Assignment III
Robotics for IEM

October 2023

1 General Notice

1. Important Dates: This assignments starts on **12/10/2023** and ends at **26/10/2023**. This means you should hand in your assignment on *BRIGHTSPACE* no later than 23:59 at **26/10/2023**.
2. You need to hand in a mini-report (made in LaTeX) along with code files. The mini-report has to be brief, clear and easy to read. **We cannot grade what we cannot read.** In this mini-report, you only need to write down the answers to the questions below. In particular, you should write down how you get your answers (e.g., the computation process) rather than only the final results. You can include figures of your drawings in the mini-report, for instance for drawing frame assignments etc.

   Code template files will be provided. These template files provide frameworks and you only need to fill in what is required. DO NOT change the frameworks of the codes (e.g., the format of the inputs of a function).
3. Your mini-report will be graded by the TAs, but the code files will be graded automatically by Matlab scripts. This means that it is your responsibility to make sure that your codes run successfully without reporting any errors. **Unrunable (erroneous) codes and incorrect answers will be graded ZERO by the automatic scripts.** Incorrect answers are those of which the differences from the correct answers exceed some manually set threshold.
4. To avoid version conflicts, you are encouraged to use **MATLAB (2022 or higher)** in UWP, provided by the university, although other versions should work. The recommended robotics toolbox by Peter Corke is **RTB10.4.** A manual named RoboticsToolboxManual about the toolbox is available on *BRIGHTSPACE*.

2 Hand in Materials for this Assignment

Each student has to hand in a zipped folder on the corresponding assignment page on *BRIGHTSPACE*. The name of the this zipped folder should be according to the syntax *SNumber_Assignmenty.zip*. For example, if S123456 is submitting the first assignment, then the zipped folder should be named *S123456_Assignment1.zip*. The contents of the zipped folder is listed below.

- PDF file of the mini-report made in LaTeXwith the answers (and derivations) to the questions in Section 3 of this document.
- MATLAB Code files necessary for this specific assignment, being `traj_plan.m` and `resolved_rate.m`

3 Questions

Throughout this assignment, we will use the lightweight, low-cost robotic manipulator AX-18A SMART ROBOTIC ARM, as shown in Figure 1.

Figure 1: AX-18A SMART ROBOTIC ARM

For this assignment, use the Robotics Toolbox version 10.4, which you can download here (choose RTB10.4.mltbx). You will use the robot arm model pArb (enclosed with this assignment). The needed functionalities of the Toolbox are described in the manual downloadable from BrightSpace.

Question 1 - Trajectory Planning (40%)

1. [20%] Fill in the provided incomplete Matlab function `traj_plan.m`. Specifically, given the starting pose of the end-effector `pose_start` and the final one `pose_stop`, plan a trajectory to move the end-effector from the starting pose to the final pose **in the joint space**. Create a Matlab figure to show the trajectory of the end-effector in the Cartesian space (i.e., in the x-y-z space) using a red dashed line, and in the same figure, plot the arm performing the movement using the `pArb.plot` function. In a separate figure, plot the trajectories of the joint variables, **put it in the mini-report**. (Hint: You may want to use the function `pArb.ikine` with a suitable `"mask"` option for the inverse kinematics (we do not care about orientation). Try help `pArb.ikine` in Matlab to check how to use this function. You may also want to use the function `jtraj`.)

2. [20%] Continue filling in the provided incomplete Matlab function `traj_plan.m`. Specifically, given the starting pose of the end-effector `pose_start` and the final one `pose_stop`, plan a trajectory for the end-effector that is straight **in the Cartesian space**. Using the SAME figure as the previous sub-question to show the trajectory of the end-effector in the Cartesian space using a blue solid line, and in the same figure, plot the arm performing the movement using the `pArb.plot` function. In a separate figure, plot the trajectories of the joint variables, **put it in the mini-report.** (Hint: You may also want to use the function `ctraj`).

Question 2 - Jacobian and Trajectory Planning (60%)

Resolvend-rate motion control ses the following relation.

$$vJ(q)\dot{q} \Leftrightarrow \dot{q} = J^{\dagger}(q)v, \tag{1}$$

where $J^{\dagger}(q) = (J^{\top}J)^{-1}J^{\top}$ is a pseudo-inverse of the Jacobian $J(q)$ (use the Matlab command `pinv(J)` to compute it). It allows us to transform desired Cartesian velocity to joint velocity.

The resulting solution using the pseudo-inverse minimizes the least-squares error of the joint-state. The motion control scheme is typically implemented in discrete-time form as

$$q_{\text{new}} = q_{\text{old}} + K dt \cdot J^{\dagger}(q_{\text{old}})v, \tag{2}$$

where dt is a small enough time step and $K > 0$ is the gain of the controller. Based on this approach, we wish to develop a resolved-rate motion controller that will lead to **approximately straight line** motion between two points.

It is important to notice that in this case **the trajectory cannot be computed all at once and then fed to the arm**, but the next position has to be computed in real time, since future configuration of the joints q_{new} is dependent on the actual configuration q_{old} and the desired straight line trajectory. Therefore, this type of feedback is implemented. A rough idea of the control scheme that has to be implemented can be seen in Figure 2.

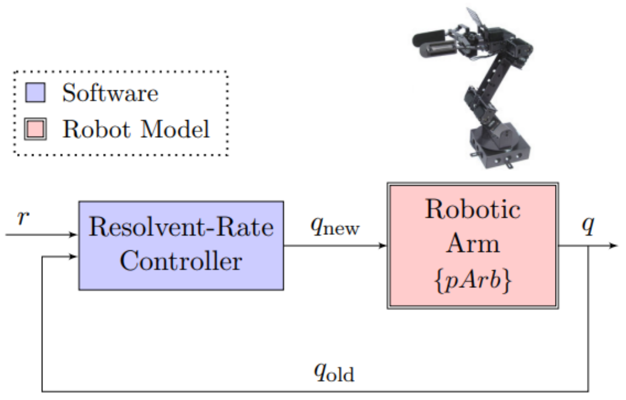

Figure 2: Idea of the control scheme that has to be implemented to apply the resolvent-rate controller to the robot arm.

Fill in the provided incomplete Matlab function resolved rate.m. We wish to move the end-effector in an approximately straight line from the initial joint configuration q0 to the final pose pose stop. The following hints/tasks can help you finish this question.

1. [40%] The motion will take n steps with the step size dt. At each time step, from 1 to n, compute the joint coordinates that the robot must reach using the resolved rate motion control algorithm. Notice that v has to change at each time instant such that it helps the robot move towards the goal point. (Hint: Use `pArb.jacob0(q)` for the Jacobian)

2. [20%] Save the trajectory points (Cartesian) as a matrix in the workspace variable. Create a Matlab figure (i) to show the trajectory of the end-effector in the Cartesian space using a blue solid line, and in the same figure, plot the arm performing the movement using the pArb.plot function. Also, create Matlab figures (ii) showing the position of the end-effector in x, y, z over the steps. In figures (ii), include 2 scenarios, one with a proper gain, and one with gain that is too high, **put both in the mini-report.**

Grading for Question 2 is done based on the correct trajectory computation ([40%]) and presenting the plots ([20%]).