CentraleSupélec - Interactive Robotic Systems

Report of Tutorials 1 and 2

Students:

Vítor ALBUQUERQUE MARANHÃO RIBEIRO and

João Victor EVANGELISTA MATOSO

1. Introduction

The objective of this report is to summarize our results to the questions proposed on the tutorial sessions of the 2nd year's subject of CentraleSupélec, "*Interactive Robotic Systems*". During the tutorials, we were able to apply the knowledge learnt in the subject by modelling and controlling a manipulator arm developed by the *Interactive Robotics Laboratory of the CEA LIST* (figure 1). The robot has a kinematic chain of serial type, and it has 6 revolute joints.

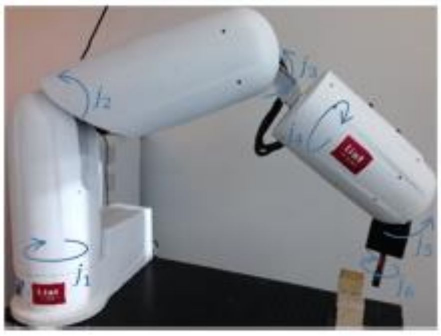


Figure 1 – Prototype of the robotic manipulator of CEA-LIST.

2. Direct Geometric Model

Question 1

Given the joints positions, their frames, and some geometric parameters of the robot, we were able to define the following frame configuration using de MDH convention (Modifed Denavit-HartenbeR, or Khalil-Kleinfinger convention) on the figure 2.

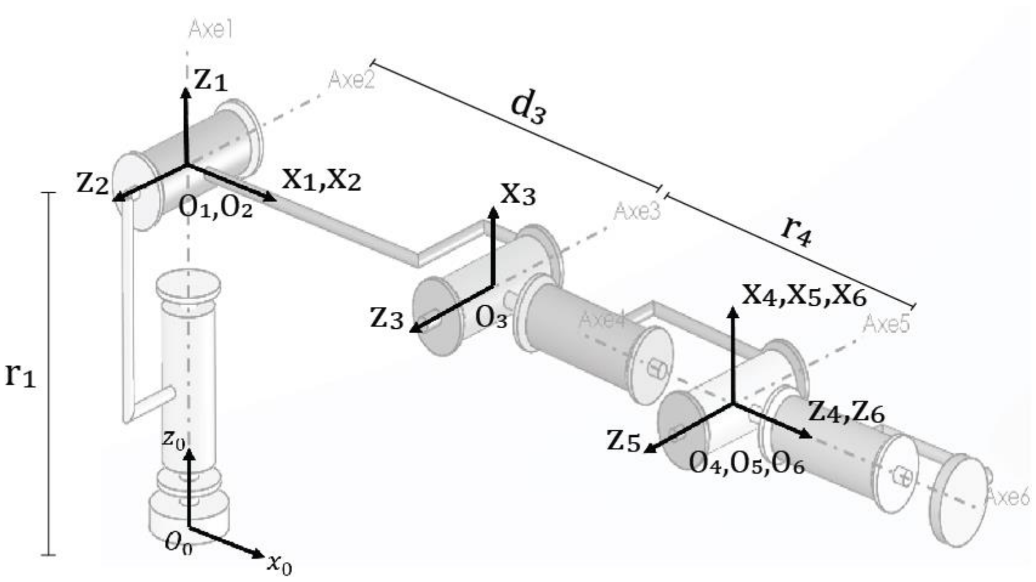


Figure 2 – Frame configuration for this study.

Question 2

Having defined the frames position, and knowing that the Figure 2 shows the robot in the zero configuration, where $q = (q_1, q_2, q_3, q_4, q_5, q_6) = (0,0,0,0,0,0)$, then we were able to complete the table with the geometric parameters of the robot on the MDH convention.

Joint (*j*)	α_j	$\mathbf{d}_j$	θ_j	r_j
1	0	0	q_1	$\boldsymbol{r_1}$
2	$\pi/2$	0	q_2	0
3	0	d_3	$q_3 + \pi/2$	0
4	$\pi/2$	0	q_4	$\boldsymbol{r_4}$
5	$-\pi/2$	0	q_5	0
6	$\pi/2$	0	q_6	0
E	0	0	0	$\boldsymbol{r_E}$

Table 1 – Geometric parameters of the robot on MDH convention.

Question 3

The transformation matrix is the matrix that expresses a frame $\mathcal{R}_{i-1}$ into a frame $\mathcal{R}_i$. For computing the transformation matrix, by the programmed function $g = TransformMatElem(\alpha_i; d_i; \theta_i; r_i)$, the expression from the slide 96 was used, where:

$$g_{(i-1)i} = \mathrm{Rot}_x(\alpha_i)\ \mathrm{Trans}_x(d_i)\ \mathrm{Rot}_z(\theta_i)\ \mathrm{Trans}_z(r_i)$$

Then, to create the function ${}^{0}X_N = ComputeDGM(\alpha; d; \theta; r)$, which computes the Direct Geometric model of any open chained robot, given its parameters, we computed g_{0N} recursively, starting from $g_{0N} = g_{01}\, g_{12} \dots g_{N-1,N}$. The returned parameters of the function are the points of the Nth frame origin of the robot, which are the coordinates of $p_{0N} = g_{0N}(1{:}3,4)$, and its orientation, given by the Euler angles, obtained by the rotation matrix $R_{0N} = g_{0N}(1{:}3,1{:}3)$.

After that, to compute the position and orientation of the robot's end-effector with respect to the base frame, the function $ComputeDGM$ was used, but now with the inputs having one more line, which corresponds to the geometric parameters of the end-effector with respect to the frame $\mathcal{R}_N$. The values used for these parameters are the last one of the table 1.

Question 4

By inputting the suggested values on the previously programmed function $ComputeDGM$, and computing the $\vec{n}$, rotation direction vector and $\phi \in [0, \pi]$, the rotation angle between the frame 0 and E by the *Inverse relationship of Rodrigues formula* (slide 61), the following results were obtained:

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

$$p_{0E_i} = (-0.1\,, -0.7\,, 0.3)^t\ [m]$$

$$n_i = (-0.5774\,, -0.5774\,, 0.5774)^t$$

$$\phi_i\ = 2.0944\ [rad]$$

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

$$p_{0E_f} = (0.6364\,, -0.1\,, 1.1364)^t\ [m]$$

$$n_f = (0.2811\,, 0.6786\,, -0.6786)^t$$

$$\phi_f = 2.5936\ [rad]$$

A better visualization of both configurations will be available in the question 5.

Question 5

To better visualize the robot's configuration, a visualization function *PlotFrame(q)* was programmed displaying the robot's tree (red-dashed lines on the plot), as well as the position and the orientation of the end-effector frame $\mathcal{R}_E = (O_E, x_E, y_E, z_E)_0$ and the base frame $\mathcal{R}_0 = (O_0, x_0. y_0, z_0)$. On this plot, there is also the rotation vector **n** and the angle of rotation $\phi(in degrees)$ from the frame $\mathcal{R}_0$ to $\mathcal{R}_E$ (described as Rot Axis in the legend of the plots). So, we will visualize the robot's configuration submitted to the joint coordinates q_i and q_f from the previous question.

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

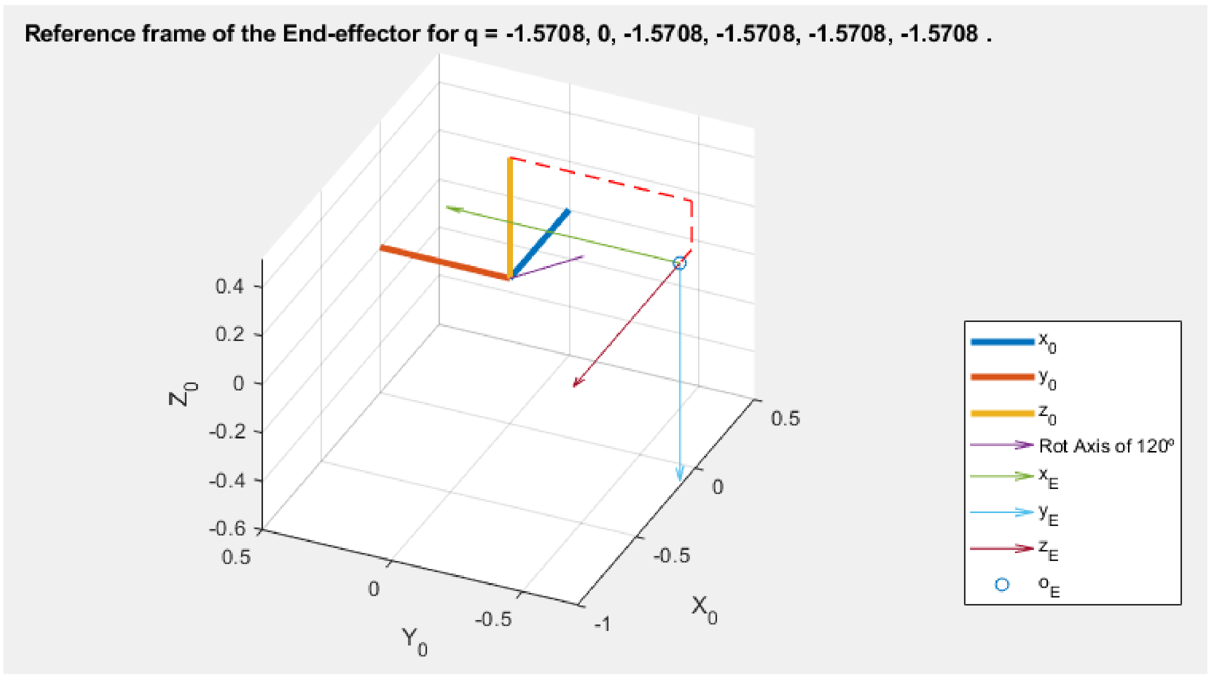


Figure 3 – Robot's configuration with the joint coordinates at q_i.

The plot agrees with the expected result, since

$$R_{0E}(q_i) = \begin{pmatrix} 0 & 0 & -1 \\ 1 & 0 & 0 \\ 0 & -1 & 0 \end{pmatrix}, i.e: \begin{cases} x_E = y_0 \\ y_E = -z_0 \\ z_E = -x_0 \end{cases}$$

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

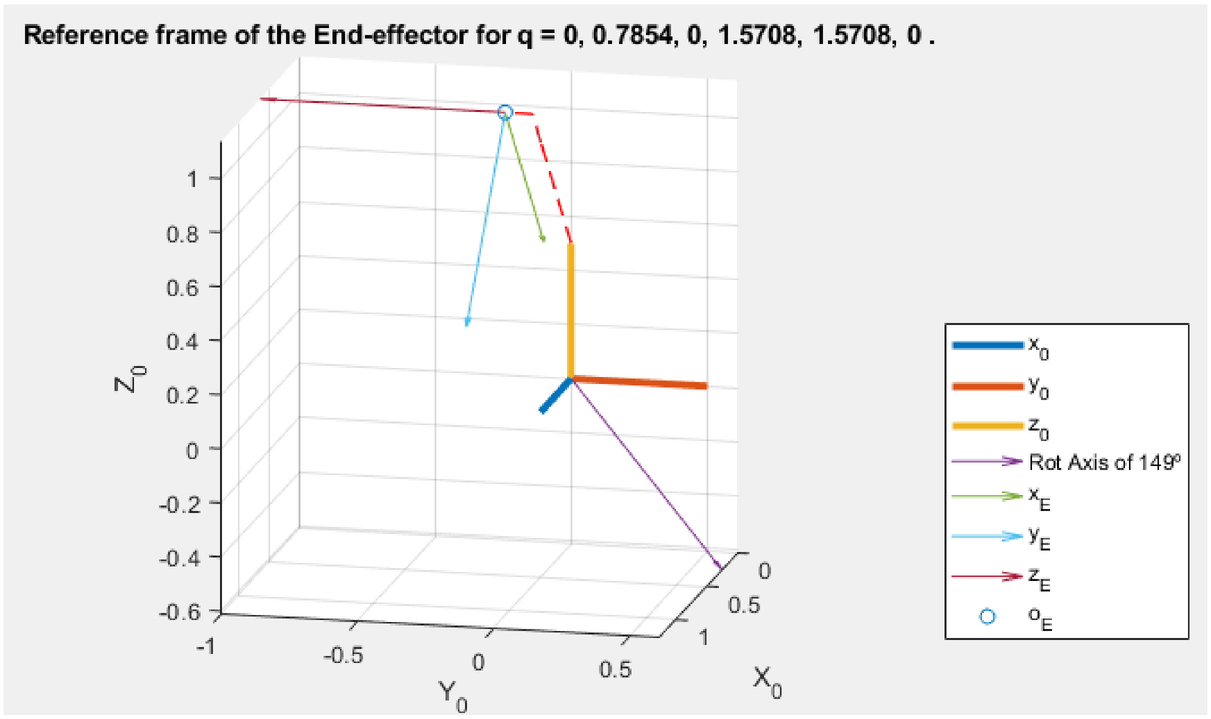


Figure 4 – Robot's configuration with the joint coordinates at q_f.

3. Direct kinematic model

Question 6

The robot's Jacobian Matrix was computed for a given joint configuration q. For this calculation, the approach from the slide 107 was used, that since the robot has only revolute joints, then:

$$J^0(q)_i = \begin{bmatrix} R_{0i} Z_i^0 \times p_{iE}^0 \\ R_{0i} Z_i^0 \end{bmatrix} \text{ (for revolute joints)}$$

Since the Jacobian Matrix relates the end-effector's velocities in the task space to the joint velocities by the following relation: $\mathcal{V}_E^0 = \begin{pmatrix} V_{0,E}(O_E) \\ \omega_{0,E} \end{pmatrix} = J^0\,(q) \cdot \dot{q}$. Then the end-effectors velocities at the previous configurations q_i and q_f for $\dot{q} = [0.5\,, 1.0\,, -0.5\,, 0.5\,, 1.0\,, -0.5]^t$ were:

$$\mathcal{V}_E^0(q_i, \dot{q}) = [0.35\,, -0.1\,, 0.6\,, 0\,, -1\,, 0]^t$$

$$\mathcal{V}_E^0(q_f, \dot{q}) = [-0.5510\,, 0.3182\,, 0.4596\,, 1.0607\,, 0\,, 0.146]^t$$

Question 7

To analyze the transmission of velocities between joint and task spaces, the velocities ellipsoids is computed by doing the SVD decomposition of the velocities part of the Jacobian matrix. $J = U\,\Sigma\,V^t$, where the columns of U represent the principal axis of the ellipsoid and Σ contain the eigen values of $J_v^0(q)$ which represent the length σ_i of the ellipsoid at each direction U_i .

Therefore, the preferred direction to transmit velocity in the task space is the first column of the matrix U_i, which has the highest length σ_1. Also, the velocity manipulability is analyzed by the volume of the ellipsoid, calculated by $\mathcal{W} = det(J \cdot J^t) = \prod_j\,(\sigma_j > 0)$.

For $q_i = \left[-\frac{\pi}{2}, 0, -\frac{\pi}{2}, -\frac{\pi}{2}, -\frac{\pi}{2}, -\frac{\pi}{2}\right]^t$, the ellipsoid of velocity manipulability is show in the plot below

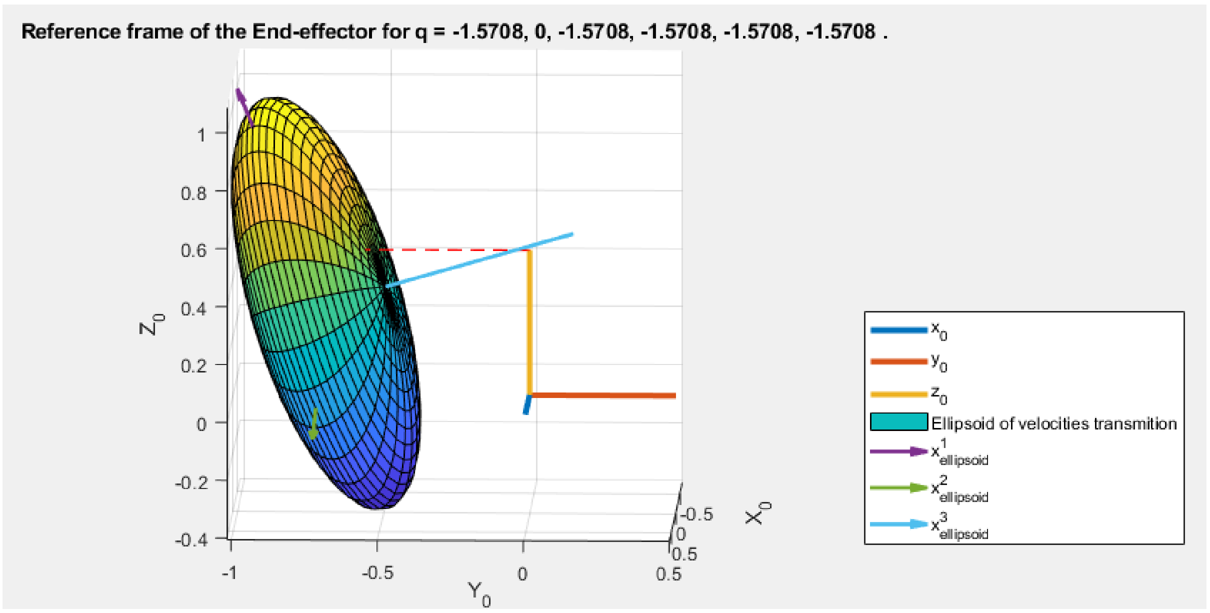


Figure 5 – Ellipsoid of velocity manipulability for the Robot's configuration with the joint coordinates at q_i.

Where $U_1(q_i) = [0.3660\,, -0.3263\,, 0.8715]^t$, and $\mathcal{W}(q_i) = 0.5212$.

And for $q_f = \left[0, \frac{\pi}{4}, 0, \frac{\pi}{2}, \frac{\pi}{2}, 0\right]^t$, the ellipsoid of velocity manipulability is show in the plot above

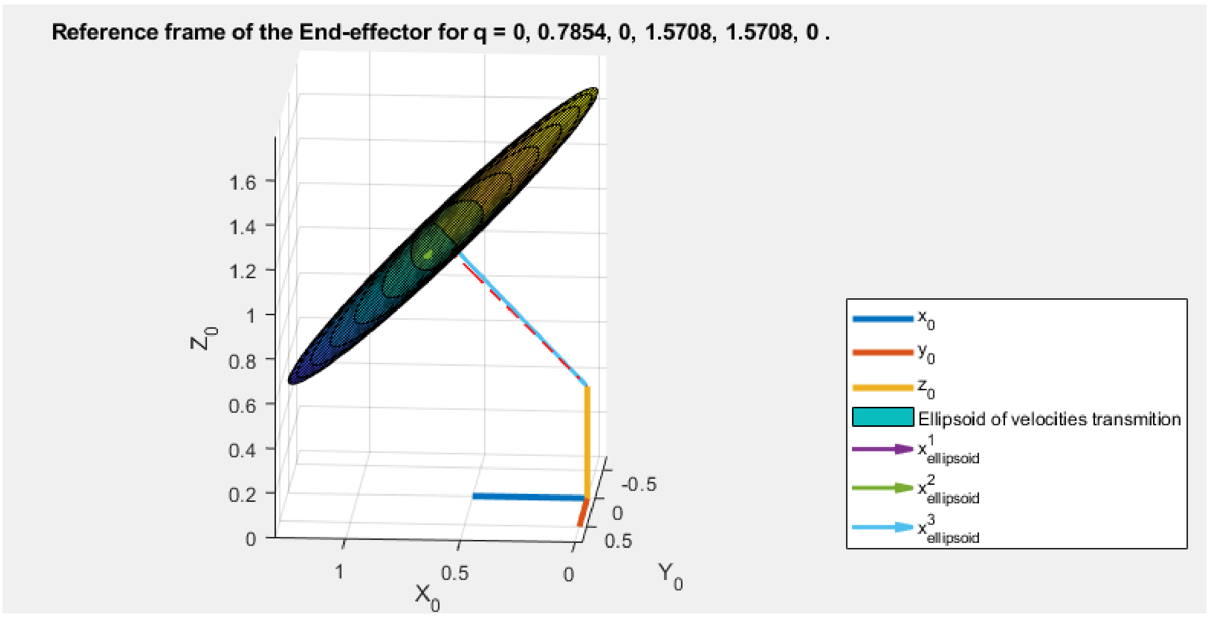


Figure 6 – Ellipsoid of velocity manipulability for the Robot's configuration with the joint coordinates at q_f.

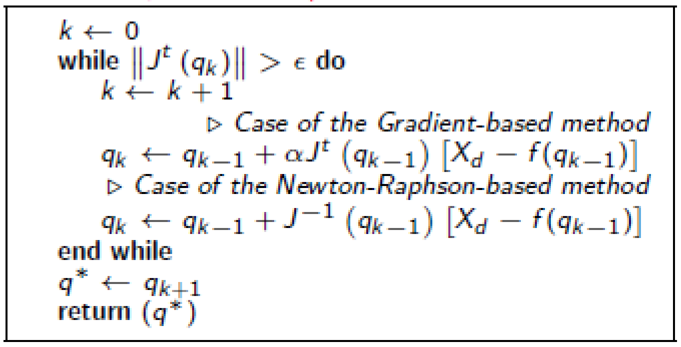
Where $U_1(q_f) = [-0.7114\,,\ -0.0975\,, 0.6959]^t$, and $\mathcal{W}(q_f) = 0.9324$.

We can see that the manipulability of the configuration q_f is higher than the one of q_i, however, for q_f, its third principal direction is significantly narrower than its other two ones, having one direction with low transmissivity.

4. Inverse geometric model

<u>Question 8</u>

The computation of the inverse geometric model of the robot was made numerically following the pseudo algorithm presented below and using the Moore-Penrose pseudo-inverse function to invert the Jacobian. Also, the used criteria for the iteration to end was the cartesian error $||X_d - f(q_k)|| < \epsilon_x$, and a maximum number of iterations.



```
$k \leftarrow 0$
while $\|J^t(q_k)\| > \epsilon$ do
    $k \leftarrow k + 1$
                    $\triangleright$ Case of the Gradient-based method
    $q_k \leftarrow q_{k-1} + \alpha J^t(q_{k-1})\left[X_d - f(q_{k-1})\right]$
    $\triangleright$ Case of the Newton-Raphson-based method
    $q_k \leftarrow q_{k-1} + J^{-1}(q_{k-1})\left[X_d - f(q_{k-1})\right]$
end while
$q^* \leftarrow q_{k+1}$
return $(q^*)$
```

Figure 7 – Pseudo-algorithm for computing the inverse geometric model of the robot.

Using $\epsilon_x = 1mm\ and\ k_{max} = 100$, the accuracy and the computational time of the inverse geometric model is analyzed for the both methods in 2 different cases described below, having $\alpha = 1.2$ for the gradient based method.

1) $X_d = X_{di} = (-0.1, -0.7, 0.3)^t$ and $q_{0i} = [-1.57, 0, -1.47, -1.47, -1.47, -1.47]^t$
2) $X_d = X_{df} = (0.64, -0.10, 1.14)^t$ and $q_{0f} = [0\ , 0.8\ , 0\ , 1\ , 2\ , 0]^t$

Knowing that the error is given by $\|X_d - f(q^*)\|$, the following table was obtained through the results

Case	Method	Error (mm)	Computational time (ms)
1	Newton-Raphson	0.2823	7.2524
1	Gradient	0.9967	7.3955
2	Newton-Raphson	0.1568	3.4254
2	Gradient	12.87	15.060

Table 2 – Performance comparation between the Gradient and the Newton-Raphson based method for computing the inverse geometric model.

Therefore, we can see that the error and computation time of the Gradient method is higher in both cases, with different magnitudes. This happens because although the Gradient has a guaranteed convergence, it has a low convergence rate, and it may converge in more iterations than the Newton-Raphson method. The latter has a quadratic convergence rate, but only when the initial condition is close to the solution.

For the following exercises, the Newton-Raphson method will be applied using the Moore-Penrose pseudo-inverse function to invert the Jacobian.

5. Inverse kinematic model

<u>Question 9</u>

First, we computed the intermediate sampled points of a rectilinear trajectory at constant speed $V = 1\ m/s$ between a desired initial point X_{di} and a desired final point X_{df}, both expressed on the task space. The expression for that is given by

$$X_{dk} = X_{di} + k\,T_s\left(\frac{X_{df} - X_{di}}{\|X_{df} - X_{di}\|}\right)\ or$$

$$X_d = (1-p)X_{di} + pX_{df}, with\ p \in [0,1]$$

After that, for each sampled $X_{d,k}$, we performed the inverse geometric model, by calculating the desired coordinate $q_{d,k}$ on the joint space providing as initial configuration the one from the previous step of the iteration $q_{d,k-1}$. Since $q_{d,k-1}$ is close to the solution of problem, then the inversion algorithm converges at a high rate.

Finally, we can analyze the $X(q_d)$ described by the robot by performing the direct geometric model. The result for the given parameters of the Question 9 can be seen on the figure 8 below, and we can verify that the end-effector has described a linear trajectory (blue line) according to expectations.

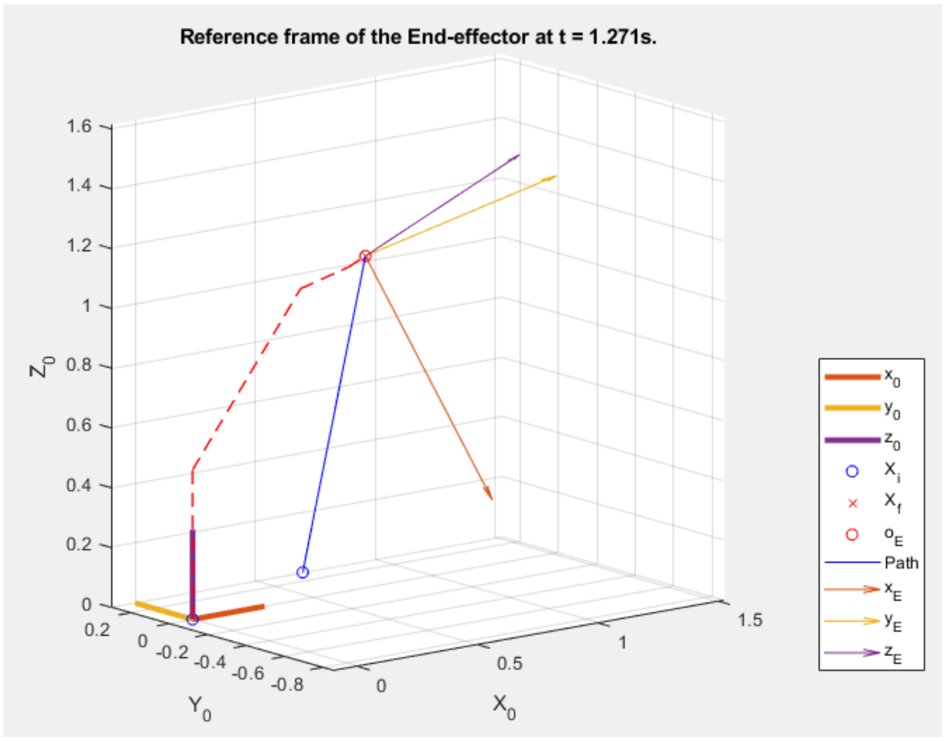


Figure 8 – Trajectory of the robot's end-effector computed by the inverse geometric model of a linear trajectory between two given points.

Question 10

By tracing the temporal evolution of each joint, with its maximum and minimum limit (figure 9), it can be observed that each joint respects its limit, with the exception of joint 5, which slightly exceeds the lower limit.

This is due to the method used to calculate the trajectory which did not take into account the limits of each joint.

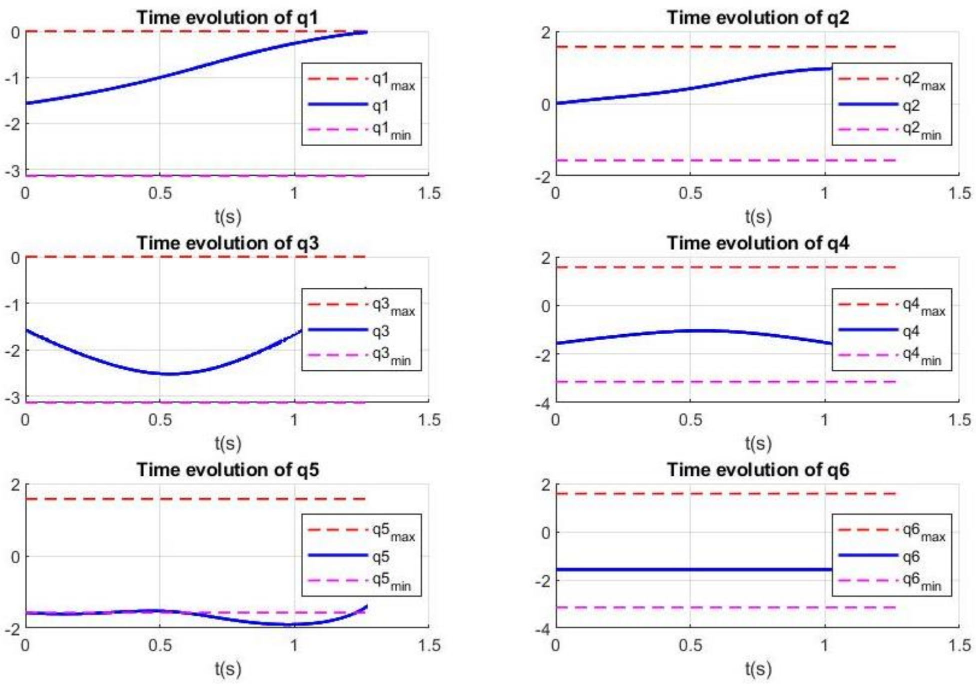


Figure 9 – Temporal evolution of the joints of the trajectory calculated in question 9.

Question 11

To overcome this problem, a secondary task was introduced to the joint space of the robot, aiming at keeping distance from the articular limits q_{min} and q_{max}, by the technique of the gradient projected into the null space of the Jacobian matrix.

The general solution for the quadratic optimization problem is

$$\dot{q}^* = J_W^{\#}\dot{X}_d + \left(I_n - J_W^{\#}J\right)\dot{q}_0$$

Where $W \in \mathbb{R}^{nxn}$ is a positive definite weighting matrix, and $J_W^{\#} = W^{-1}J^t(JW^{-1}J^t)^{-1}$. For this problem, $W = I_n$.

For adding a secondary task into the solution, then the homogeneous solution is given by

$$\dot{q}_0 = -\alpha\nabla_q H(q)$$

The gradient function that repels the q's from their limit positions can be defined by

$$H_{lim}(q) = \sum_{i=1}^{n}\left(\frac{q_i-\bar{q}_i}{q_{i,max}-q_{i,min}}\right)^2, where\ \bar{q}_i = \frac{q_{i,max}+q_{i,min}}{2}. \text{ Thus}$$

$$\nabla_q H_{lim(q)} = \begin{bmatrix} \vdots \\ \dfrac{2(q_i-\bar{q}_i)}{\left(q_{i,max}-q_{i,min}\right)^2} \\ \vdots \end{bmatrix}_{nx1}$$

The framework to compute the new coordinates of q with the secondary task was first to compute the values of q_d the desired trajectory on the joint space with the steps from the previous question. After that, at each time step, the vector $\dot{q}$ was computed with respect to the $\dot{q_0}$ defined above by the gradient function and the Jacobian matrix ${}^{0}J_{vOE}(q_{d,k})$. Finally, with all vectors $\dot{q}$ and the initial values of q, an integrator was used to obtain the new values of q limited.

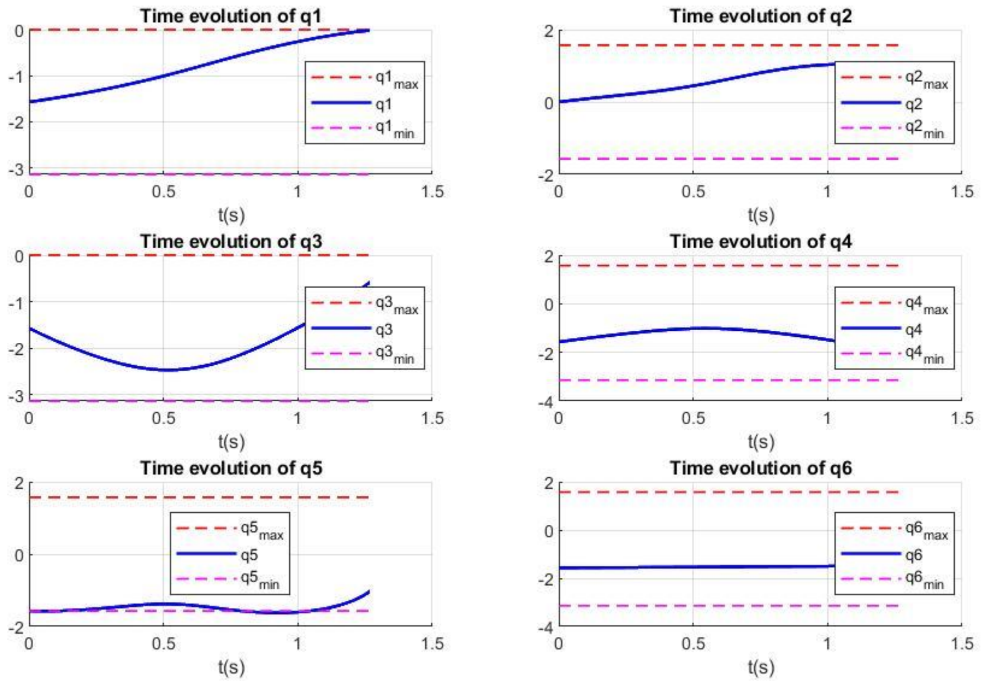


Figure 10 – Temporal evolution of the joints coordinates of the trajectory calculated in question 10 for α=1.

It can be verified that q_5 is now repelled from the $q_{5,min}$, as desired, while the other joint coordinates were maintained similar to the previous result. It was also verified that increasing the value of α repelled even further the joint coordinates from their limit values until a threshold for which the q's would overpass their limits again.

6. Dynamic model

Question 12

In order to compute de Jacobian matrix for the velocity of the center of mass G_i of each link, we use the following formula given in tutorial, and then, we pick the columns from 1 to i from ${}^{0}J_{Gi}$ and we complete with zeros.

$$ {}^{0}J_{Gi} = \begin{bmatrix} I_{3x3} & -{}^{0}\widehat{\overrightarrow{O_E G_i}} \\ 0_{3x3} & I_{3x3} \end{bmatrix} {}^{0}J_{OE} $$

To compute ${}^{0}J_{Gi}$, we calculate ${}^{0}J_{OE}$ which is the Jacobian matrix for the velocity of the end effector by using $ComputeJac(alpha, d, theta, r)$ function, and we also calculate ${}^{0}\overrightarrow{O_E G_i}$ which is the vector from the end-effector to the center of mass G_i using the following relation:

$$ {}^{0}\overrightarrow{O_E G_i} = {}^{0}\overrightarrow{O_0 O_i} + {}^{0}\overrightarrow{O_i G_i} - {}^{0}\overrightarrow{O_0 O_E} $$

These vectors are easily obtained by changing the frames with the transformation matrices and by calculating ${}^{0}\overrightarrow{O_iG_i}$ from ${}^{i}\overrightarrow{O_iG_i}$ which is given.

Question 13

In order to calculate the inertia matrix, we use the following formula obtained from the course, for a poly-articulated chain made of N bodies:

$$A(q) = \sum_{i=1}^{N} \left[m_i \cdot {}^{0}J_{Vg_i}^{t}(q) \cdot {}^{0}J_{Vg_i}(q) \ + \ {}^{0}J_{\omega_i}^{t}(q) \cdot {}^{0}I_i \cdot {}^{0}J_{\omega_i}(q) \right]$$

The masses are known and the terms ${}^{0}J_{Vg_i}(q)$ and ${}^{0}J_{\omega_i}(q)$ are easily obtained thanks to the function $ComputeJacGi$ from the previous question.

To compute de inertia tensor ${}^{0}I_i$ in relation to the frame $\mathcal{R}_{O_0}$from the given ${}^{i}I_{G_i}$ in relation to the frame $\mathcal{R}_{G_i}$, we first use the Huygens theorem to obtain ${}^{i}I_{O_i}$ and then we apply the following relation using the rotational matrices for each frame:

$${}^{0}I_i = {}^{0}R_i \cdot {}^{i}I_{O_i} \cdot {}^{0}R_i^{t}$$

Finally, we add the rotor inertia $r_{red}^{2} \cdot J_m$ of each actuator in the corresponding element of the diagonal of A.

Question 14

The inertia matrix $A(q) \in \mathbb{R}^{n \times n}$ is symmetric, positive definite and upper and lower bounded respectively by μ_1 and μ_2. Since the robot is composed only by revolute joints, μ_1 and μ_2 are constant, which is, they do not depend on any joint configuration, can be defined as the minimal and maximal eigenvalues of A (q), for all allowable value of $q \in [q_{min}, q_{max}]$.

Performing a sweep between the values of value of $q \in [q_{min}, q_{max}]$, through 117.649 points, while the maximum and minimum eigen values of A(q) found were being stored. The obtained bounds for the Inertia matrix of the robot are the ones below:

$$\mu_1 = 0.0577$$

$$\mu_2 = 10.1986$$

Question 15

The gravitational torque is computed using the analytical expression of the gradient of the kinetic energy:

$$G(q) = -\left({}^{0}J_{Vg_1}^{t} \cdot m_1 \cdot g + \cdots + {}^{0}J_{Vg_6}^{t} \cdot m_6 \cdot g \right)$$

Where ${}^{0}J_{Vg_i}$ is computed by the function $ComputeJacGi$ from question 12.

Question 16

Using the same algorithm from question 14, we calculate the gravitational torque for multiple values of $q \in [q_{min}, q_{max}]$, calculate the norm 1 of each configuration and store its maximum value.

For $q_{min} = \begin{pmatrix} -\pi & -\frac{\pi}{2} & -\pi & -\pi & -\frac{\pi}{2} & -\pi \end{pmatrix}^{t}$ and $q_{max} = \begin{pmatrix} 0 & \frac{\pi}{2} & 0 & \frac{\pi}{2} & \frac{\pi}{2} & \frac{\pi}{2} \end{pmatrix}^{t}$, we find a maximal value for $\|G(q)\|_1$ of $116.816\ Nm$.

Question 17

The direct model of the robot arm is computed using the following formula from the course, where Γ is the input, $\Gamma_f(\dot{q})$ is the friction torque, $C(q,\dot{q})$ is due to Coriolis effect and $G(q)$ is the gravitational torque:

$$\ddot{q} = A^{-1} \cdot \left(\Gamma - \Gamma_f(\dot{q}) - C(q,\dot{q}) - G(q)\right)$$

This model is simulated using a Simulink block which receives the external torque as input and returns the positions q and velocities $\dot{q}$ of the joints. This direct dynamic model is shown in the figure below:

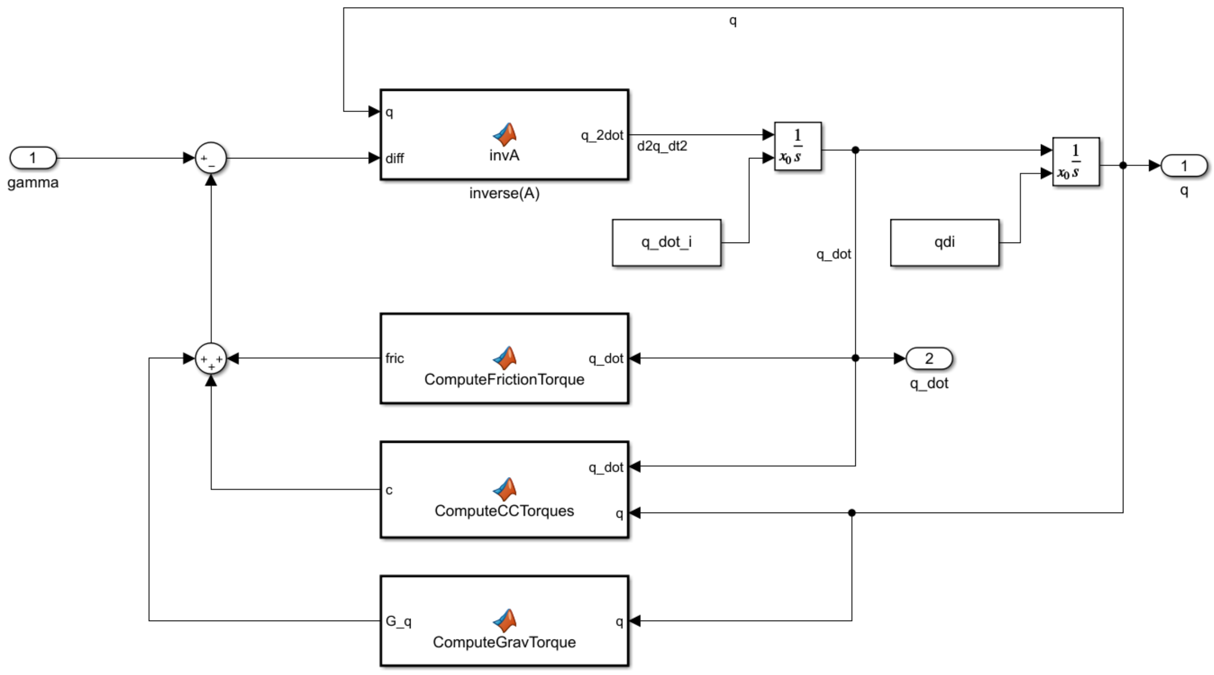


Figure 11 – Direct dynamic model on Simulink

The friction torque is computed by multiplying each velocity $\dot{q}$ with its viscous friction constant F_v. $ComputeCCTorques$ is given, $ComputeGravTorque$ is the same as in question 15 and $invA$ is a Matlab Function that computes the inertia matrix A at a given position q, it computes its inverse and it multiplies by the sum of the torques returning the joint acceleration $\ddot{q}$. Finally, the joint acceleration is integrated twice in order to obtain the velocity $\dot{q}$ and position q.

7. Trajectory generation in the joint space

Question 18

A polynomial trajectory of degree 5 to be followed in the joint space from one given initial position q_{di} to a final one q_{df}, to be reach in minimal time t_f at zero initial and final velocities and accelerations.

To model that, the trajectory can be described by

$$q(t) = q_0 + r(t)D, \text{ where } D = q_f - q_0$$

$$r(t) = 10\left(\frac{t}{t_f}\right)^3 - 15\left(\frac{t}{t_f}\right)^4 + 6\left(\frac{t}{t_f}\right)^5$$

The minimal time t_{fj} for each jth joint to achieve the final position, accounting only the vector k_a of maximum joint accelerations is given by

$$t_{fj} = \frac{10|D_j|}{\sqrt{3}k_{aj}}, \text{ where } k_{aj} = \frac{\tau_{max,j} \cdot r_{red,j}}{\mu_2}$$

With $\tau_{max,j}$ being the maximal torque of joint j, and $r_{red,j}$being the reduction factor of joint j.

Then the minimal global time for the robot to achieve the final point was given by $t_f = max(t_{fj})_{j=1,\cdots,n} = 0.4102$ s.

Question 19

By implementing the trajectory model presented on the previous question for $t_f = 0.5s$ and $q_{di} = [-1;\ 0;\ -1;\ -1;\ -1;\ -1]\ rad$ and $q_{df} = [0;\ 1;\ 0;\ 0;\ 0;\ 0]\ rad$, we have the following trajectories:

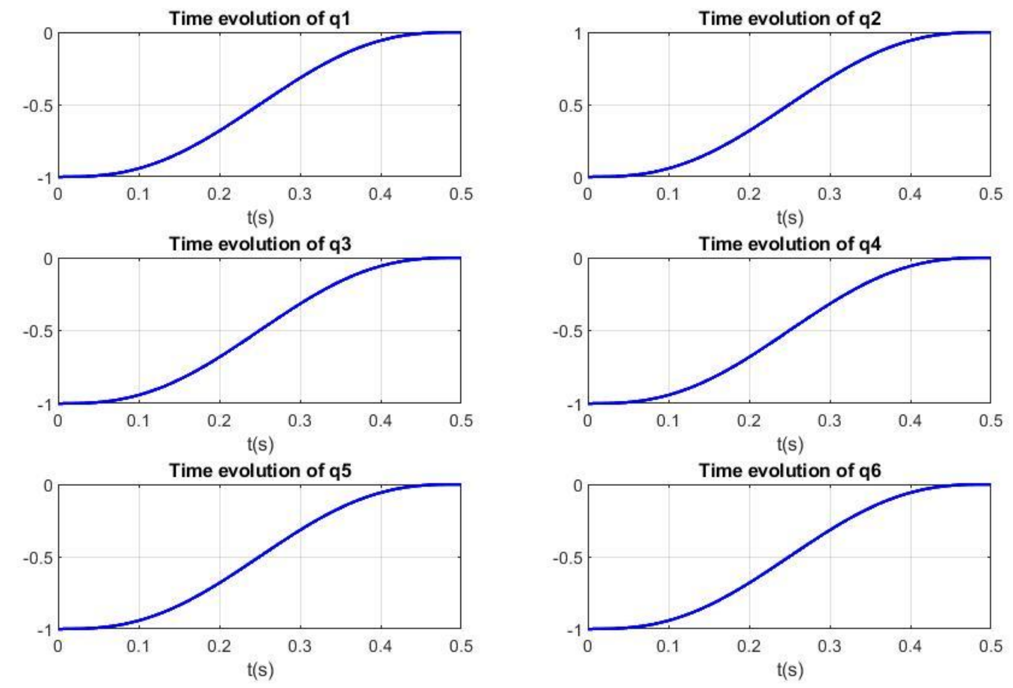


Figure 12 – Calculated trajectories for $t_f = 0.5$ *s.*

8. Joint control law

Question 20

The proposed control for this system is:

$$\Gamma = K_p(q_d - q) + K_d(\dot{q_d} - \dot{q}) + \hat{G}(q)$$

Using Simulink, we implemented the control system as follows, where the $ComputeGravTorque$ is the same function implemented in question 15.

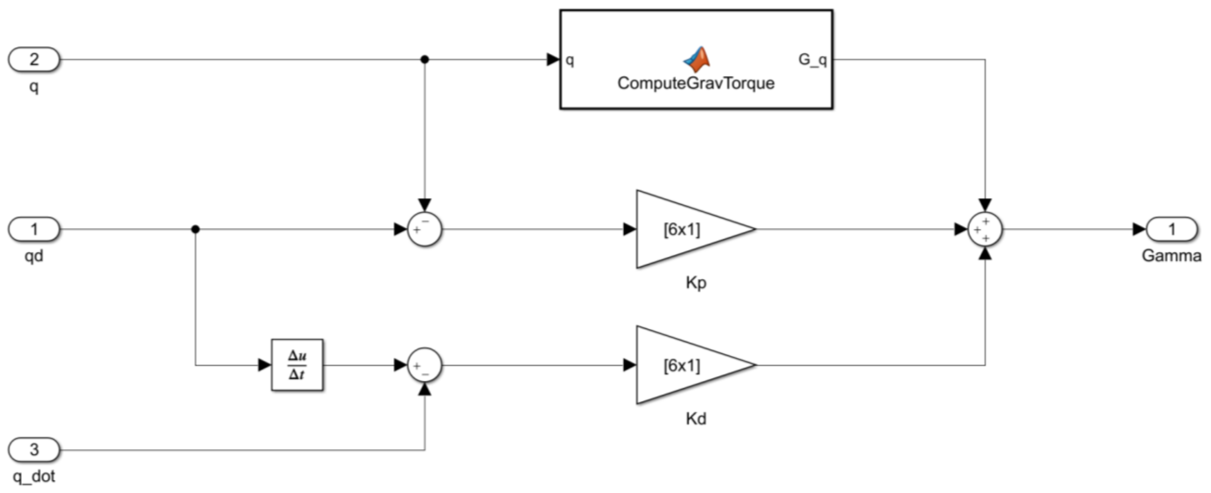


Figure 13 – Block diagram of the robot control system

By assembling all the systems together, we have:

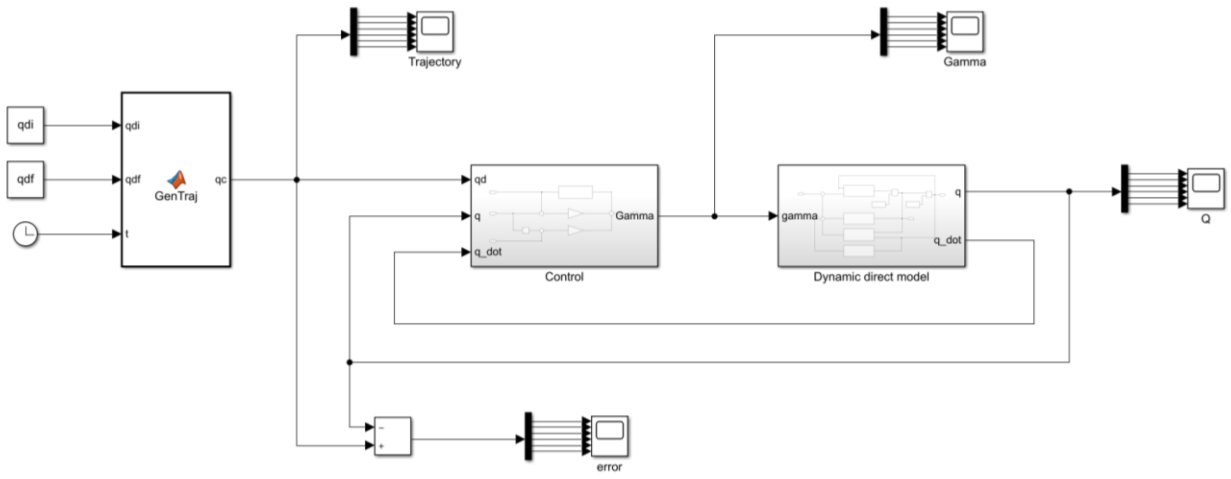


Figure 14 – Direct dynamic model with trajectory generation and control.

By analyzing the error, maximum joint speeds and accelerations, we manually set the control parameters K_p and K_d. We obtained the following gains:

$$K_p = [5000; 8000; 5500; 800; 5500; 2500]$$

$$K_d = [1000; 1000; 500; 100; 100; 500]$$

The figures below show the evolution of position, error and torque at each moment using these control gains.

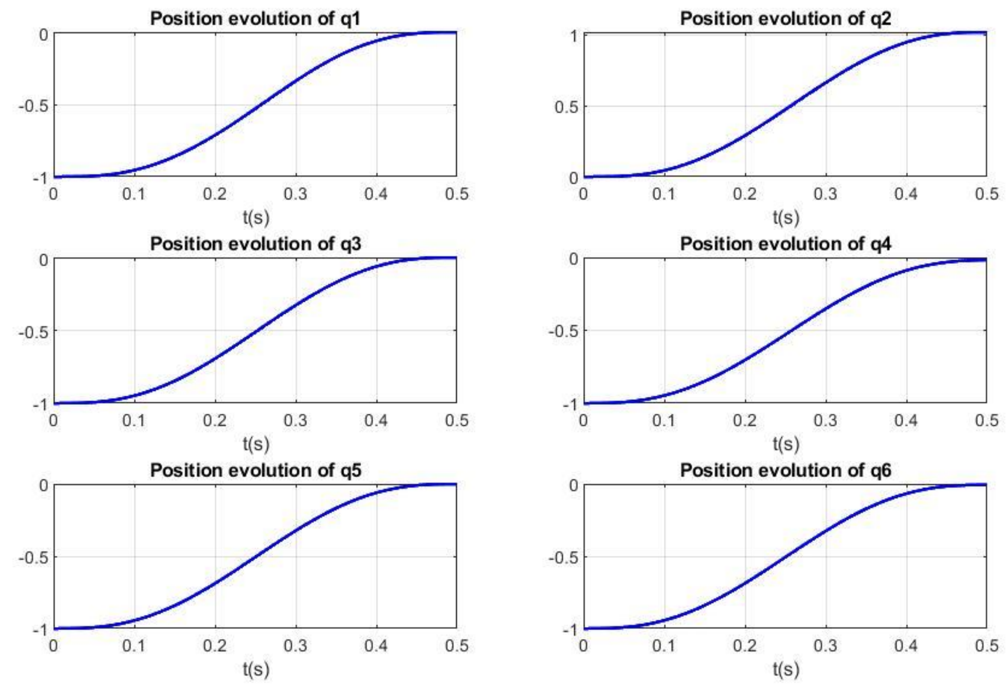


Figure 15 – Position evolution with the proposed K_p and K_d.

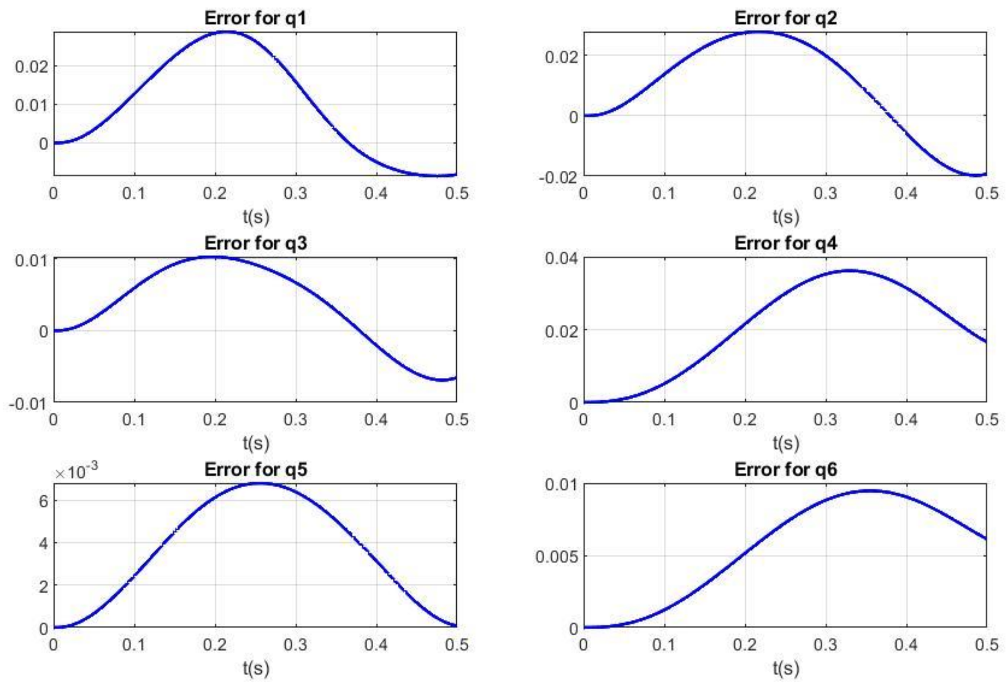


Figure 16 – Error evolution for each joint

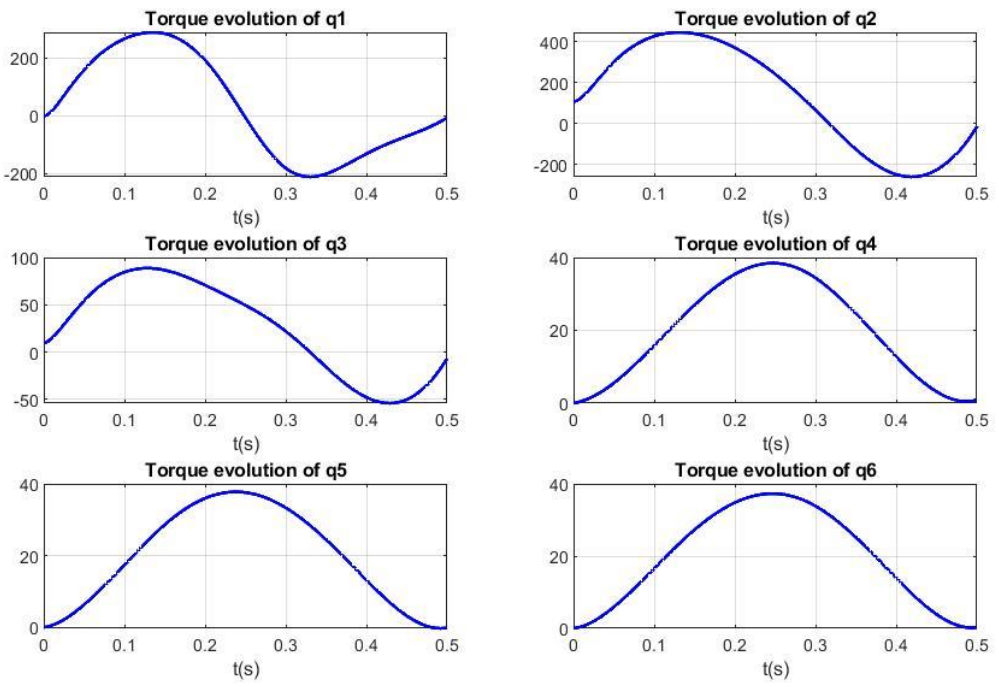


Figure 17 – Torque evolution calculated for each joint.

As can be seen, with these values of K_p and K_d, the trajectory is well followed, and the error remains less than 0.05 rad for each joint during the whole trajectory. The maximal torque is calculated by $\tau_{max_i} \cdot r_{red_i}$, which is $[500, 500, 500, 350, 350, 350]\ Nm$. When we look at the evolution of the torque, it is also between the limits of each joint.