TBROB-18/19

Preparatory Tasks

(50% of lab session grade)

TA: Carlo Cenedese (e-mail: c.cenedese@rug.nl)
Weijia Yao (e-mail: w.yao@rug.nl)

September 24, 2019

1 Introduction

The main goal of this lab session is to get more knowledge on differences between calculating on an ideal simple robot and working with a more complex real robot. Over this course we will use a lightweight, low-cost robotic manipulator AX-18A SMART ROBOTIC ARM.
The lab session contains 2 parts.

1. Preparatory questions. (This document)

2. Lab session exercises.

The weight of each part will be as following:

- Preparatory exercises (50%)
  A part of the correctness of the answers, the full mark will be reached only if:

  - the document carefully and concisely describes how the results are obtained;
  - the code is well organized and has the right amount of comments.

  Notice: if an answer is not clear some additional questions or explanations can be asked during the Lab session.

- Lab session exercises (50%)
  The main criterion of evaluation are:

  - how much time you will take to complete the lab assignments (all exercises must be done within **2 hours** of the lab session to get full score). Maximal extension of half an hour might be possible upon request.
  - working attitude (your independence and preparation for the experience will be considered)

Along all the tasks it is required to know Matlab and some commands of the Robotics Toolbox developed by Peter Corke, that you can download here. The needed functionalities of the Toolbox are described in the manual downloadable from the course portal in Nestor.

Hand In the preparatory questions (Deadline: midnight of 09/10/2019)

In order to complete the preparatory tasks every group of 2 students has to send an email to **RUG.TBROB1819@gmail.com** attaching a zipped folder, named using the following syntax
`ROBPrepTasks_GroupNumber_Name1Surname1_Name2Surname2`,
e.g., `ROBPrepTasks_7_KevinSpacey_MorganFreeman` .
The folder must contain:

- A **PDF** file with the answers and explanations to the tasks below. The pdf must be created via LaTeX, Word or equivalent editors[1]. **NO handwritten document will be accepted.**
- The three Matlab scripts introduced in the five task, i.e. `arm_creation_val.m`, `fk.m`, `ik.m` and `traj_planning.m` and `lab_session.m`.

We will release the correct inverse kinematics and forward kinematics answers after the deadline to submit your work. These answers are very helpful for you to finish your lab session exercises. Please check you answer and prepare well before the lab session.

The preparatory homework will be graded during the lab sessions. To examine your understanding of the preparatory tasks, you are expected to answer some questions asked by the TAs. This is also part of the grading.

2 Assignments

2.1 Basic Assignments

1. (15 points) FORWARD KINEMATICS

   (a) Determine the Denavit-Hartenberg (DH) Parameters for the AX-18A SMART ROBOTIC ARM (in Fig. 2) using the following parameters:
   $L_1 = 17$ cm, $L_2 = 17$ cm, $L_3 = 7$ cm, $L_4 = 4$ cm, $L_5 = 4$ cm, $L_6 = 9$ cm.

   (b) Derive the neighboring homogeneous matrices ${}^{i-1}_{i}T, i = 1, ..., 6$ and ${}^{0}p_6$.

2. (15 points) INVERSE KINEMATICS

[1]Because of the high number of mathematical formulas that you will write, we suggest to adopt LaTeX.

Assuming that $\theta_4 \equiv 0$ and $\theta_5 \equiv 0$ analytically derive the inverse-position solution for this robot. For the given position of the end-effector i.e the fourth column of ${}^0_6T$ you have to find all possible configurations of $\theta_i, i = 1, 2, 3$

3. (10 points) JACOBIANS

   (a) Derive the basic Jacobian relating joint velocities to the end-effector's linear and angular velocities in frame X_0, Y_0, Z_0.

   (b) Find the singularities. For each type of singularity that you found in the previous part explain the physical interpretation of the singularity, by sketching the arm in a singular configuration and describing the resulting limitation on its movement (the pictures can be also done by hand and then scanned, as far as they are easily comprehensible).

4. CODING TASKS (10 points)

   Note: IF YOUR CODES FAIL TO RUN IN MATLAB, YOU WILL GET A ZERO GRADE.

   (a) Using the DH parameters and the Robotics Toolbox, create the model of the robotic arm (name the arm `pArb`). Then plot it in the zero (like in Fig. 1) and in a random configuration (*Hint*: to modellize the gripper try the command `pArb.tool`).

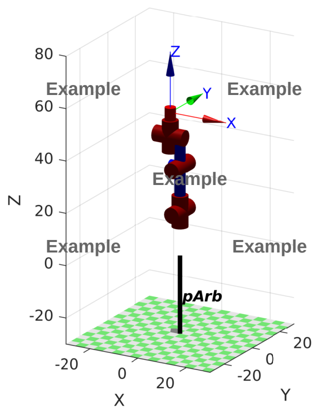

Figure 1: Plot of `pArb` in the zero configuration.

   (b) Use `pArb` to validate your answers to 1(b) and 3(a). In particular, show that, given a random configuration q_r, the value of ${}^0p_6$ (respectively of J_0) obtained by `pArb` and your answer of

1(b) (respectively 3(a)) evaluated in q_r give the same results (*Hint:* you can try to use the commands `fkine` and `jacob0`). Organize <u>all</u> the code used for 4(a) and 4(b) in a Matlab script `arm_creation_val.m`, when executed it has to display only the relevant calculations results and plots.

(c) Use Matlab to create your custom functions[2] to calculate the forward and the inverse kinematics for this arm (the latter under the assumptions done in Task 2) .

  i. The forward kinematics function has to meet the following specifics:

  - *Input:* a 5×1 vector `q`, that represent the configuration of the arm.
  - *Output:* a 3×1 vector `p` and a 3×3 matrix `R`, respectively the translational and rotational part of the transformation matrix from the base to the end-effector.
  - *Syntax:* The function has to follow the syntax

```
[p,R] = fk (q)
```

  ii. The inverse kinematics function instead should fulfil the next requirements:

  - *Input:* a 3×1 vector `p`, the sought position of the end-effector.
  - *Output:* a 5×1 vector `q`, that represent the configuration of the arm leading to the position `p` of the end-effector.
  - *Syntax:* The function has to follow the syntax

```
q = ik (p)
```

  (*Hint:* because of the assumption done during the calculation of the inverse kinematics, some of the components of the output will have a "special" value)

(d) Chose a starting pose of the end effector $\mathtt{T}_{\mathtt{start}}$ and a final one $\mathtt{T}_{\mathtt{stop}}$, then plan the trajectory to move the end-effector from the first to the last one **in the join space**. Plot the arm performing the movement and the graph of the movement of the end-effector in the Cartesian space.

[2]This implies that you cannot use the Robotics Toolbox for this task.

(e) Chose two suitable points in the Cartesian space space $\texttt{p}_{\texttt{start}}$ and $\texttt{p}_{\texttt{stop}}$ and plan a trajectory for the end-effector that is straight **in the Cartesian space**. Also in this case, plot the arm performing the movement and the graph of the movement of the end-effector in the Cartesian space. Which are the limitations that you face in this case and how you can overcome them (explain your choices) ? (*Hint:* Try to use the `ikine` function with a suitable "mask" option).

Organize the code of this point and of the previous one in a Matlab script and name it `traj_planning.m`.

2.2 Lab-session-related Assignment

Joint-space Controller

The main goal of the joint space control is to design a feedback controller such that the joint coordinates track the desired motion as closely as possible. Firstly, the desired motion, which is described in terms of end-effector coordinates, is converted to a corresponding joint trajectory using the inverse kinematics of the manipulator. Then the feedback controller determines the new set-points for joints local controllers to move the manipulator along the desired trajectory specified in joint coordinates starting from measurements of the current joint states. Due to its simple structure, this kind of control schemes offers many advantages. For example, by using independent-joint control, communication among different joints is saved. Moreover, since the computational load of controllers may be reduced, only low-cost hardware is required in actual implementations. Finally, independent-joint control has the feature of scalability, since the controllers on all the joints have the same formulation.

Please finish the following exercises with the simulated robot arm that you have created in Subsection 2.1.

(a) Define two positions in the workspace $p_a = [x_a, y_a, z_a]^\top$ and $p_b = [x_b, y_b, z_b]^\top$ and determine the appropriate configurations of joints. (*Hint:* try the command `pArb.ikine()` with a "mask" and the custom function `ik.m` built above)

(b) Plan a smooth trajectory between these points. (*Hint:* try the command `jtraj(.)`)

(c) Save the trajectory points as a matrix in the workspace variables and check your result based on mathematical model of the robot. Plot the resulting trajectory of the end-effector in the Cartesian space and the one of each joint in the joint space (*Hint:* try the command `pArb.fkine(.)`).

Organize the code in a matlab file called `lab_assignments.m`, that can be immediately run to check the three points above.
In the **lab session**, we will discuss the above three exercises (a), (b), (c). Then you need to apply this trajectory on the real robotic arm.

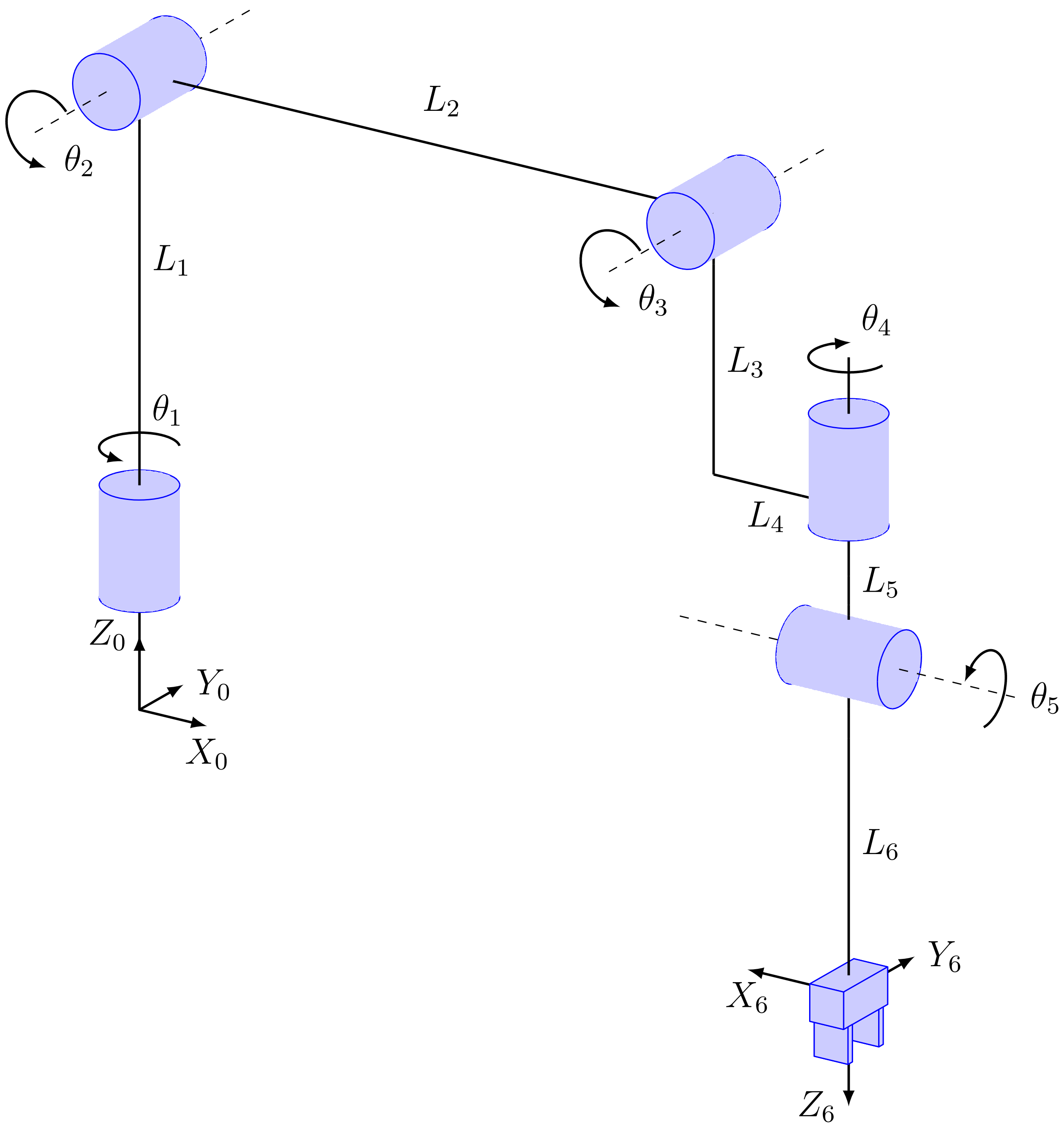

Figure 2: Schematic of the AX-18A SMART ROBOTIC ARM.
This configuration corresponds to the following joint set:
$\theta_1 = 0, \theta_2 = -90°, \theta_3 = -90°, \theta_4 = 0, \theta_5 = 0,$