 = x_{\text{init}} \\
& s_0 = 0, \quad s_N = 1
\end{aligned}
$$

4 Questions

The template code for the assignment is implemented in `A2_template.py`

- Implement the cost and constraint functions required for path tracking

  `define_running_cost_and_dynamics():`
  `define_terminal_cost_and_constraints():`

  Do not change the tasks' weights.
  Report and discuss the resulting trajectories.

- Let us assume that the task must be performed *cyclically* for a certain number of repetitions (for instance, to execute a cutting operation that requires multiple passes). Rather than optimizing the entire duration of the complete task, one can optimize a single execution and then repeat it as many times as needed. To ensure that the same control trajectory can be applied repeatedly in a cyclic manner, the robot must return to its initial configuration at the end of each execution.

  – Implement a terminal cost that penalizes the distance between the initial and final state with a weight $w_{final} = 1$.
  – Report and discuss the resulting trajectories.

- **Comparison with trajectory tracking.** Modify the "*cyclic*" (with final cost) formulation to perform trajectory tracking instead of path tracking where the path-dynamics are fixed (i.e., $w(t) = 1/N$). Hint: remove the equality constraint enforcing the coincidence between s and the end-effector position and introduce a quadratic penalty term with weight $w_p = 10^2$ to track the trajectory.
  - Why is the trajectory tracking achieved using a cost instead of a constraint? Would the same results be obtained if constraints (instead of costs) were used?
  - Report and discuss the resulting trajectories (highlighting the difference between these and the one obtained via path tracking)
- **Bonus question: Minimum-time formulation.** Extend both the path tracking and trajectory tracking problems to include a *minimum-time* objective by treating the time step δt as a bounded decision variable $\delta t \in [0.001, 0.1]$ and adding the running cost $10^{-1}\delta t^2$. For this task, assign $w_p, w_v, w_a, w_w, w_{final} = 10^2, 10^{-6}, 10^{-6}, 10^{-6}, 10^{-6}$. Add
  `"ipopt.hessian_approximation":"limited-memory"`
  to `opts` when solving the path tracking problem.
  Report and compare the results.