Optimal control of a SCARA robot (Project 2)

ROBT615 Optimal Control and Planning
ROBT703 Advanced Optimal Control and Planning

Dr. Ton Duc Do

1 System and task description

The objective is to solve an optimal control problem for the same robot as in Project 1, i.e., a SCARA robot that has the same structure of the FANUC SR-3iA shown in Fig. 1, although with slightly different parameters in terms of link lengths and limits on the angular displacements of the links. Our objective is to

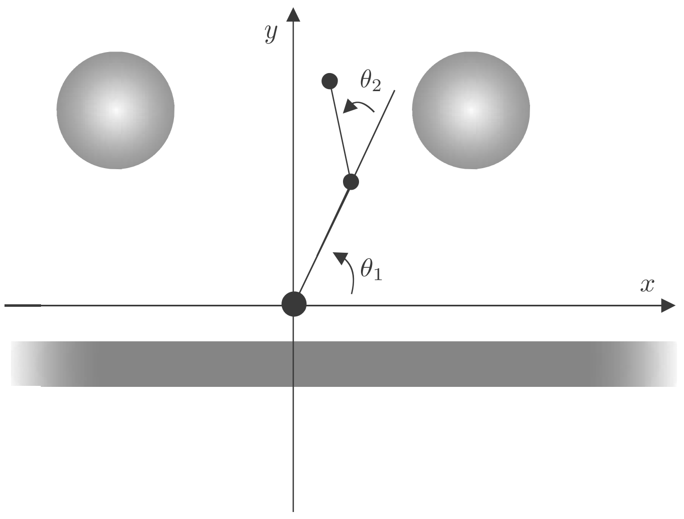

Figure 1: FANUC SR-3iA (left) and schematic of the robot and obstacles modeling for a reduced 2R configuration (right).

pick up an object at a given position in space, and place it at a different position, using an electromagnet at the end effector, which can be activated or deactivated at will. The vertical motion of the robot can happen at any time without affecting collisions with obstacles: as a consequence, the vertical motion is planned independently, and we only have to determine the motion of the manipulator in the horizontal plane. The robot can thus be modeled as a simple 2R planar manipulator. The robot base (i.e., where link 1 is fixed to the ground) is at $(x, y) = (0, 0)$. The links have lengths $\ell_1 = 0.5$ m and $\ell_2 = 0.4$ m, respectively. The first obstacle to be avoided is a wall, which runs parallel to the x axis, keeping a distance of 0.1 m from it (see Fig. 1). Also, there are two other obstacles: these have a fixed position and have been conservatively represented by two circles, both with radius $B = 0.2$ m, and with center at $(x_{c1}, y_{c1}) = (-0.6, 0.7)$ m and $(x_{c2}, y_{c2}) = (0.6, 0.7)$ m, respectively. The thickness of the links can be neglected, as the sizes of all obstacles have been already augmented to account for robot link thickness. The angular motion of link 1 is only limited by the presence of the wall (so no additional constraints have to be inserted), while link 2 can only move within a range of $\pm 90^\circ$ with respect to the configuration in which it is perfectly aligned with link 1 (i.e., $\theta_2 \in [-\pi/2, \pi/2]$). Our task is to plan a motion from a specific initial configuration (where the object is picked) to another configuration (where the object is placed), chosen in the free space, avoiding any collision during the robot motion.

2 Robot dynamics, free space, and obstacle space

First of all, we have to represent the robot configuration space, the configuration being $q = (\theta_1, \theta_2)$, and coinciding with the system state $x = (x_1, x_2)$. We assume to be able to impose the speed of each link as a component of the input vector $u = (u_1, u_2)$, thus obtaining a system dynamics given by

$$\dot{x}_1 = u_1$$
$$\dot{x}_2 = u_2$$

Regarding collisions, from a visual inspection, we notice that link 1 can collide with the wall, but not with any of the circular obstacles: as a consequence, there is no need to define spheres around link 1. As for link 2, this will instead be necessary: you can choose an arbitrary number of spheres, but the suggestion is to use at least 3 of them to cover the whole link length.

The motion has to take place in $T = 5$ s, with the imposed initial and final configurations chosen as

$$x(0) = \begin{bmatrix} 0 \\ \pi/6 \end{bmatrix}, \; x(T) = \begin{bmatrix} 5\pi/6 \\ 0 \end{bmatrix}.$$

During the robot motion, we need to avoid the obstacles, and guarantee that the joint angles are within the given limits. No additional limits are imposed on the joint speeds.

3 Optimal control problem

With the given path and boundary constraints, your first task is to formulate a continuous-time OCP (on paper) so as to minimize a cost function (Lagrange term only) defined to limit the joint speeds, which are not constrained, as

$$J(u(t), x(t)) = \int_0^T \left(u_1^2(t) + u_2^2(t)\right) dt.$$

In order to solve the OCP, you first need to discretize the system dynamics. To do that, first you have to choose the number N of samples in the 5s-interval between 25 and 50, thus obtaining a discretization interval $\Delta t \in [0.1, 0.2]$ s. In your code, leave N as a parameter that you can easily change without rewriting the whole code. As this is a linear system, we can use exact discretization, and obtain the discrete-time system

$$x^+ = f_d(x, u) = x + \Delta t \cdot u$$

which defines $x^+ = x_{k+1}$ given $(x, u) = (x_k, u_k)$. After defining the discretized dynamics, formulate the whole OCP in discrete time. In MATLAB, express it as an NLP to be solved with `fmincon`, by using the sequential approach. To do that, you have to express all values of x_k, $k = 1, \dots, N$, as functions of (x_0, U)

via forward simulation, so that the equality constraints defining the system dynamics will not appear in the OCP, and your only decision variables will be x_0 (constrained at the given value) and the control sequence $U = \{u_0, u_1, \ldots, u_{N-1}\}$.

The `fmincon` function might not be able to find a solution: therefore, look at the `help` for this MATLAB function, try to change the initial guess for the decision variables, the number of steps N, and the used algorithm (`'interior-point'` or `'sqp'`).

At the end, plot the time evolution of input and state variables. Only the input sequence is given by the solver by using the sequential approach, but you can obtain the state sequence running one forward simulation. From these plots, verify that all constraints are satisfied. To verify the satisfaction of collision (path) constraints, generate a figure where you plot the two robot links for all the considered time instants (in the 2D space represented in the right half of Fig. 1) and you verify that no collisions occur. Alternatively, you can generate a simple video of the robot motion, which shows the absence of collisions. Both solutions are accepted, and will give you full score if properly made.

4 Program output and grading

As for the first project, your program overall can be composed of an arbitrary number of scripts and functions, which should be properly commented (including the comments at the beginning of each script/function, which define how it is used). Name `main.m` the main script to be run. The overall program, when run, has to generate the following figures:

1. time evolution of input variables;
2. time evolution of state variables;
3. figure to check the absence of collisions.

You have to submit the Matlab files together as a zip file, plus a pdf file containing a clear formulation of the continuous-time OCP and its discretization. The grading will be determined based on how well the following tasks are executed: description of continuous-time and discrete-time OCPs on paper (20%), formulation of the OCP with MATLAB (40%), generated figures (and video, if present) (20%), overall clarity of the code (20%).