ROBOT MODELING AND CONTROL

ECE470S

LAB1 : Forward and Inverse Kinematics for the PUMA 560 Manipulator

1 Purpose

The purpose of this lab is to derive the forward and inverse kinematics equations for the PUMA 560 robot, to implement these equations as Matlab functions, and to use these functions to perform basic computer simulations.

2 Introduction

The PUMA (Programmable Universal Machine for Assembly) is a robot developed by Unimation in the late 1970s for industrial automation in General Motors plants. This robot was particularly well-designed, mechanically robust, and very reliable, so much so that it remained in production for twenty years even after Unimation was bought out. In this lab you will familiarize yourself with the kinematics of the PUMA 560, depicted in Figure 1. This is an articulated robot with six joints, the last three of which are a spherical wrist.

Using the Denavit-Hartenberg (DH) convention, you will assign six frames to the six moving links of the robot, and you will compile the associated DH table. The i-th row of the DH table contains four parameters associated with link i:

- a_i: link length. This is the length of link i, measured along the common normal to axes z_{i-1} and z_i.
- α_i: link twist. This is the angle from axis z_{i-1} to axis z_i, measured through a rotation about axis x_i. The sign of α_i is determined by applying the right-hand rule around axis x_i.
- d_i: link offset. This is the distance between the origins of frames $i-1$ and i, measured along axis z_{i-1}.
- θ_i: joint angle. This is the angle between axis x_{i-1} and axis x_i, measured using the right-hand rule around axis z_{i-1}.

2.1 Forward Kinematics

With the DH table, we can compute the homogeneous transformation matrix from frame $i-1$ to frame i as

$$H_i^{i-1} = \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Composition of these matrices gives the homogeneous transformation matrix from the base frame to link 6:

$$H_6^0 = \begin{bmatrix} R_6^0 & o_6^0 \\ 0 & 1 \end{bmatrix} = H_1^0 \cdot \ldots \cdot H_6^5.$$

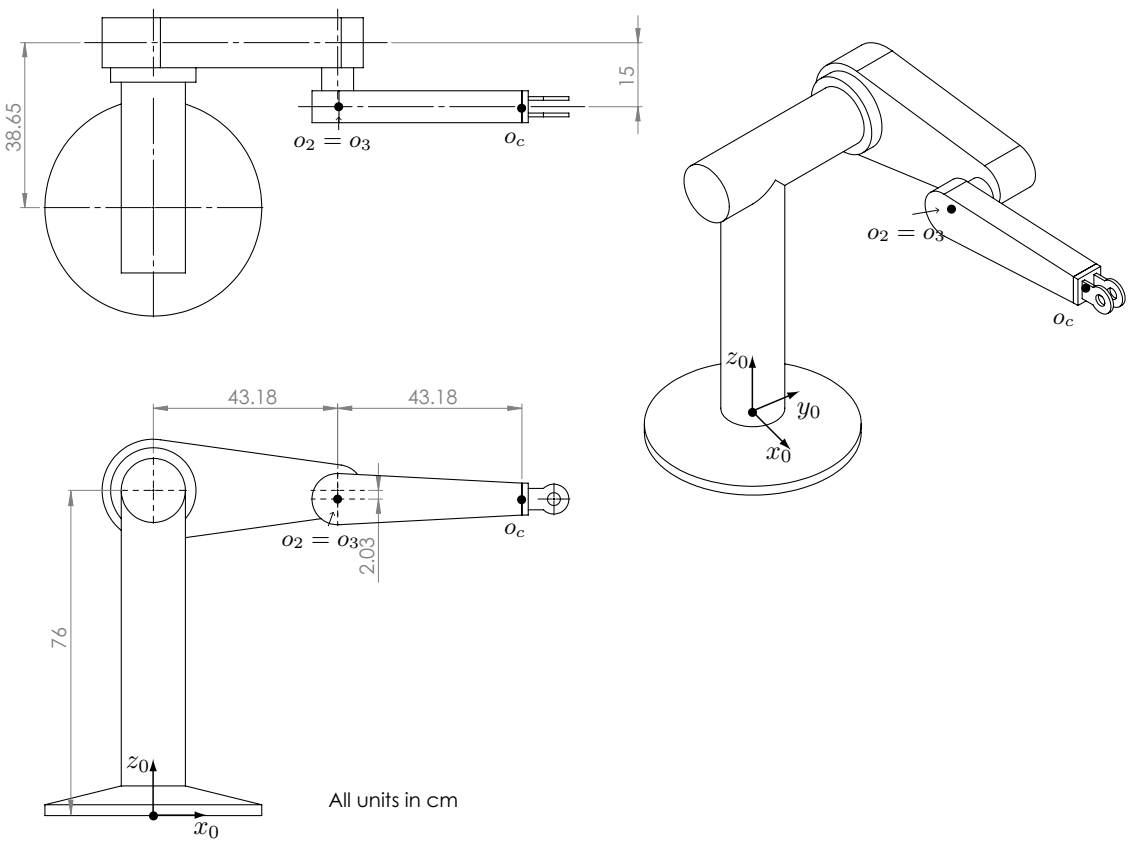

Figure 1: Schematics of the PUMA560 robot

By extracting the 3×3 matrix R_6^0 and the 3×1 vector o_6^0 from H_6^0, the forward kinematics computation is complete. Recall the geometric meaning of R_6^0 and o_6^0: the columns of R_6^0 are the unit axes of the end effector frame 6 represented in the coordinates of frame 0: $R_6^0 = [x_6^0 \mid y_6^0 \mid z_6^0]$; the vector o_n^0 is the origin of frame 6 expressed in the coordinates of frame 0.

2.2 Inverse Kinematics

Given a desired position $o_d^0 \in \mathbb{R}^3$ and desired orientation $R_d \in \mathrm{SO}(3)$ of the end effector, the inverse kinematics problem is to find $(\theta_1, \ldots, \theta_6)$ such that $R_6^0(\theta_1, \ldots, \theta_6) = R_d$ and $o_6^0(\theta_1, \ldots, \theta_6) = o_d^0$.

Since the PUMA 560 has six links and a spherical wrist, one can solve the inverse kinematics problem by the technique of kinematic decoupling. In kinematic decoupling, the problem is divided in two parts: inverse position and inverse orientation.

Inverse position. The position of the wrist centre o_c only depends on the angles of the first three joints, $(\theta_1, \theta_2, \theta_3)$. The idea is to compute the desired location of the wrist centre and then find $(\theta_1, \theta_2, \theta_3)$ accordingly.

- Compute the vector

$$o_d^0 - R_d \begin{bmatrix} 0 \\ 0 \\ d_6 \end{bmatrix}.$$

- Find $(\theta_1, \theta_2, \theta_3)$ such that $o_c^0(\theta_1, \theta_2, \theta_3) = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = o_d^0 - R_d \begin{bmatrix} 0 \\ 0 \\ d_6 \end{bmatrix}$.

Inverse orientation. We need to impose that $R_6^0 = R_d$. Having computed $(\theta_1, \theta_2, \theta_3)$, we can compute the rotation matrix $R_3^0(\theta_1, \theta_2, \theta_3)$. Using the fact that $R_6^0 = R_3^0 R_6^3$, we need to impose that

$$R_3^0(\theta_1, \theta_2, \theta_3) R_6^3(\theta_4, \theta_5, \theta_6) = R_d.$$

Multiplying both sides by $(R_3^0)^\top$ we obtain

$$R_6^3(\theta_4, \theta_5, \theta_6) = (R_3^0)^\top R_d.$$

This equation must be solved for $(\theta_4, \theta_5, \theta_6)$. This is easy because, as we have seen in class, the angles $(\theta_4, \theta_5, \theta_6)$ are just the ZYZ Euler angle coordinates of the rotation matrix $(R_3^0)^\top R_d$ (when the x_4 axis is chosen as $x_4 = -z_3 \times z_4$).

In summary, the procedure to find $(\theta_4, \theta_5, \theta_6)$ is this.

- Compute $H_3^0(\theta_1, \theta_2, \theta_3) = H_1^0(\theta_1) H_2^1(\theta_2) H_3^2(\theta_3)$, and extract the rotation matrix R_3^0.
- Compute $(R_3^0)^\top R_d$.
- Find the ZYZ Euler angles coordinates (θ, ϕ, ψ) of $(R_3^0)^\top R_d$ by using the equations given in class, or equations (2.28), (2.30), (2.31) (or (2.29), (2.32), (2.33)) of the textbook, assuming that the x_4 axis has been chosen as $x_4 = -z_3 \times z_4$. Please note that the equations in the textbook use the convention $\mathsf{atan2(x, y)}$, whereas in Matlab (and in class), the convention $\mathsf{atan2(y, x)}$ is adopted. To avoid errors, refer to the equations presented in class.
- Set $\theta_4 = \phi$, $\theta_4 = \theta$, $\theta_6 = \psi$.

3 Preparation

1. Print out page 2 of this document and draw frames $0, 1, 2, 3$ on the three-dimensional schematic of the PUMA 560. Place frame 0 at the base of the robot. Assign frames 1,2,3 according to the DH convention. In particular, your assignment of frame 3 should be consistent with the presence of a spherical wrist (not represented in the figure), so that the z_3 axis should run parallel to the length of the last link. Place the origins of frames 2 and 3 in the location indicated on the PUMA schematic.
2. Based on your frame assignment, and the lengths marked on Figure 1, write down the 6×4 DH table of the PUMA 560 robot, assuming that the wrist length is $d_6 = 20$cm and that $x_4 = -z_3 \times z_4$.
3. Write down closed-form expressions of the inverse kinematics of the PUMA 560. Specifically, you should write $(\theta_1, \ldots, \theta_6)$ as functions of the entries R_d and o_d^0. Please note the position of the wrist centre o_c marked on the PUMA schematic.

4 Experiment

In this lab you will write Matlab functions implementing the forward and inverse kinematics of the PUMA 560. These functions will be used in all subsequent labs of this course, so it is important that you test them accurately. You will use the robotics toolbox to define and plot the robot configuration in three-space.

4.1 Definition of robot structure

Write a function `mypuma560.m` that defines a new robot structure with the PUMA 560 parameters you determined in your preparation. Your function must have these arguments

```
myrobot = mypuma560(DH)
```

where DH is a 6×4 matrix of DH parameters when the robot is in its rest position (i.e., when all θ_i's are zero). The i-th row of the matrix `DH` will contain the parameters $[\theta_i \quad d_i \quad a_i \quad \alpha_i]$ in this order.

To create the robot structure `myrobot`, your function should use the commands `Link` and `SerialLink` provided by the Robotics Toolbox. See the Robotics Toolbox help to see how these commands work. The Robotics Toolbox manual is the file `robot.pdf` posted on Blackboard.

4.2 Plot a sample joint space trajectory

Now you will generate a sequence of 200 joint angles in a 200×6 matrix q defined as follows: the i-th row of q contains the values of joint angles $\theta_1, \ldots, \theta_6$ at step i. The values of the joint angles should be selected as follows:

- θ_1 should transition from 0 to π in 200 steps.
- θ_2 should transition from 0 to $\pi/2$ in 200 steps.
- θ_3 should transition from 0 to π in 200 steps.
- θ_4 should transition from $\pi/4$ to $3\pi/4$ in 200 steps.
- θ_5 should transition from $-\pi/3$ to $\pi/3$ in 200 steps.
- θ_6 should transition from 0 to 2π in 200 steps.

To generate each row θ_i, use the Matlab command `theta=linspace(initial,final,N)`.

Plot the robot evolution corresponding to the joint angles above by issuing the command

```
plot(myrobot,q)
```

Observe the geometric representation of the robot. The `plot` command plots the minimum amount of information needed to represent the motion of the robot: namely, the origin of the coordinate frames and segments connecting them, but it does not represent detailed geometric structure of the robot such as the fact that some links are L-shaped.

4.3 Forward Kinematics

Develop a function `forward.m`

```
H = forward(joint,myrobot)
```

where:

- `joint` is a 6×1 vector containing the values of the six joint angles.
- `myrobot` is the robot structure generated by your function `mypuma560`.
- Your function must return `H`, the homogeneous transformation matrix $H_6^0 = A_1(\theta_1) \cdot \ldots \cdot A_6(\theta_6)$.

Note that `H` will encode information about the position and orientation of the end effector. Specifically, the orientation of the end effector with respect to the base frame is represented by the rotation matrix `H(1:3,1:3)`, while the coordinates of the origin of the end effector are given by `H(1:3,4)`.

As a test, consider again the matrix q defined in Section 4.2. Using the function `forward`, for each row of `q` compute the coordinates of the end effector. Store all these coordinates in a 200×3 matrix named `o`. Then, plot the trajectory $o(t)$ with red colour:

```
>> plot3(o(:,1),o(:,2),o(:,3),'r')
>> hold on
```

Now issue the command

```
>> plot(myrobot,q)
```

and verify that the origin of the end effector traces out precisely the red trajectory you created using your forward function.

4.4 Inverse Kinematics

Develop a function `inverse.m`

```
q = inverse(H,myrobot)
```

where:

- `H` is the 4×4 homogeneous transformation matrix containing the desired position and orientation of the end effector.
- `myrobot` is the robot structure generated by your function `mypuma560`.
- Your function must return `q`, the 1×6 vector of joint angles solving the inverse kinematics problem.

It's very important that you get this function working properly. You'll use it in lab 3. Test your function as follows: when `H` is given as (units are in metres)

```
H = [cos(pi/4) -sin(pi/4) 0 20; sin(pi/4) cos(pi/4) 0 23; 0 0 1 15; 0 0 0 1]
```

your `inverse` function should return

```
q = [ -0.0331 -1.0667 1.0283 3.1416 3.1032 0.8185]
```

Next, suppose we want the PUMA to pick up an object on the table at $(x, y, z) = (10, 23, 15)$ (units in centimetres), then transport the object to $(x, y, z) = (30, 30, 100)$. Meanwhile, we want the end effector to have a constant orientation defined by a constant matrix R. Keep in mind that the inverse kinematics are not unique. To overcome this, it may be useful to pick a solution (e.g. left-arm, elbow-up) and develop your inverse function with this convention. Note also that your solution contains the square root operator which might return a complex number with zero real part. Use the command `real(sqrt(...))` to avoid complications.

1. Define a 100×3 matrix `d` like this: every row of `o` contains a desired position of the end effector, (x, y, z). The sequence of end effector positions must start at $(10, 23, 15)$, end at $(30, 30, 100)$, traveling along a straight line.
2. Define a 3×3 matrix `R` to be the rotation matrix $R_{z,\pi/4}$. This matrix defines the desired constant orientation of the end effector throughout the pick-and-place operation.

3. For each row `d`, use your function `inverse` to solve the inverse kinematics problem (i.e., use `d(i,:)` and `R` defined earlier in the inverse kinematics problem). Store the result as a row of a matrix `q`. At the end of this iteration, you will have a 100×6 matrix q of joint angles.

4. Following the procedure of Section 4.3, plot the configuration of the PUMA corresponding to the sequence of joint angles you've just found, and verify that the end effector has constant orientation, and that its origin traces out the straight line from $(10, 23, 15)$ to $(30, 30, 100)$.

5 Submission

Each student will submit on Quercus a preparation report and a zipped folder containing your Matlab functions. Thoroughly comment your code. Your folder may contain intermediate Matlab functions, but calls to the main functions `forward` and `inverse` must work without any modification required. The main file in the folder should be called `Lab1.m`, and it should work out the various steps in Sections 4.1-4.4, clearly displaying the main steps.