UNIVERSITÀ DEGLI STUDI DI GENOVA

DIBRIS

DEPARTMENT OF COMPUTER SCIENCE AND TECHNOLOGY,
BIOENGINEERING, ROBOTICS AND SYSTEM ENGINEERING

MODELING AND CONTROL OF MANIPULATORS

Third Assignment

Jacobian Matrices and Inverse Kinematics

Author:

Balia Gian Marco
Salterini Filippo
Polese Carolina

Student ID:

s4398275
s4516129
s4862903

Professors:

Enrico Simetti
Giorgio Cannata

Tutors:

Andrea Tiranti
Luca Tarasi
George Kurshakov

December 20, 2024

Contents

Mathematical expression	Definition	MATLAB expression
$< w >$	World Coordinate Frame	w
${}^{a}_{b}R$	Rotation matrix of frame $< b >$ with respect to frame $< a >$	aRb
${}^{a}_{b}T$	Transformation matrix of frame $< b >$ with respect to frame $< a >$	aTb
${}^{a}O_{b}$	Vector defining frame $< b >$ wit respect to frame $< a >$	aOb

Table 1: Nomenclature Table

1 Assignment description

The third assignment of Modelling and Control of Manipulators focuses on Inverse Kinematics (IK) control of a robotic manipulator.

The third assignment consists of three exercises. You are asked to:

- Download the .zip file called MCM_assignment3.zip from the Aulaweb page of this course.
- Implement the code to solve the exercises on MATLAB by filling in the predefined files. In particular, you will find two different main files: *"ex1.m"* for the first exercise and *"ex2.m"* for the second exercise.
- Write a report motivating your answers, following the predefined format on this document.
- **Putting code in the report is not an explanation!**

1.1 Exercise 1

Given the geometric model of an industrial manipulator used in the previous assignment, you have to add a tool frame. The tool frame is rigidly attached to the robot end-effector according to the following specifications:

Use the following specifications ${}^{e}\eta_{t/e} = [0, 0, \pi/10], {}^{e}O_t = [0.2, 0, 0]^{\top}(cm)$ where ${}^{e}\eta_{t/e}$ represents the YPR values from end effector frame to tool frame.

To complete this task you should modify the class *geometricModel* by adding a new method called *getToolTransformWrtBase*

1.2 Exercise 2

Implement an inverse kinematic control loop to control the tool of the manipulator. You should be able to complete this exercise by using the MATLAB classes implemented for the previous assignment (*geometricModel*,*kinematicModel*), and also you need to implement a new class *cartesianControl* (see the template attached). The procedure can be split into the following phases

Q2.1 Compute the cartesian error between the robot end-effector frame ${}^{b}_{t}T$ and the goal frame ${}^{b}_{g}T$.

The goal frame must be defined knowing that:

- The goal position with respect to the base frame is ${}^{b}O_g = (-0.14, -0.85, 0.6)^{\top}(m)$
- The goal frame is rotated of ${}^{b}\eta_g = (-3.02, -0.40, -1.33)^{\top}(rad)$ around the y-axis of the base frame (inertial frame).

Q2.2 Compute the desired angular and linear reference velocities of the end-effector with respect to the base: ${}^{b}\nu^{*}_{t/0} = \begin{bmatrix} \kappa_a & 0 \\ 0 & \kappa_l \end{bmatrix} \cdot \begin{bmatrix} \omega^{*}_{t/0} \\ v^{*}_{t/0} \end{bmatrix}$, such that $\kappa_a = 0.8$,$\kappa_l = 0.8$ is the gain.

Q2.3 Compute the desired joint velocities $\dot{\bar{q}}$

Q2.4 Simulate the robot motion by implementing the function: *"KinematicSimulation()"* for integrating the joint velocities in time.

2 Exercise 1

From the previous assignment are implemented the same functions `geometricModel` and `kinematicModel` with some differences. In the `geometricModel` is added a function `getToolTransformWrtBase` that compute the tool transformation matrix with respect to the base frame ${}^{b}_{t}T ={}^{b}_{e} T \cdot {}^{e}_{t} T$. The transformation matrix ${}^{b}_{e}T$ is computed as in the previous assignment with `getTransformWrtBase`.

3 Exercise 2

The Jacobian matrix in `kinematicModel` due to the presents of the tool is

$$ {}^{b}_{t}J ={}^{b} \mathbb{S}_{t/e} \cdot{}^{b} J_{e/b} \tag{1}$$

where ${}^{b}J_{e/b}$ is the basic Jacobian matrix and ${}^{b}\mathbb{S}_{t/e} \in \mathbb{R}^{6\times6}$ is the rigid body Jacobian matrix

$$ {}^{b}\mathbb{S}_{t/e} = \begin{pmatrix} I_{3\times3} & O_{3\times3} \\ [{}^{b}\mathbf{r}_{t/e}\times]^{\top} & I_{3\times3} \end{pmatrix} $$

where ${}^{b}\mathbf{r}_{t/e} ={}^{b} \mathbf{r}_{t/b} -{}^{b} \mathbf{r}_{e/b}$.

3.1 Q2.1

From ${}^{b}\boldsymbol{\eta}_g$ it is computed ${}^{b}_{g}R = {}^{b}_{g} R_z(\boldsymbol{\eta}_{g,\mathbf{k}}) \cdot {}^{b}_{g} R_y(\boldsymbol{\eta}_{g,\mathbf{j}}) \cdot {}^{b}_{g} R_x(\boldsymbol{\eta}_{g,\mathbf{i}})$, checking if it is a proper rotation matrix. Then it is possible to obtain the transformation matrix

$$
{}^{b}_{g}T = \begin{pmatrix} {}^{b}_{g}R & {}^{b}\mathbf{O}_g \\ \mathbf{O}_{1\times 3} & 1 \end{pmatrix}
$$

It is implemented the new class `cartesianControl`, where is present a function `getCartesianReference` that given the ${}^{b}_{g}T$ computes the Cartesian error. This function at first compute ${}^{t}_{g}T = {}^{b}_{t}T^{-1} \cdot {}^{b}_{g} T$, and then obtain ${}^{t}\mathbf{h}_{g/t}$ and θ from ${}^{t}_{g}R$. The Cartesian error between the robot end-effector frame and the goal frame is

$$
{}^{b}\mathbf{e}_{t/g} = \begin{pmatrix} {}^{b}\boldsymbol{\rho}_{t/g} \\ {}^{b}\mathbf{r}_{t/g} \end{pmatrix} = \begin{pmatrix} {}^{b}_{t}R \cdot {}^{t}\mathbf{h}_{g/t} \cdot \theta \\ {}^{b}\mathbf{r}_{g/b} - {}^{b}\mathbf{r}_{t/b} \end{pmatrix}
$$

where ${}^{b}_{t}R$ is obtained by ${}^{b}_{t}T$ from the Section 2 with the function `getTransformWrtBase`.

3.2 Q2.2

Desired angular and linear reference velocities of the end-effector with respect to the base are computed as follow

$$
{}^{b}\boldsymbol{\nu}^{*}_{t/0} = \begin{bmatrix} \kappa_a & 0 \\ 0 & \kappa_l \end{bmatrix} \cdot {}^{b}\mathbf{e}_{t/g}
$$

where $\kappa_a = 0.8$, $\kappa_l = 0.8$ are the gain.

3.3 Q2.3

The desired joint velocities are computed with the following equation

$$
\dot{\mathbf{q}} = {}^{b}_{t} J^{\dagger} \cdot {}^{b}\boldsymbol{\nu}^{*}_{t/0}
$$

where ${}^{b}_{t}J^{\dagger}$ is the pseudoinverse of the Jacobian matrix computed with the Equation 1.

3.4 Q2.4

The simulation of the robot's motion is implemented in the function `KinematicSimulation`. It takes in input:

- the current robot configuration (vector of joint positions) $\mathbf{q}$;
- the joints velocity $\dot{\mathbf{q}}$;
- sample time $t_s = \frac{t_{\text{end}} - t_{\text{start}}}{\text{Number of samples}} = 0.1\,s$;
- lower joint bounds $\mathbf{q}_{\text{min}}$;
- upper joint bounds $\mathbf{q}_{\text{max}}$.

And update q_i, where i is the i-th joint, with the new value $q_i + \dot{q}_i \cdot t_s$, if it is not greater of $\mathbf{q}_{\text{max}}$ and lower than $\mathbf{q}_{\text{min}}$.
During the simulation the robot's tool moves directly to the goal, see Figure 2. This is coherent with the graph in Figure 3, indeed both angular and linear velocities are constant, this means that the tool configuration changes constantly (without variation of the velocities).

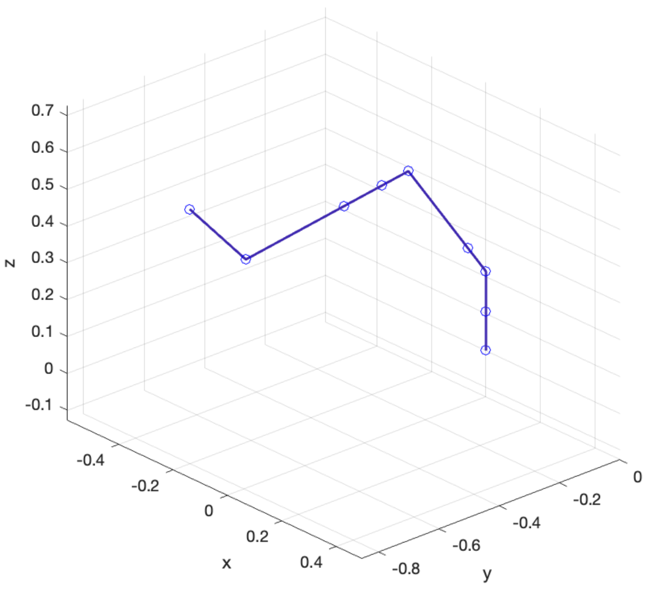

Figure 1: Final configuration of the robot during simulated motion in a three-dimensional graph. The measurement units for the three axes are meters.

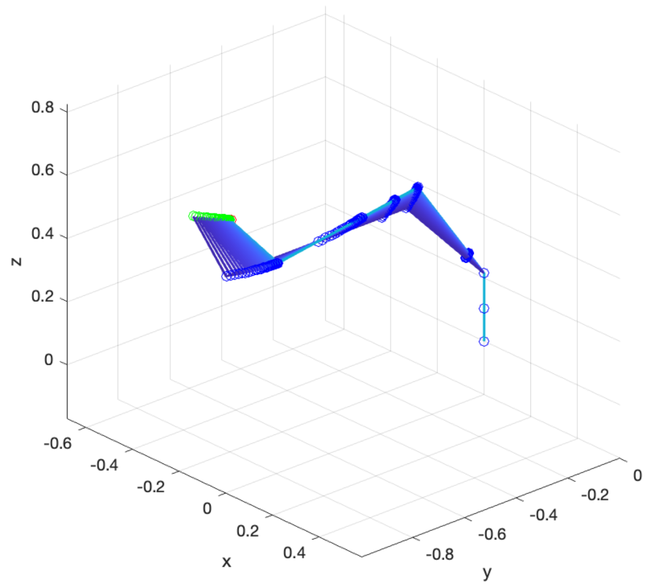

Figure 2: Description of the manipulator's simulation motion in a three-dimensional graph. The horizontal axes, denoted by x and y, represent the horizontal motion, while the vertical axis, denoted by z, represents the vertical motion. The measurement units for all three axes are meters.

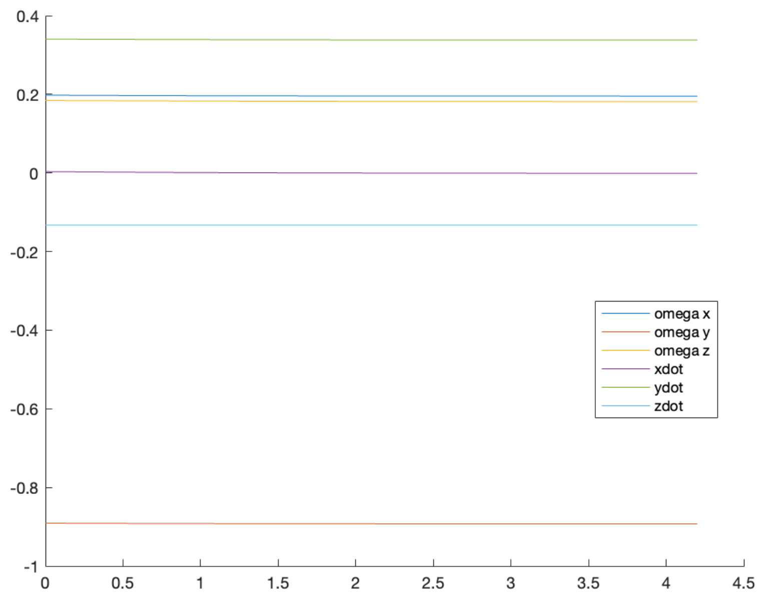

Figure 3: Envelope of end-effector angular ($\frac{rad}{s}$) and linear ($\frac{m}{s}$) velocities' direction.

4 Appendix

[Comment] Add here additional material (if needed)

4.1 Appendix A

4.2 Appendix B