Applied Robotics - FRTF20
Hand-in - Fall 2020

Kinematics - Robotics Toolbox

Hicham Mohamad (hi8826mo-s)

22 October 2020

Contents

introduction

In this hand-in, we are going to work on practical implementations in Robotics and it consists of two parts. First part will cover kinematics problem applied on a humanoid robot that consists of 35 joints. In the second part, we will deal with a dynamics problem applied on a 3 DOF joints IRB140 ABB robot. At the end, we will practice and get familiar with using the robotics toolbox.

The resulting plots and values in this report are obtained by running the implemented Matlab scripts `FRTF20_handin_humanoid.m` for problem 1, and `FRTF20_handin_dynamics_problem2.m` for problem 2.

D-H Representation

One systematic way of describing the geometry of a **serial chain of links** and joints is Denavit-Hartenberg notation.

In Denavit-Hartenberg notation, a **link** defines the spatial relationship between two neighboring joint axes. A link is specified by four parameters. The relationship between two link coordinate frames would ordinarily entail **six parameters**, three each for translation and rotation. For DH notation there are only **four parameters** but there are also **two constraints**:

1. Axis x_j intersects z_{j-1}.
2. Axis x_j is perpendicular to z_{j-1}.

For a manipulator with N joints numbered from 1 to N, there are N + 1 links, numbered from 0 to N. Joint j connects link $j-1$ to link j and moves them relative to each other. It follows that link l connects joint l to joint $l+1$. Link 0 is the base of the robot, typically fixed and link N, the last link of the robot, carries the end-effector or tool.

Denavit-Hartenberg convention is based on the fact that *the axis x_i intersects and is perpendicular to axis z_{i-1}*, and it is characterized by the following DH parameters definitions:

- **Joint angle** θ_i: the angle between x_{i-1} and x_i measured about z_{i-1}.
  Note: θ_i is variable if joint i is **revolute**.
- **link offset** d_i: the distance along z_{i-1} from the origin O_{i-1} to the intersection of x_i and z_{i-1} axes.
  Note: d_i is variable if joint i is **prismatic**.
- **Link length** a_i: the distance along x_i from the origin O_i to the intersection of x_i and z_{i-1} axes.
- **Link twist** α_i: the angle between z_{i-1} and z_i measured about x_i.

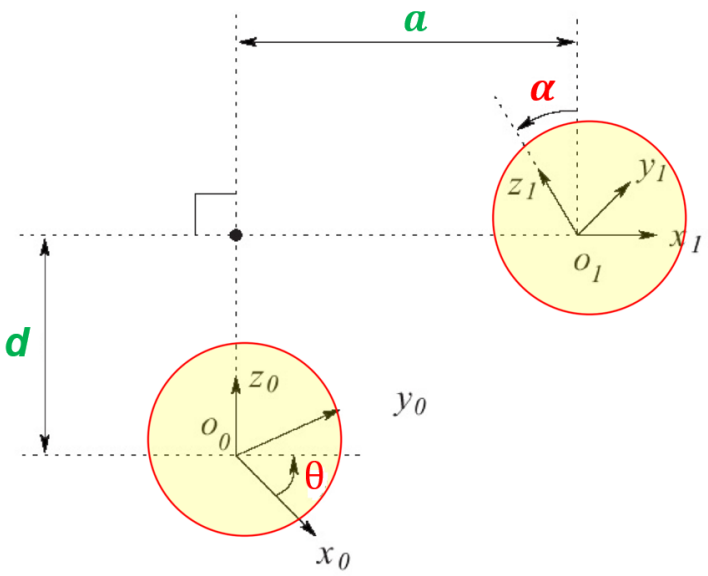


Figure 1: Schematic overview for the Denavit-Hartenberg convention: The axis x_i intersects and is perpendicular to axis z_{i-1}

1 Problem 1: Kinematics

A research group is enrolled in the implementation of a humanoid robot. The conceptual design appears below. It consists of 35 revolute joints but observe that 30 of them are organized in groups of three whose axes are intersecting in a single point to reproduce spherical motions (in the way as for the three last revolute joints of a decoupled robot), as shown below.

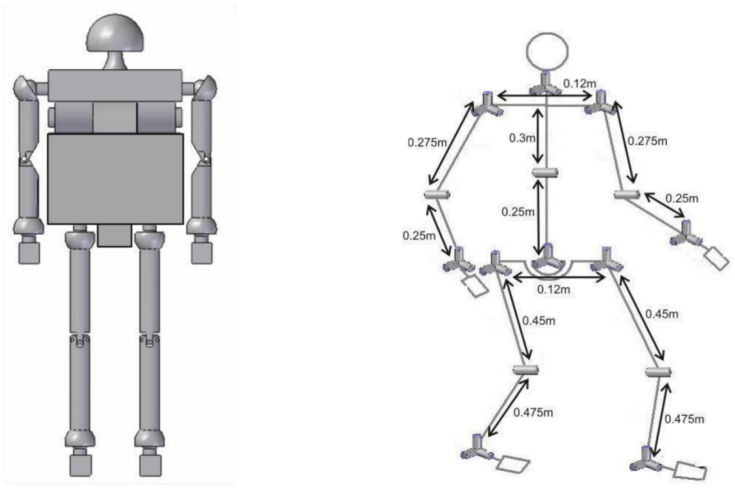


Figure 2: Humanoid robot.

The four limbs of the robot have the same kinematic structure that can be schematically represented as follows:

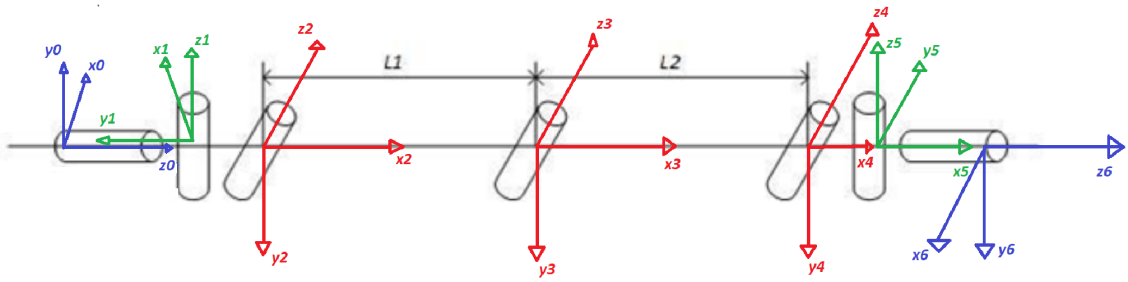


Figure 3: Schematic of the limbs and the corresponding coordinate systems of the kinematic chain for the Humanoid robot.

1.1 DH parameters for the limbs

By following the DH convention, we find the 4 parameters of the DH transformation from frame $i-1$ to frame i for the limbs, i.e. the DH-parameters for the kinematic chain in the above scheme in Figure 3. It is important to remember that the x-axis of a coordinate system is normal to the z-axis of the previous coordinate system. The resulting DH parameters are illustrated in Table 1.

Links	θ_i	d_i	a_i	α_i
Link 1	θ_1	0	0	$-\frac{\pi}{2}$
Link 2	$\theta_2 - \frac{\pi}{2}$	0	0	$-\frac{\pi}{2}$
Link 3	θ_3	0	L_1	0
Link 4	θ_4	0	L_2	0
link 5	θ_5	0	0	$\frac{\pi}{2}$
Link 6	$\theta_6 - \frac{\pi}{2}$	0	0	$-\frac{\pi}{2}$
Link 7	0	0	0	0

Table 1: the DH-parameters for the kinematic chain in the above scheme in Figure 3.

1.2 Geometric model of one arm

In this task, we need to build the geometric model of one arm using the Robotics Toolbox and simulate an arbitrary joint-interpolated motion between two arbitrary configurations. The implementation is found in the corresponding script where the links are joined into a **serial-link robot manipulator** using the robot toolbox. The resulting model is shown in Figure 4.

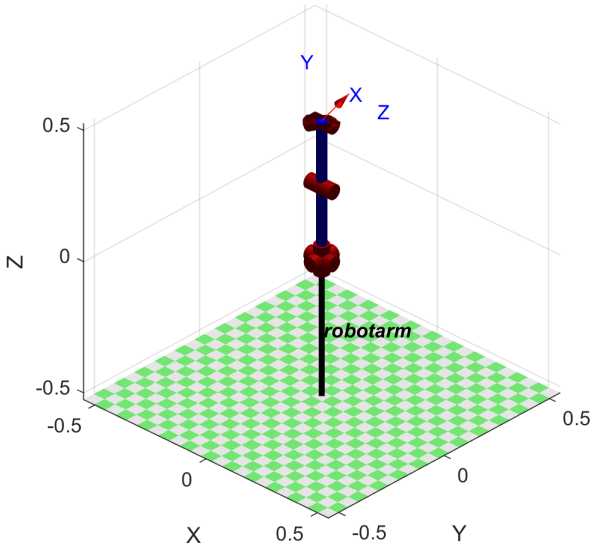


Figure 4: The obtained model of one robot arm of the Humanoid robot.

1.3 Solutions to the inverse kinematics

In this task, we need to investigate how many solutions to the inverse kinematics does the simulated **kinematics chain** have.

Thus, by considering the **inverse problem**, our goal is to find a solution, possibly many, of the equation

$$H_7^0 = A_1(q_1)A_2(q_2)\cdots A_7(q_7) = H \tag{1}$$

where H is a given 4×4 **homogeneous transformation** obtained as a solution of the **forward kinematics problem**

$$H = \begin{bmatrix} \boldsymbol{R}_{3\times 3} & \boldsymbol{o}_{3\times 1} \\ \boldsymbol{0}_{1\times 3} & 1 \end{bmatrix} \in SE(3) \tag{2}$$

So we have **12 equation** with respect to 7 variables: $q_1, q_2, \cdots q_7$. *However, these equations are not independent since 9 entries of the* ***rotation matrix*** *satisfy certain equations and can be defined by just 3* ***independent variables****, such Euler angles* (as mentioned in the lecture notes on page 58 in [2]). Consequently, it follows that we have 6 independent variables including the 3 variables of the vector $\boldsymbol{o}$.

In this way, we can say that there may be several/infinity of solutions for particular target position and orientation since $n = 7 > 6$ in our problem. If $n < 6$ we would say that there is no solution or not every target position/orientation can be achieved.

1.4 Assembling two legs and two arms

Here we need to assemble two legs and two arms so that the result corresponds to the designed robot in the case in which the robot's trunk joints are blocked.

In addition to the arm DH parameters found in the previous section, we need to find the DH parameters for one leg. Then, in the same way we can construct the model for the right arm and leg by shifting the base the necessary distance measured in Figure 2. The resulting assembled arms and legs are illustrated in Figure 5.

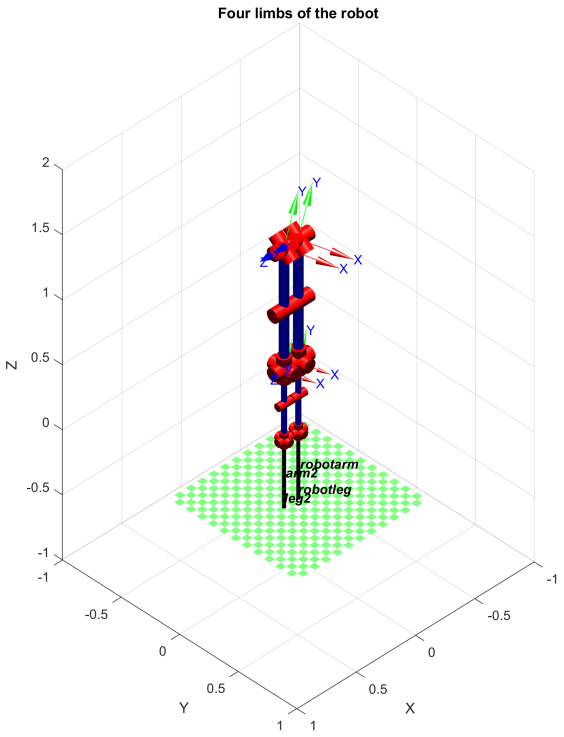
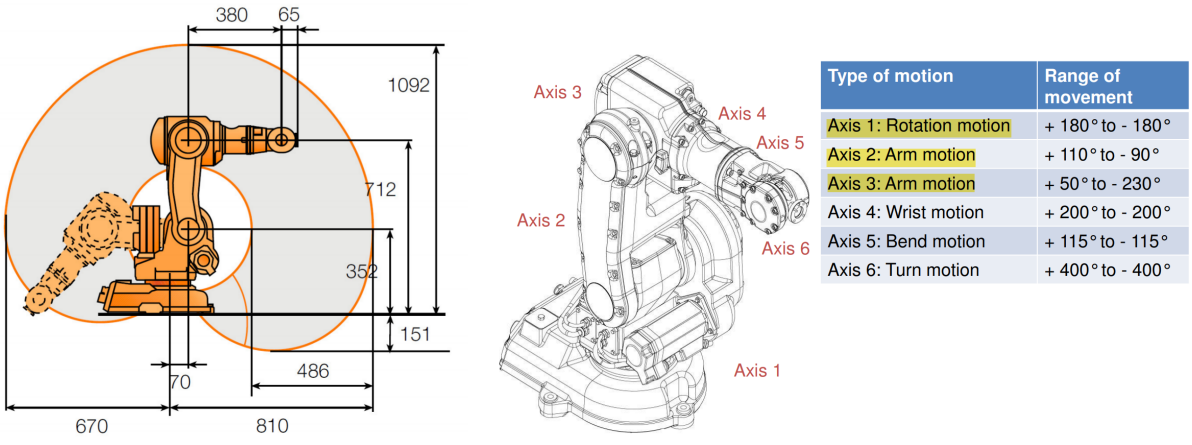


Figure 5: The resulting model after assembling the 4 limbs of Humanoid robot.

2 Problem 2: Robotics Toolbox / Dynamics



Type of motion	Range of movement
Axis 1: Rotation motion	+ 180° to - 180°
Axis 2: Arm motion	+ 110° to - 90°
Axis 3: Arm motion	+ 50° to - 230°
Axis 4: Wrist motion	+ 200° to - 200°
Axis 5: Bend motion	+ 115° to - 115°
Axis 6: Turn motion	+ 400° to - 400°

(a) (b)

*Figure 6: Illustration of **ABB IRB 140** industrial robot: a) Schematic view of workspace for ABB IRB140 [6]. b) The six joints/axes ot the robot arm*

2.1 a) Model of the 3 DOF robot

By using Peter Corke's toolbox, we can set up the model of the 3 DOF robot which corresponds to the first three joints of IRB140, shown in Figure 6a. By following the DH convention, we find the 4 parameters of the **DH transformation** from frame $i-1$ to frame i for the kinematic chain in the scheme in Figure 7. The resulting DH parameters are found in Table 2.

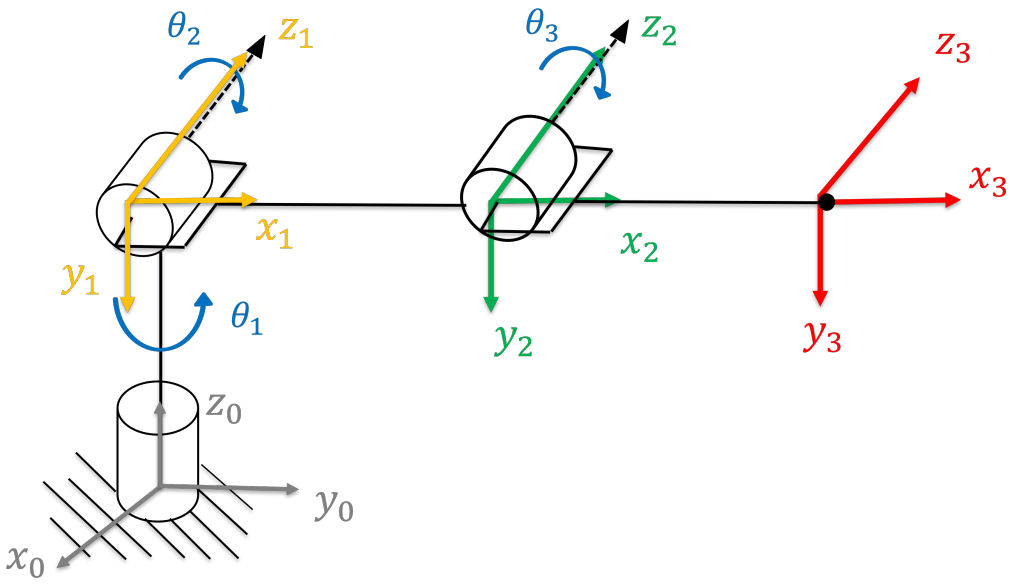


Figure 7: Schematic of the 3 DOF robot and the corresponding coordinate systems of the kinematic chain.

Here we should in addition to the kinematic parameters also supply enough parameters like total robot weight (100kg) to be able to get a dynamic model representation.

Links	θ_i	d_i	a_i	α_i
Link 1	θ_1	L_1	0	$-\frac{\pi}{2}$
Link 2	θ_2	0	L_2	0
Link 3	θ_3	0	L_3	0

Table 2: the DH-parameters for the kinematic chain of the 3 DOF robot illustrated in Figure 7.

2.2 b) Stationary torque

Here we need to find out how much stationary torque is needed on the motor side to overcome the influence of gravity for the configuration $(\theta_1 = 0°, \theta_2 = 10°, \theta_3 = 40°)$. As implemented in the script, we use the function `TAU = R.rne(Q, QD, QDD, OPTIONS)`, which calculates the **joint torque** required for the robot R to achieve the specified joint position Q (1xN), velocity QD (1xN) and acceleration QDD (1xN), where N is the number of robot joints.

2.3 c) Path-generation/simulation

For making a path-generation/simulation for joint-wise or linear motion from $(\theta_1, \theta_2, \theta_3) = (10°, 20°, 30°)$ to $(-10°, 40°, 10°)$, we use the function `[Q,QD,QDD] = jtraj(deg2rad()`.

2.4 d) Simulation of a free-falling robot

For making a simulation of a "free-falling" robot from its home-position (joint torques =0), the function `RNF = R.nofriction()` is used, which returns a copy of the robot object that is similar in all respects except that the **Coulomb friction** is zero. To simulate the motion from rest in the zero angle pose with zero applied joint torques, we write in Matlab the following

```
1 [t q qd] = R.nofriction.fdyn(10, [], zeros(1,3));
```

where `fdyn()` denotes the **forward dynamics**, i.e. it is the computation of joint accelerations given position and velocity state, and actuator torques. It is useful in simulation of a robot control system.

No considerations need to be taken to joint limitations and colliding links. The resulting model is shown in Figure 8

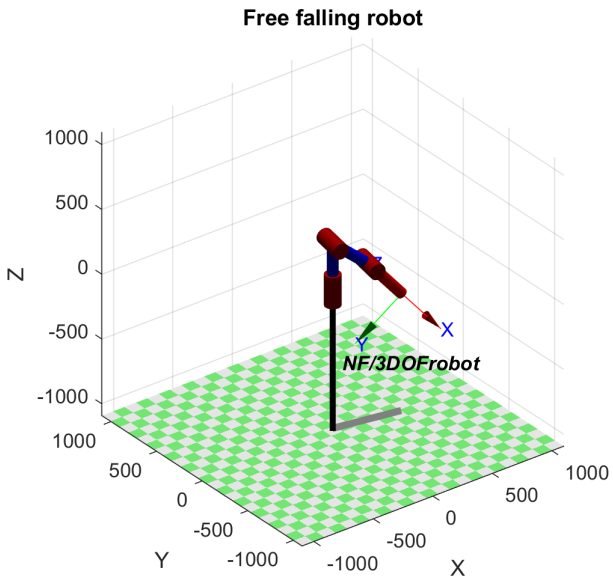


Figure 8: Simulation of the free falling robot.

Appendix

References

[1] Anders Robertsson, Applied Robotics - FRTN20, Lectures notes, LTH

[2] L. Freidovich, Control Methods for Robotic Applications, 2017

[3] B. SICILIANO, L. Sciavicco, L. Villani, G. Oriolo, Robotics — Modelling, Planning and Control, Springer Advanced Textbooks in Control and Signal Processing Series, London, UK, 2009 Italian edition: Robotica — Modellistica, Pianificazione e Controllo, McGraw-Hill Libri Italia, Milano, 2008

[4] P.I. Corke, Robotics, Vision and Control, Springer, ISBN 978-3-319-54413-7,

[5] P.I. Corke, Using the Robotics Toolbox with a real robot

[6] ABB Robotics (2004): "Operating manual — IRC5 with FlexPendant", Document ID: 3HAC16590-1, Revision: P

[7] P.I. Corke (2014): "4 is harder than 6: Inverse kinematics for underactuated robots", `http://www.petercorke.com/doc/rtb_4dof.pdf` (Visited: Sept 2018).