CentraleSupélec

Interactive Robotic Systems (2EL1120)

Tutorial 1: Direct and inverse kinematics

Submitted By:
Pedro Javier Soler Olivares
Armin Wessel

May 16, 2021

CentraleSupélec

Direct geometric model

Q1

The frame orientations are marked in Figure 1.

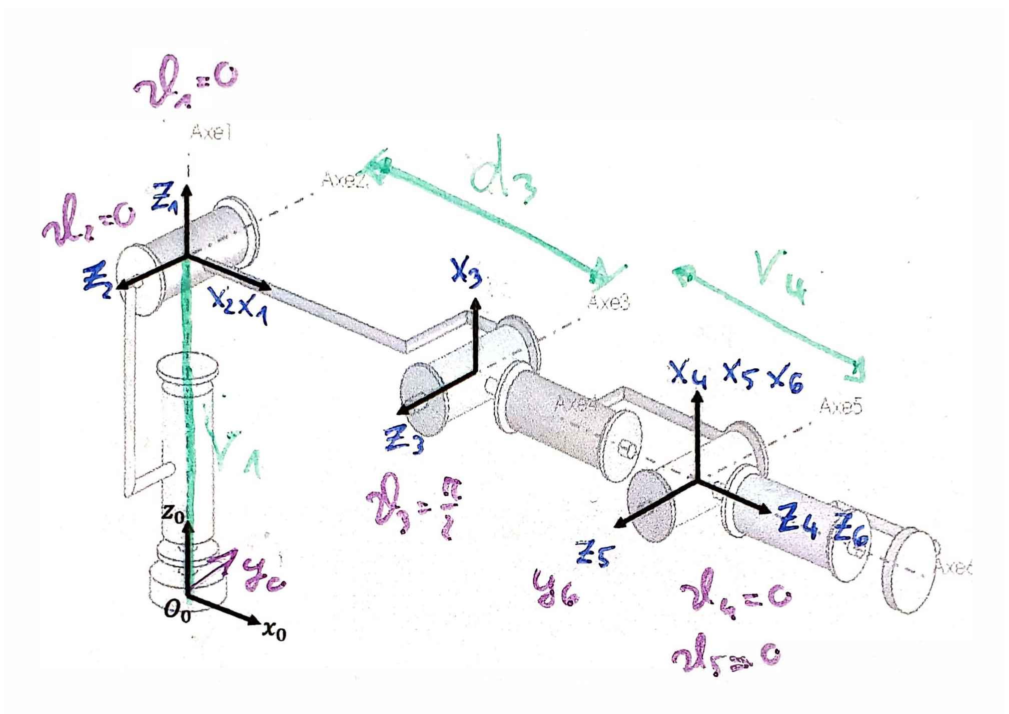

Figure 1: Coordinate frames according to MDH convention

Q2

The parameters of the robot are shown in Table 1. The joint configurations are determined by the angles θ_i. The variable θ_3 notably has an offset, which needs to be respected in the computations.

i	α_i	d_i	θ_i	r_i
1	0	0	θ_1	r_1
2	$\frac{\pi}{2}$	0	θ_2	0
3	0	d_3	$\theta_3 + \frac{\pi}{2}$	0
4	$\frac{\pi}{2}$	0	θ_4	r_4
5	$-\frac{\pi}{2}$	0	θ_5	0
6	$\frac{\pi}{2}$	0	θ_6	0

Table 1: Geometric parameters of the robot

Q3

The transformation from one frame to the next using the the parameters obtained from the MDH convention is defined as the following chain of elementary transformations:

$$\bar{g}_{(i-1)i} = R_{x,\alpha_i} T_{x,d_i} R_{z,\theta_i} T_{z,r_i}.$$

This transformation is implemented in the matlab function `TransformMatElem`.

The function `ComputeDGM` accepts a vector for each parameter per the MDH convention. The function applies the transformation defined above successively for all elements of the parameter vectors. Finally, as the endeffector position is offset from the last coordinate frame by r_E, an offset is applied.

Q4

Given the two joint configuration vectors

$$q_i = \left[-\frac{\pi}{2}, 0, -\frac{\pi}{2}, -\frac{\pi}{2}, -\frac{\pi}{2}, -\frac{\pi}{2}\right]^T$$

and

$$q_f = \left[0, \frac{\pi}{4}, 0, \frac{\pi}{2}, \frac{\pi}{2}, 0\right]^T$$

the following results were computed by the script `DGM.m`:
For q_i

$$t = \begin{bmatrix} -0.1000 \\ -0.7000 \\ 0.3000 \end{bmatrix}$$

$$\omega = \begin{bmatrix} -0.5774 \\ -0.5774 \\ 0.5774 \end{bmatrix}$$

$$\theta = 2.0944$$

For q_f

$$t = \begin{bmatrix} 0.6364 \\ -0.1000 \\ 1.1364 \end{bmatrix}$$
$$\omega = \begin{bmatrix} 0.2811 \\ 0.6786 \\ -0.6786 \end{bmatrix}$$
$$\theta = 2.5936$$

Q5

The function `PlotFrame` creates a 3D plot showing the locations and orientations of the coordinate frames of the endeffector. The plot is shown in Figure 2.

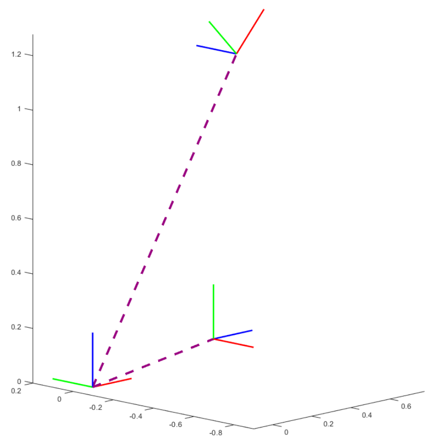

Figure 2: Endeffector coordinate frames for q_i and q_f

Direct kinematic model

Q6

The robot has only rotatory joints. By the methodology of velocity composition, the Jacobian ${}^0J(q)$ for the robot consists of the columns

$${}^0J(q) = \begin{bmatrix} {}^0J_1(q) & {}^0J_2(q) & {}^0J_3(q) & {}^0J_4(q) & {}^0J_5(q) & {}^0J_6(q) \end{bmatrix},$$

where

$$^{0}J_i(q) = \begin{bmatrix} R_{0i}(Z_i \times p_{iN}) \\ R_{0i}Z_i \end{bmatrix}.$$

Using the method above, `ComputeJac` calculates the twist $^{0}\nu_{0,E}$ by multiplying the Jacobian with the vector of joint velocities

$$\dot{q} = \begin{bmatrix} 0.5 \\ 1 \\ -0.5 \\ 0.5 \\ 1 \\ -0.5 \end{bmatrix}.$$

The resulting twist

$$^{0}\nu_{0,E} = \begin{bmatrix} ^{0}J_v(q) \\ ^{0}J_\omega(q) \end{bmatrix} \dot{q}$$

evaluates to

$$^{0}\nu_{0,E} = \begin{bmatrix} 0.35 \\ -0.10 \\ 0.60 \\ -0.00 \\ -1.00 \\ 0.00 \end{bmatrix}$$

for $q = q_i$ and

$$^{0}\nu_{0,E} = \begin{bmatrix} -0.5510 \\ 0.3182 \\ 0.4596 \\ 1.0607 \\ -0.0000 \\ 0.1464 \end{bmatrix}$$

for $q = q_f$.

Q7

We can use the singular value decomposition of the velocity Jacobians J_{v_i} and J_{v_f} to examine the velocity transfer characteristics of the configurations q_i and q_f . The decomposition of a matrix J results in

$$J = U\Sigma V^T.$$

The matrix Σ gives insight into the manipulabilities. It is a diagonal matrix, with the singular values σ_i on the diagonal. For the decomposition of $J(q_i)$ we get

$$\Sigma_i = \begin{bmatrix} 0.74 & 0 & 0 & 0 \\ 0 & 0.70 & 0 & 0 \\ 0 & 0 & 0.21 & 0 \end{bmatrix},$$

for $J(q_f)$ we get

$$\Sigma_f = \begin{bmatrix} 0.93 & 0 & 0 & 0 \\ 0 & 0.64 & 0 & 0 \\ 0 & 0 & 0.10 & 0 \end{bmatrix}.$$

The preferred direction is the direction with the highest associated σ value, for both cases this is the x-direction. The manipulabilities

$$W = \prod_{i=1}^{r} \sigma_i$$

are connected to the volumes of the ellipsoids. The values are

$$\begin{aligned} W_i &= 0.11156 \\ W_f &= 0.05902 \end{aligned}$$

respectively. The corresponding velocity ellipsoids are shown in Figure 3. As a supplement to the above, a singular configuration is examined. A singular configuration is obtained, by setting

$$q_i^* = \left[-\frac{\pi}{2}, 0, 0, 0, 0, 0\right]^T$$

instead of

$$q_i = \left[-\frac{\pi}{2}, 0, -\frac{\pi}{2}, -\frac{\pi}{2}, -\frac{\pi}{2}, -\frac{\pi}{2}\right]^T.$$

This results in one of the singular values being zero (apart from numerical tolerances). As a result the ellipsoid degenerates to an ellipse, the manipulability W_i^* is zero, see Figure 4.

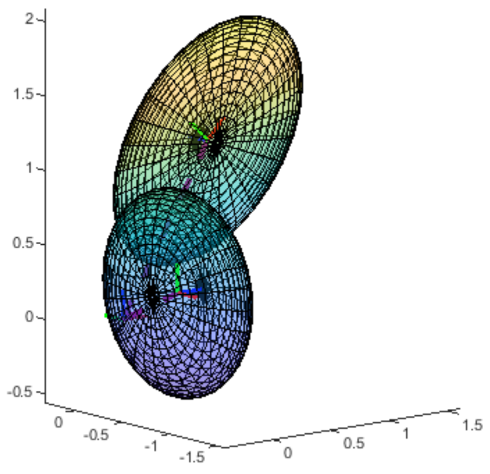

Figure 3: Velocity Manipulability Ellipses

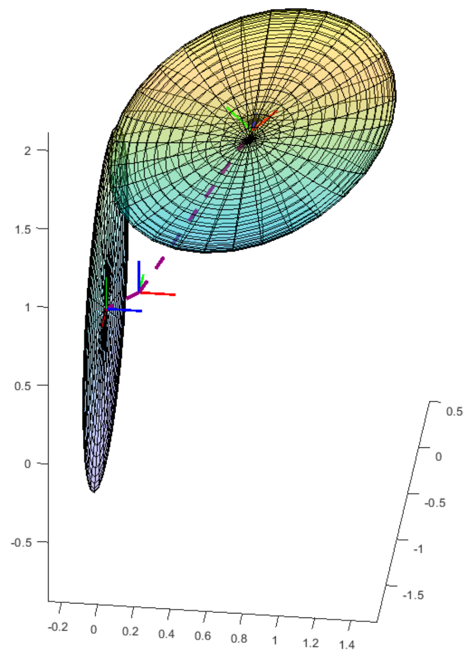

Figure 4: Velocity Manipulability Ellipses with q_i in a singular configuration

Inverse geometric model

Q8

The function `ComputeIGM` iteratively searches for a joint configuration that produces a task space position within an error margin of the desired value. Such a process is sensitive to starting conditions. The function is using a Newton-Raphson-based iteration with a cartesian error.

1

For a desired configuration

$$X_{d_i} = \begin{bmatrix} -0.1 \\ -0.7 \\ 0.3 \end{bmatrix}$$

with the starting vector given in the exercise, the algorithm finds

$$q_i^* = \begin{bmatrix} -1.572 \\ 0.015 \\ -1.541 \\ -1.478 \\ -1.464 \\ -1.414 \end{bmatrix}.$$

The error

$$e_i = X_{d_i} - f(q_i^*) = 10^{-7} \cdot \begin{bmatrix} 0.0521 \\ -0.6635 \\ -0.3991 \end{bmatrix}$$

is vanishingly small. Figure 5 shows that only a few iterations were needed to arrive at the final result.

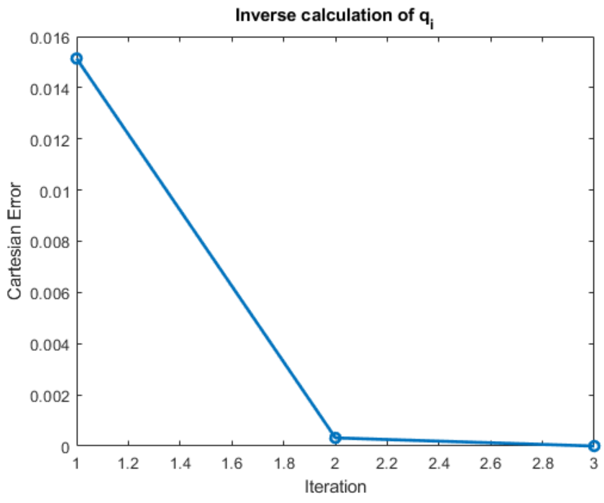

Figure 5: Cartesian error for each iteration step of q_i^*

2

For

$$X_{d_f} = \begin{bmatrix} 0.64 \\ -0.1 \\ 1.14 \end{bmatrix}$$

with the given starting vector, the algorithm finds

$$q_f^* = \begin{bmatrix} -6.422 \\ 0.511 \\ -5.320 \\ -132.778 \\ 18.717 \\ 122.082 \end{bmatrix}.$$

The resulting error

$$e_f = X_{d_f} - f(q_f^*) = 10^{-3} \cdot \begin{bmatrix} -0.3695 \\ -0.6890 \\ 0.4804 \end{bmatrix}$$

is within tolerances, however the angles are out of the feasible area of the joints. This behaviour is likely a result of the starting condition given in the exercise. The starting condition is relatively far away from the future solution, so the algorithm is mislead, and finds a solution that is offset with multiple full rotations. Figure 6 shows that much more iterations were needed, and that the error does not decrease uniformly.

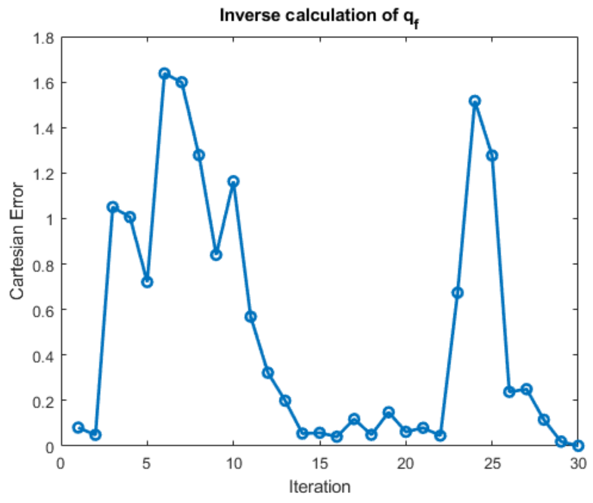

Figure 6: Cartesian error for each iteration step of q_f^*

Inverse kinematic model

Q9

An endeffector trajectory is made up of a sequence of points the endeffector has to reach for a given time sequence. To get from one point to another, the trajectory can be planned in the joint space or in the tasks space. In the following, we develop a function to transform a task-space trajectory into a joint-space trajectory. Figure 7 shows the plot from Q5 with the trajectory and a few frames added. The plot shows, that the orientation is not changing for most of the trajectory. This is to be expected, as each successive frame is computed by very slightly modifying the parameters of the previous one. As long as the robots reach allows it, this results in a constant orientation. However at some point the geometry might force another orientation, which will result in a jump in the joint trajectory.

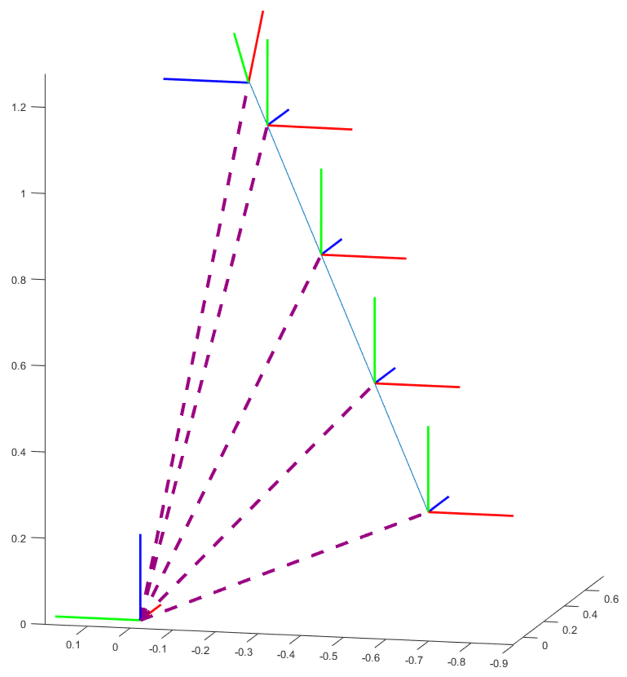

Figure 7: Trajectory in task-space with a few frames added along the way

Q10

The evolution of the joint variables is shown in Figure 8. The green area marks the interval within the physical joint limits. The sudden jump around iteration 1170 is due to q_3 reaching a singularity. The value of q_3 first goes very close to zero, than jumps to around 26.4. With q_3 the other variables also experience a discontinuity.

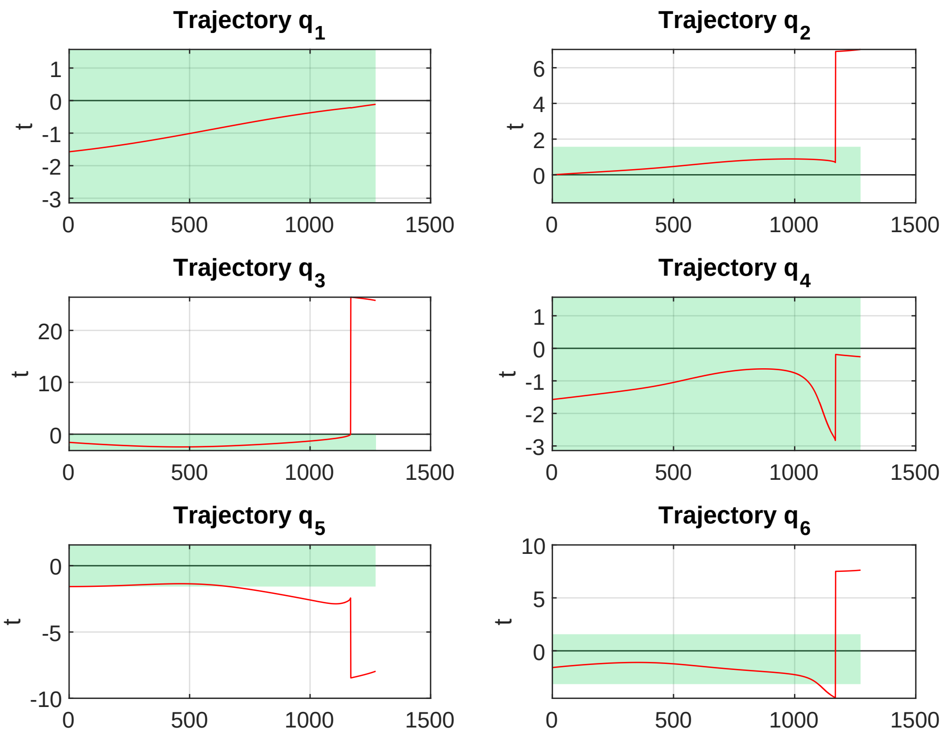

Figure 8: Temporal evolution of joint variables for Q10

Q11

For this last part the algorithm developed for Q9 is modified. The modifications are stored in `ComputeIKMlimits` and `ComputeIGMlimits`. Because the robot is redundant when only the location of the endeffector is under constraints, the remaining degrees of freedom can be used to optimise the joint trajectories. The following method considers a quadratic distance between the middle position of the joint and the current joint position as an objective to be minimized. The resulting task-space trajectory is shown in Figure 9.

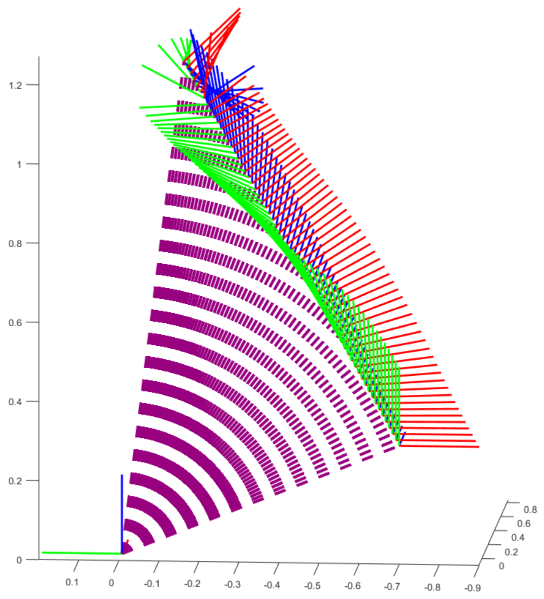

Figure 9: Trajectory in task-space with a many frames added along the way

Figure 10 shows the evolution of the joint variables. The added attraction towards the mean fixes the issue of the singularity for q_3. Some other joints still experience a jump, however the behavior is improved. There exist many additional methods that make use of redundancies to optimize the joint trajectories.

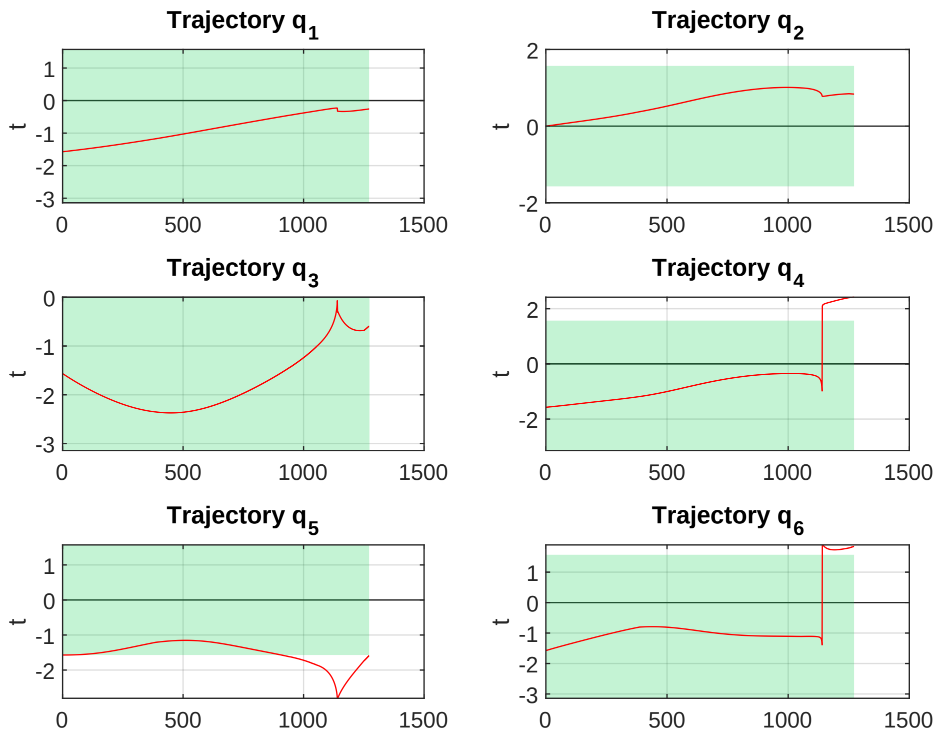

Figure 10: Temporal evolution of joint variables for Q11