DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE DELL'INFORMAZIONE

SCUOLA POLITECNICA E DELLE SCIENZE DI BASE

FOUNDATIONS OF ROBOTICS

INGEGNERIA DELL'AUTOMAZIONE E ROBOTICA

ANALYSIS AND CONTROL OF A SCARA ROBOT

Students

Daniele Fontana	P38000297
Paola Percuoco	P38000264

Academic Year 2023/2024

Sommario

1 -Introduction

In the following pages, problem of controlling a 4-DOF SCARA manipulator will be addressed.

This is a widely used model, above all in assembly departments: in fact, it can approach a spacework from the above, and it can be very useful for operation demanding high level of precision.

This kind of robot manipulator is provided with 3 revolute joints and 1 prismatic one: so we have 4 joints.

Since n=4, we can consider n+1=5 links, tied to just as many frames.

I can consider an operational space, whose dimension is four so:

$$x_e = \begin{pmatrix} x \\ y \\ z \\ \phi \end{pmatrix}$$

2.1-Direct Kinematic

The goal of direct kinematics is to express the POSE (position and orientation) of the end effector as a function of the joint variables.

The pose of a rigid body with respect to a reference frame is completely described by:

- the position vector of the origin of the frame attached to the body, with respect to the reference frame
- the components of the unit vectors of the frame attached to the body, with respect to the reference frame

Therefore, direct kinematics can be expressed by a transformation matrix such as:

$$T_e^b = \begin{bmatrix} n_e^b & s_e^b & a_e^b & p_e^b \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Which can be computed recursively as:

$$T_e^b = T_0^b (A_1^0 \, A_2^1 \, \ldots\ldots A_n^{n-1}) T_e^n$$

is the transformation matrix between the frames attached to two consecutive links of the manipulator;
the Denavit-Hartenberg convention gives an easy method to compute this matrix.

2.2-Denavit-Hartenberg

In order to find a systematic method to compute the transformation matrix between two consecutive frames, it's necessary to follow some rules to place said frames.

If *axis i* indicates the axis of *joint i* which connects *link i-1* to *link i,* then f*rame i* must be chosen as:

1) z_i is chosen along the axis of joint i+1
2) O_i is placed at the intersection of and the common normal to z_{i-1} and z_i. O'_i is placed at the intersection of z_{i-1} and the common normal.
3) x_i is chosen along the common normal to z_{i-1} and z_i, directed from joint I to joint i+1.
4) y_i is chosen so as to complete a right-handed frame.

Once the frames have been positioned, the transformation matrix between two consecutive frames is completely specified by 4 parameters:

1) a_i is the distance between O_i and O_i'
2) α_i is the angle between z_{i-1} and z_i about axis x_i to be taken positive when rotation is counter-clock wise
3) d_i is the coordinate of O_i' along z_{i-1}
4) θ_i is the angle between x_{i-1} and x_i about axis z_{i-1} to be taken positive when rotation is counter-clock wise

Parameters a_i and α_i are always fixed and depend on the structure of the manipulator, d_i and θ_i can be variable:

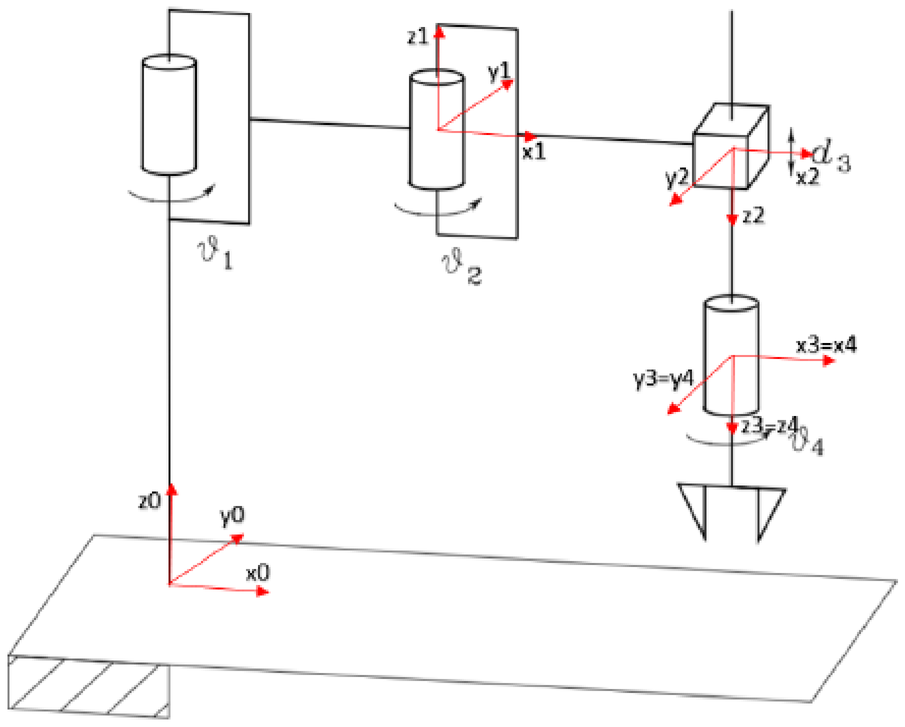
- d_i is variable if joint I is a prismatic joint
- θ_i is variable if joint I is a revolute joint

Hence, each transformation matrix only depends on a single parameter which is the joint variable.

The transformation matrix is:

$$A_i^{i-1}(q_i) = \begin{bmatrix} C\theta_i & -S\theta_i C\alpha_i & S\theta_i S\alpha_i & \alpha_i C\theta_i \\ S\theta_i & C\theta_i C\alpha_i & -C\theta_i S\alpha_i & \alpha_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Using DH, frames can be positioned as in figure, resulting in the set of parameters in the table.



Links	a_i	P_i	d_i	α_i
0	0	0	1	0
1	0.5	P_1	0	0
2	0.5	P_2	0	π
3	0	0	d_3	0
4	0	P_4	0	0

The transformation matrix between the base frame and frame 0 is:

$$T_0^b = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_0 \\ 0 & 0 & 0 & 1 \end{bmatrix};$$

Thereby the direct kinematic is given by:

$$T_4^b = T_0^b * T_4^0 = \begin{bmatrix} C_{12-4} & -S_{12-4} & 0 & a_1C_1 + a_2C_{12} \\ S_{12-4} & C_{12-4} & 0 & a_1S_1 + a_2S_{12} \\ 0 & 0 & -1 & 1 - d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Since the SCARA manipulator has a very simple structure, it's easy to derive a direct kinematics function "k" between the joint variables and the end effector pose specified in terms of a minimal number of parameters:

$$x_e = \begin{bmatrix} p_e \\ \varphi_e \end{bmatrix} = \begin{bmatrix} a_1C_1 + a_2C_{12} \\ a_1S_1 + a_2S_{12} \\ d_0 - d_3 \\ \theta_1 + \theta_2 - \theta_4 \end{bmatrix}$$

2.3-Differential kinematic

The aim of differential kinematics is to find a relationship between the joint velocities and the end effector linear and angular velocity :

$$v_e = J(q)\,\dot{q}$$

Since the trjectory is given in the operational space and the expression for $k(q)$ is available we computed the the analytical Jacobian .

$$J_A(q) = \frac{\partial xe}{\partial q} = \begin{bmatrix} \frac{dx_1}{dq_1} & \cdots & \frac{dx_1}{dq_4} \\ \vdots & \ddots & \vdots \\ \frac{dx_4}{dq_1} & \cdots & \frac{dx_4}{dq_4} \end{bmatrix} = \begin{pmatrix} -a_1S_1 - a_2S_{12} & -a_2S_{12} & 0 & 0 \\ a_1C_1 + a_2C_{12} & a_2C_{12} & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 1 & 1 & 0 & -1 \end{pmatrix}$$

Since our degrees of freedom let the system rotate just around z axis, geometric and analytical Jacobian are equivalent. So:

$$J(q) = J_A(q)$$

2.4-Kinematic singularities

Let's look at the mapping expressed by the Jacobian:

$$v_e = J(q)\,\dot{q}$$

Kinematic singularities are those configurations in which J is not full rank. The determinant of J is:

$$det(J) = a_1a_2$$

So the singularities are given by:

$$\theta_2 = 0; \;\; e\ per\ \theta_2 = \pi$$

Since:

$$dimRange\,(J) = rank(J)$$

$$dimNull\,(J) = n - rank(J)$$

The null space of J is certainly not empty in a singularity.

2.5-Redundancy

If the dimension of the operational space (r) is smaller than the one of the joint space (n), the manipulator is redundant.
In such case J is an (rxn) flat matrix and its rank can be at most equal to r, thus the null space of J is always not empty for a redundant manipulator.

If $\dot{q} \in Null(J)$ $then\ ve = 0$, which means that in the current configuration there are some joints velocities that don't cause any movement on the end effector.

A vector of velocities $\dot{q_0}$ can be added to $\dot{q}$ with no effects on the end effector velocity, as long as it is projected in the null space.

This vector can be used to generate internal motions that enhance the performance of the manipulator.

3-Manipolability ellipsoids

The velocity (force) manipulability ellipsoid represents the way the manipulator transforms joint velocities (torques) into end-effector velocities (forces) in every direction of the operational space.
In the SCARA manipulator, the manipulability is effected only by the first two joints, which are the ones that make the end effector move in the x-y plane. Thus the manipulability analysis is going to consider only a subset of joint velocities (torques) and end effector velocities (forces):

$$\dot{q} = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} \qquad \tau = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix}$$

$$v_e = \begin{bmatrix} \dot{p_e} \\ \dot{\varphi_e} \end{bmatrix} \qquad \gamma = \begin{bmatrix} f_x \\ f_y \end{bmatrix}$$

A reduced version of the Jacobian can be computed as:

$$J_A = \begin{bmatrix} -a_1S_1 - a_2S_{12} & -a_2S_{12} \\ a_1C_1 + a_2C_{12} & a_2C_{12} \end{bmatrix}$$

3.1-Velocity ellipsoids

Let’s consider a set of constant unit norm joint velocities

$$\dot{q}^T\dot{q} = 1$$

This equation describes the points on the surface of a sphere in the joint velocity space. Through differential kinematics, we obtain the quadratic form:

$$v_e^T(J(q)J^T(q))^{-1}v_e^T = 1$$

The equation above describes the surface of an ellipsoid in which:

- The principal axes directions are determined by the eigenvectors of J(q)J’(q)
- The principal axes dimensions are determined by the singular values of J(q)

So the shape of the ellipsoid is clearly a function of the manipulator’s posture.

Along the major principal axis, the velocity is amplified, along the minor axis it is reduced. This means that with the same joint velocities the end effector can reach higher velocities in certain directions than in others.

The volume of the ellipsoid is a manipulability measure, it’s proportional to:

$$\omega(q) = \sqrt{det(J(q)\,J^T(q)}$$

if the manipulator is redundant, or

$$\omega(q) = \det(J)$$

If it isn' t.

When the manipulator is in a singularity ,the manipulability measure is $\omega = 0$ therefore the volume of the ellipsoid is null; the closer the manipulator gets to a singularity the smaller the volume of the ellipsoid becomes.

3.2-Force manipulability ellipsoids

On the basis of the kineto- static duality it is possible to describe the manipulability with reference to forces.

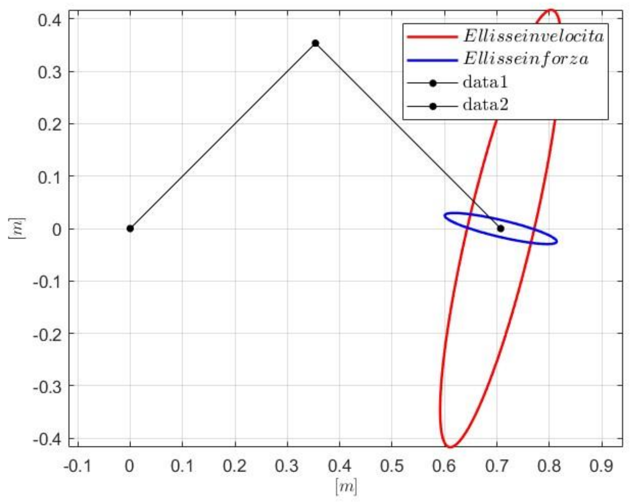
Let's consider a set of unit norm joint torques:

$$\tau^T \tau = 1$$

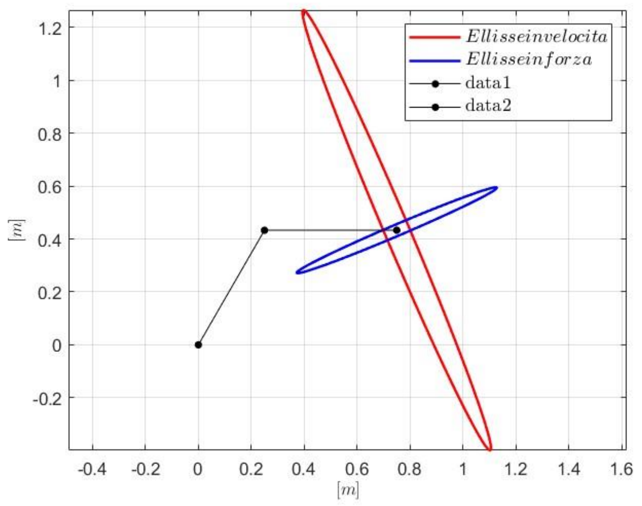
Which leads to:

$$\gamma_e^T (J(q)J^T(q))\gamma_e^T = 1$$

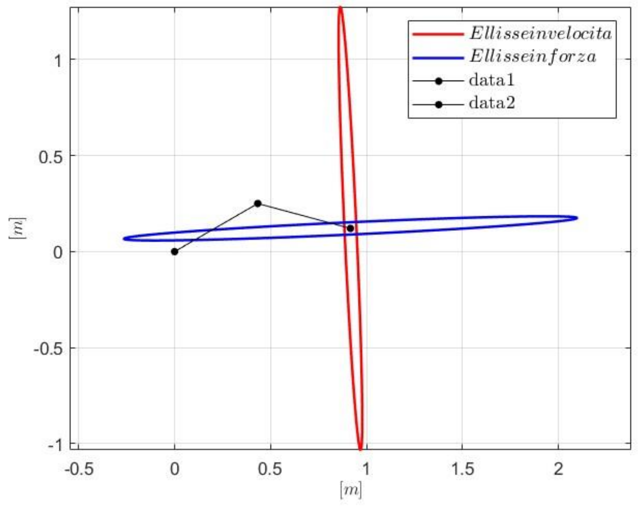
Again, the equation describes the surface of an ellipsoid. It can be observed that the core matrix of the force ellipsoid is the inverse of the one of the velocity ellipsoid. Hence, the two ellipsoids' axis have the same direction, however their dimensions are in inverse proportion. Below are shown the manipulability ellipsoids:



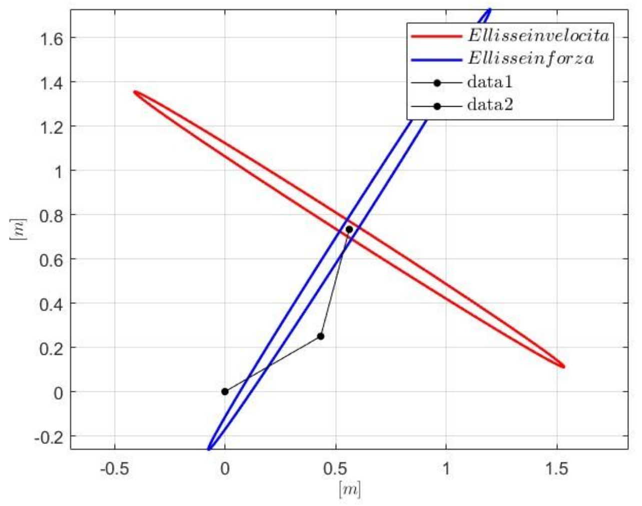
teta=[pi/3 -pi/2]



teta=[pi/6 -pi/4]



teta=[pi/8 -pi/4]



teta=[pi/12 -pi/9]

We can notice that the closer we get to the singularity the more the ellipsoid will collapse degenerating into a line.

4 -Trajectory planning

Once path points are assigned in the operational space, inverse kinematics can be used to interpolate them directly in the joint space, in order to obtain joint references. Although this procedure is simple, nonlinear effects due to direct kinematics could cause the end-effector's configuration to evolve in an unpredictable manner between path points. To avoid this occurrence, path and via points are to be interpolated directly in the operational space, both for end-effector position and orientation. The timing law along the path can be assigned via the arc length function s(t).

The point are the followings:

```
p = [
    0.3, 0.2, 0, 0;
    0.35, 0.2, 0, pi/12;
    0.4, 0.2, 0, pi/10;
    0.4, 0.2, 0.2, pi/8;
    0.4, 0.2, 0.4, pi/6;
    0.45, 0.2, 0.4, pi/4;
    0.45, 0.15, 0.4, pi/2;
    0.45, 0.1, 0.4, 2*pi/3;
    0.45, 0.05, 0.4, pi/2;
    0.4, 0.05, 0.4, pi/4;
    0.4, 0.05, 0.2, pi/6;
    0.4, 0.05, 0, pi/8;
    0.35, 0.05, 0, pi/10;
    0.3, 0.05, 0, pi/12;
    0.25, 0.05, 0, pi/12;
    0.2, 0.05, 0, pi/12;
    0.2, 0.1, 0, pi/10;
    0.2, 0.15, 0, pi/8;
    0.2,0.17,0,pi/8;
    0.3,0.18,0,pi/8;
    0.2, 0.2, 0, pi/6;
    0.22,0.2,0,pi/4;
    0.25, 0.2, 0, pi/2;
    0.3, 0.2, 0, pi;
    0.35, 0.2, 0, pi;]
```

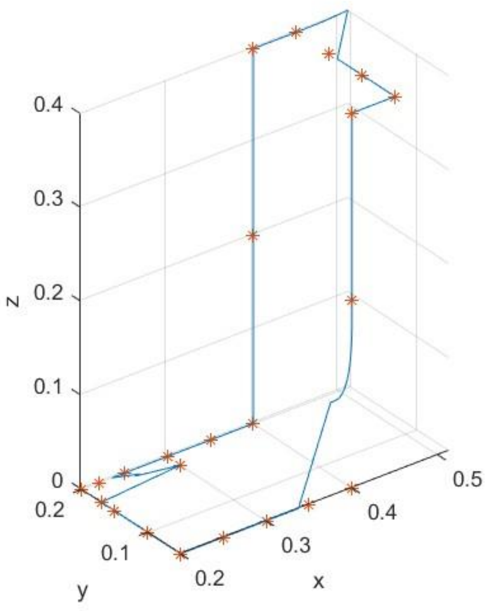
Linear and round traits have been parameterized through the use of *s*(t) arc length, while as regard via points, necessary time to go from v1 and v2 has been established (0.3 s). Instead, necessary time to go from initial via point to the final one is equal to 0.5 s. Moreover, both positions and orientation have been interpolated as it follows:

```
P1=retta(p(1,1:3),p(2,1:3));
P2=retta(p(2,1:3),p(3,1:3));
P3=viapoints(p(3,1:3),p(4,1:3),p(5,1:3),0.6);
P4=retta(p(4,1:3),p(5,1:3));
P5=circon(p(6,1:3),[0.46,0.6,0.4],-pi/20);
P6=retta(p(7,1:3),p(8,1:3));
P7=retta(p(8,1:3),p(9,1:3));
P8=retta(p(9,1:3),p(10,1:3));
P9=retta(p(10,1:3),p(11,1:3));
P10=viapoints(p(11,1:3),p(12,1:3),p(13,1:3),0.8);
P11=retta(p(13,1:3),p(14,1:3));
P12=retta(p(14,1:3),p(15,1:3))
P13=retta(p(15,1:3),p(16,1:3));
```

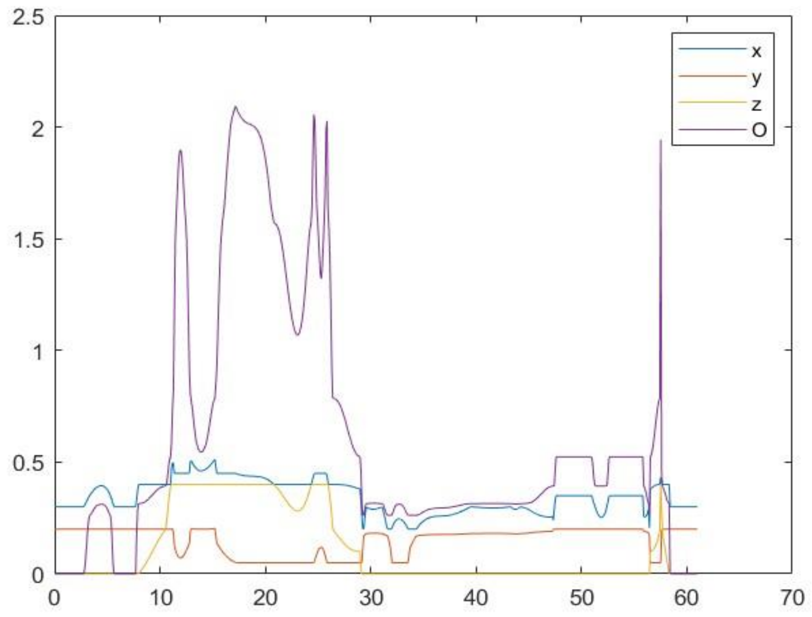
```
P14=retta(p(17,1:3),p(18,1:3));
 P15=retta(p(18,1:3),p(19,1:3));
 P16=retta(p(19,1:3),p(20,1:3));
 P17=viapoints(p(20,1:3),p(21,1:3),p(22,1:3),0.5);
  P18=retta(p(22,1:3),p(23,1:3));
  P19=retta(p(23,1:3),p(24,1:3));
 P20=retta(p(24,1:3),p(25,1:3));
```

As regards position, the arc length function s(t) for each segment of the path was assigned with the spline technique, with a total completion time of 60s. Namely, Matlab's spline() function was used, specifying null endpoint slopes.

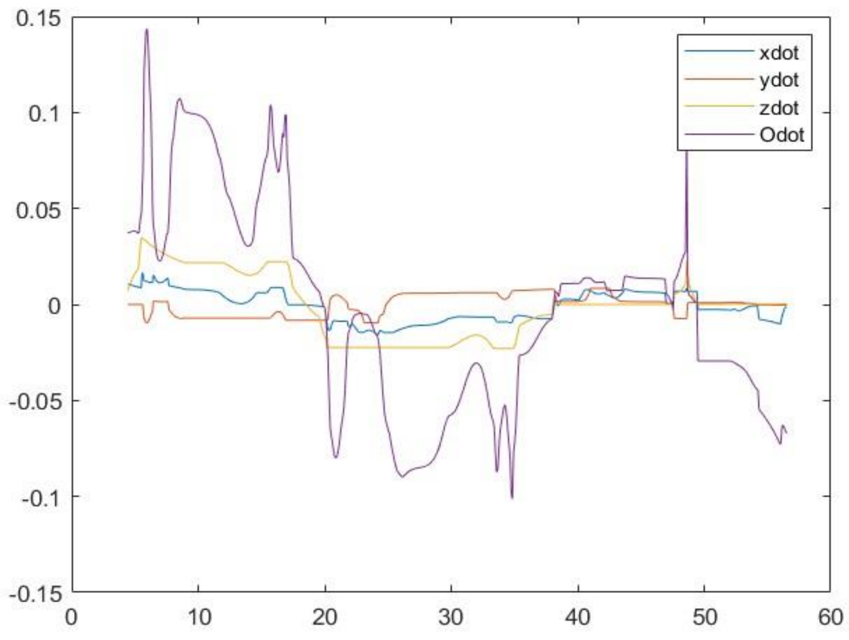
The next figure shows the trajectory:



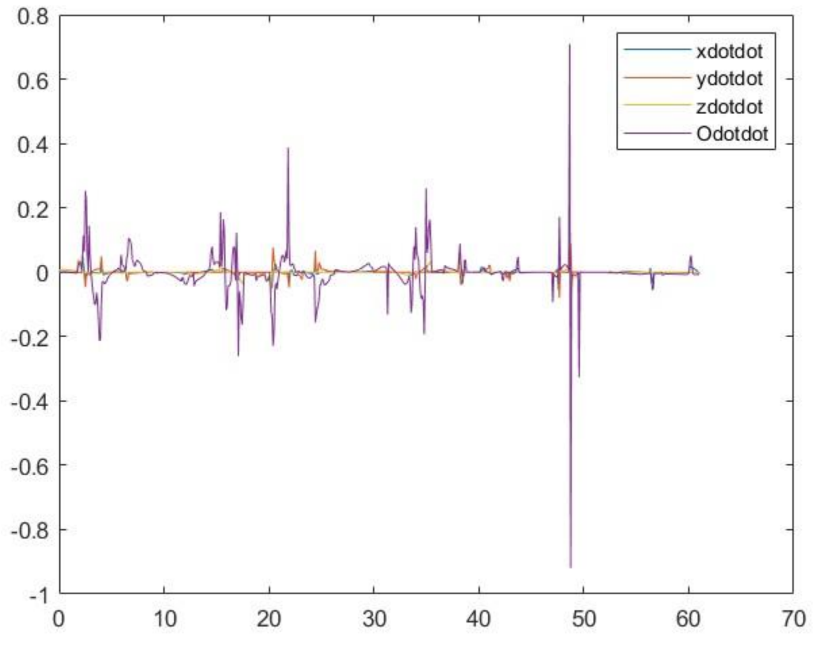
The figure shows the position in operational space:



The figure shows velocity in operational space:



The figure shows acceleration in operational space:



5-Inverse kinematic

The control action is usually applied in the joint space, meaning that the desired pose of the end effector has to be transformed in a sequence of desired joint variables values.

Inverting the direct kinematics function is not easy because the relationship between the operational space variables and the joint space variables is non-linear. On the other hand, the differential kinematics equation gives a linear mapping between the velocities in the two spaces, so the easiest thing to do is to compute $\dot{q}$ as:

$$\dot{\mathrm{q}} = J^{-1}(q)v_e$$

and then getting q via a numerical integration as:

$$q(t_{k+1}) = q(t_k) + J^{-1}\big(q(t_k)\big)v_e(t_k)\Delta t$$

However, each iteration of the numerical integration is affected by an error which causes a drift in the longrun.

This means that the computed q does not correspond to the desired end effector pose anymore.

This problem can be solved implementing an algorithm that works on the error between the desired pose and the actual pose in the operational space:

$$e = x_d - x_e$$

If $\dot{q}$ is chosen appropriately the error converges to 0.

5.1-Jacobian inverse

Choosing:

$$\dot{q} = J_A^{-1}(q)(\dot{x}_d + Ke)$$

Leads to:

$$\dot{e} + Ke = 0$$

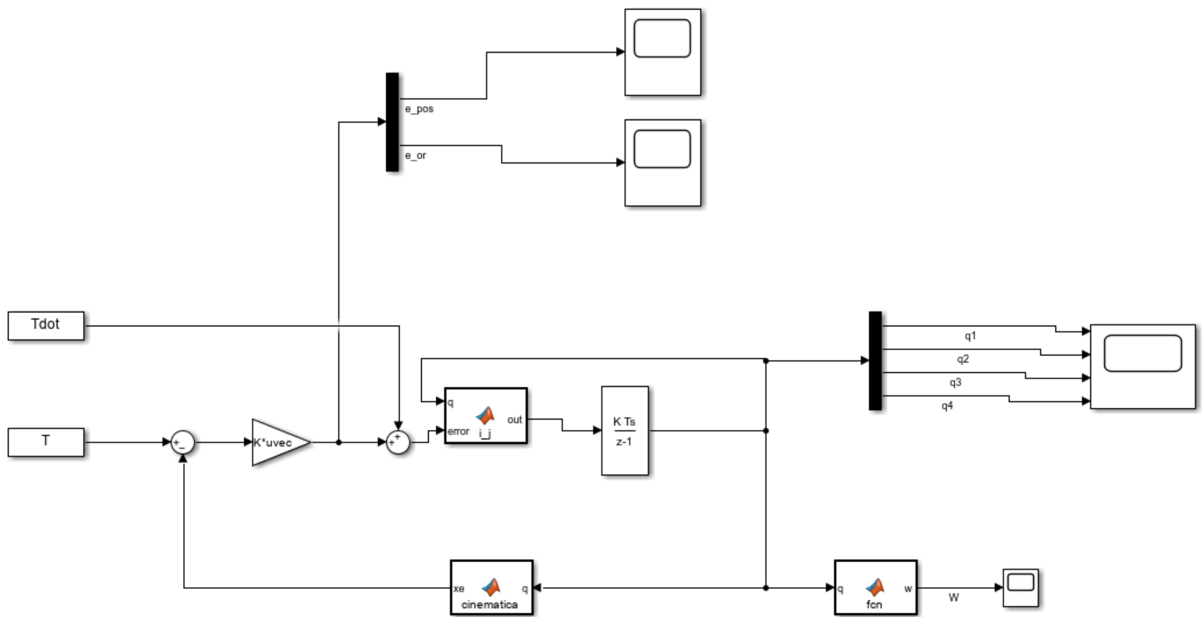
Meaning that the error converges to 0 if K is a positive definite matrix.

The matrix K is responsible for the convergence rate of the error: the higher the eigenvalues of K the faster the convergence, so one would be inclined to choose K as big as possible. However, the norm of K has an upper bound related to the sampling time in order to maintain the stability of the system.

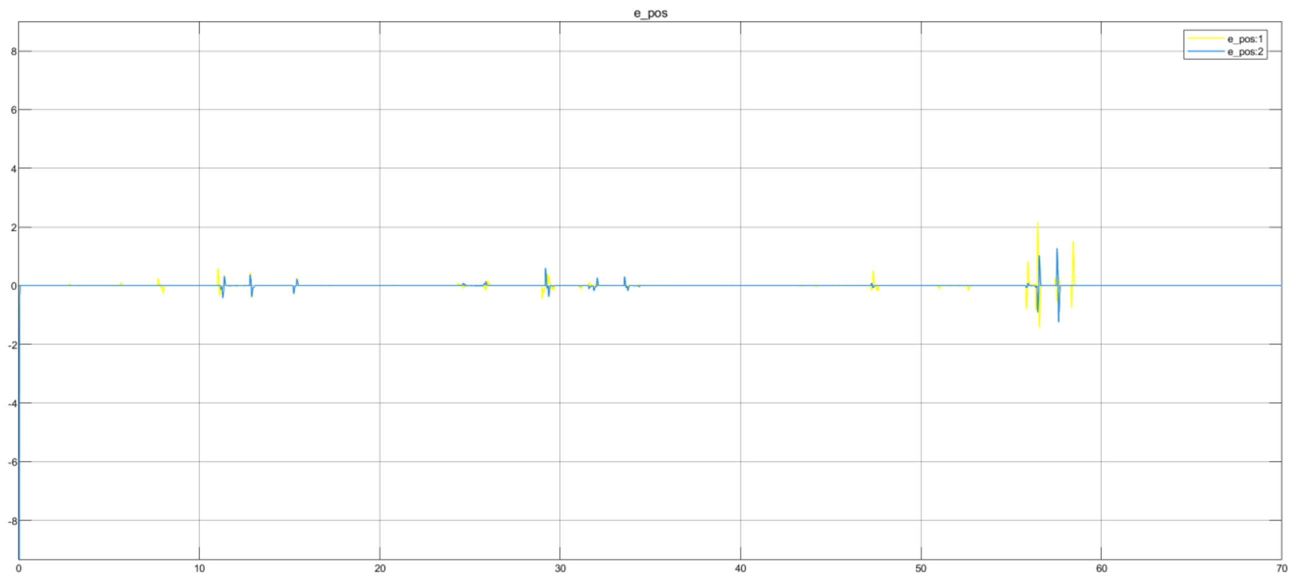
Hence we choose :

$$\mathrm{Kp} = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 100 \end{bmatrix};$$

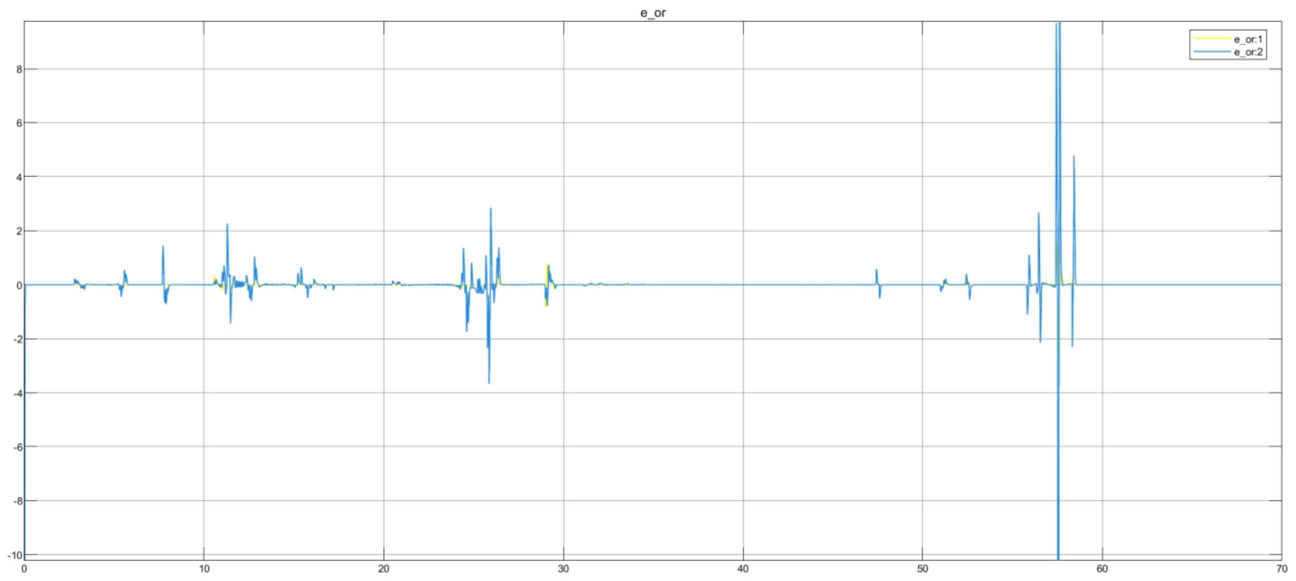
The next figure shows the block scheme of the algorithm:



The next figure shows the position error:



The next figure shows the orientation error:



5.2-Jacobian transpose

In the previous method, thanks to the specific choice of $\dot{q}$, the error dynamics was linearized, as it was described by a linear differential equation.
This approach keeps a non-linear error dynamics instead, making the algorithm computationally easier; however the stability of the system now has to be proved via the Lyapunov method.

Let's choose the Lyapunov candidate function as:

$$\frac{1}{2}e^T ke > 0; \quad \forall\, e \neq 0$$

Which leads to:

$$\dot{v} = e^T k\dot{xd} - e^T k e J_A \dot{q};$$

Choosing $\dot{q}$ as:

$$\dot{q} = J_A^T k e$$

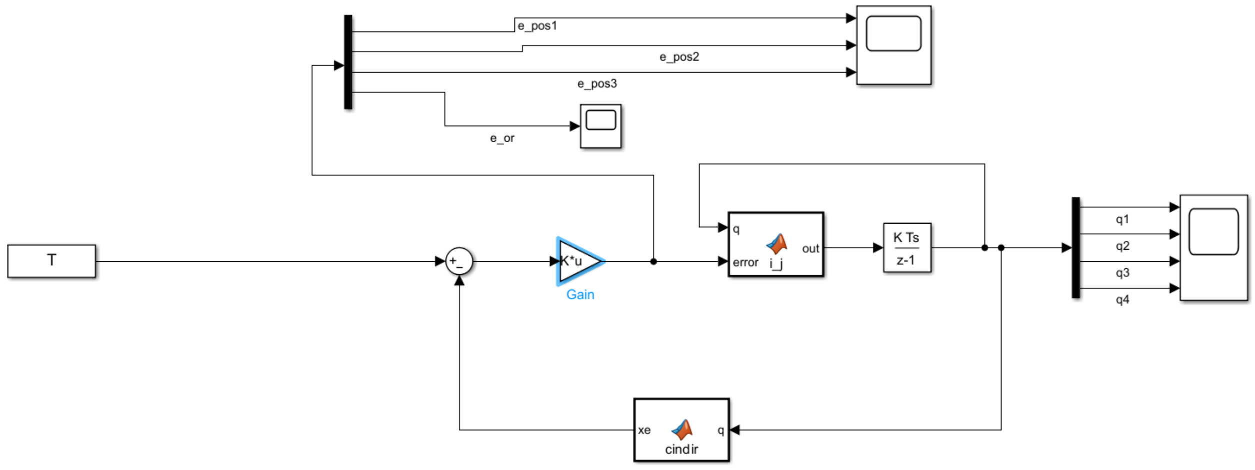
Gives:

$$\dot{v} = e^T k\dot{xd} - e^T k e J_A J_A^T k e$$

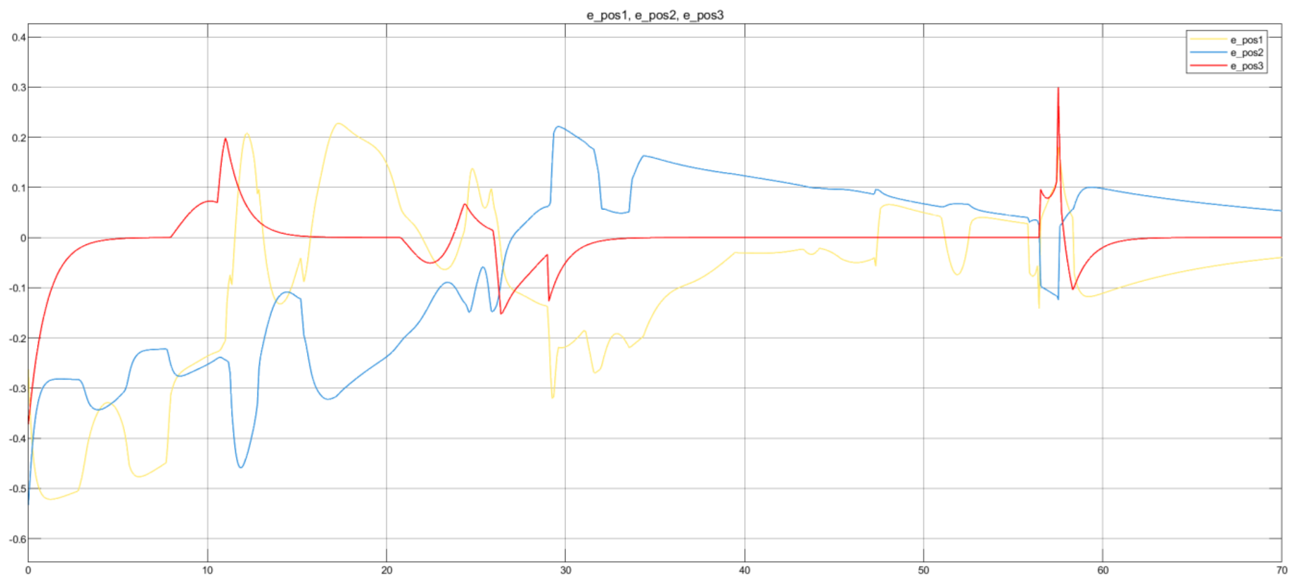
The second term of right-end side of the previous equation is always positive, as it is a quadratic form, thus the stability of the system only depends on the first term.
Nothing can be said about the sign of the first term, so the asymptotical stability is not ensured; choosing a larger K could reduce the norm of the error, however it could also cause instability.
So we choose:

$$K = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix};$$

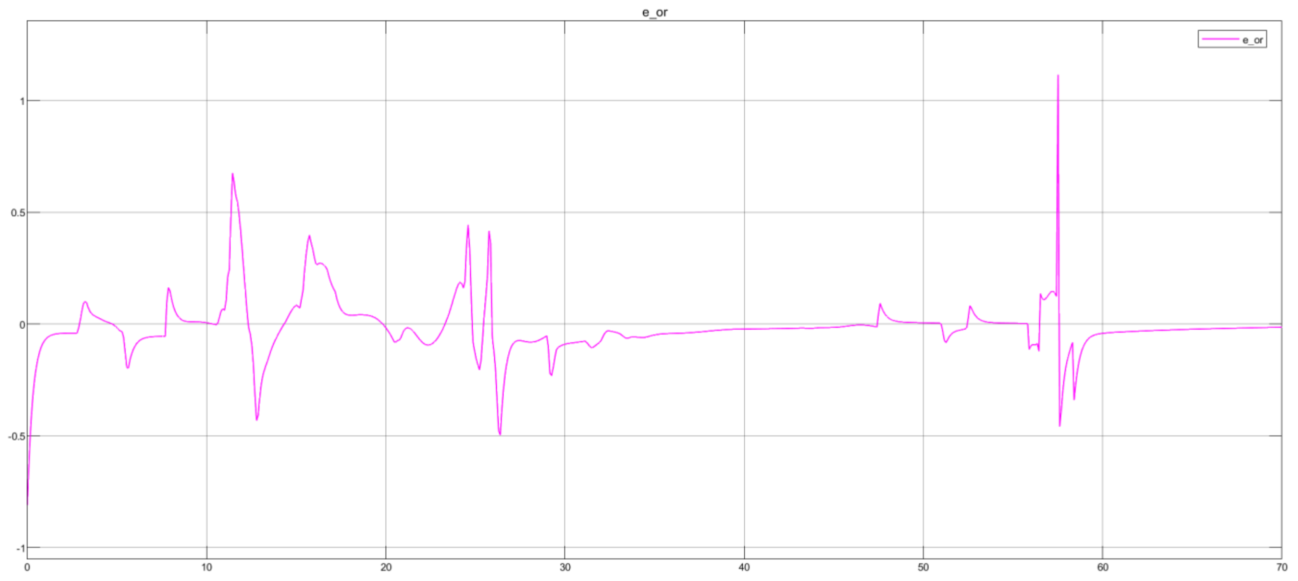
The next figures show the scheme block:



The next figure shows the position error:



The next figure shows the orientation error:



5.3-Jacobian pseudo inverse

Manipulator, although non-redundant, can be made functionally redundant by planning a trajectory which involves only a reduced number of operational space components.

Redundancy ensure a more dexterous robotic system and the capability to avoid singular configurations or obstacles. To do this we relax one of the operational space components while trying to maximize manipulability measure (optimize a dexterous constraint).

We choose to relax x axis component (y, z and ϕ will remain).

By doing that, Jacobian becomes a 3x4 matrix (reduced Jacobian), earned from deleting its first raw.

So the Jacobian became :

$$J_A(q) = \begin{pmatrix} a_1\cos(\theta_1) + a_2\cos(\theta_1+\theta_2) & a_2\cos(\theta_1+\theta_2) & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 1 & 1 & 0 & -1 \end{pmatrix}$$

The Jacobian matrix is a low-rectangular matrix, whose determinant is zero, so we can't define the inverse matrix but only the pseudo inverse.

The pseudo-inverse matrix is

$$\dot{q} = J^{\dagger}(q)v_e$$

where:

$$J^{\dagger}(q) = J^T(JJ^T)^{-1}$$

In this case, solution is made up of 2 terms: the first one is represented by the previous solution of the Jacobian inverse scheme, while the other one is represented by an internal motion allowed by the matrix:

$$P = I - J^{\dagger}J,$$

that project the vector $\dot{q}_0$ in the null space of J.

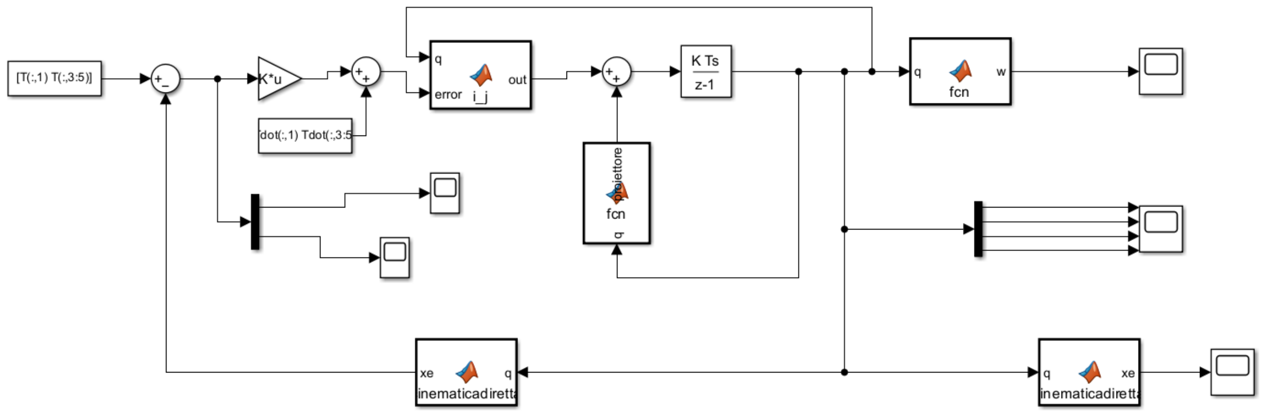
The equation becomes:

$$\dot{q} = J^{\dagger}(q)(\dot{x}_d + Ke) + (I - J^{\dagger}(q)J(q))\dot{q}_0$$

K is choosen as:

$$Kp = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The next figure shows the scheme of the pseudo-inverse CLIK algorithm:



The $\dot{q}_0$ was chosen as to maximize the manipulability measure:

$$\dot{q}_0 = k_0\left(\frac{\partial \omega(q)}{\partial q}\right)^T$$

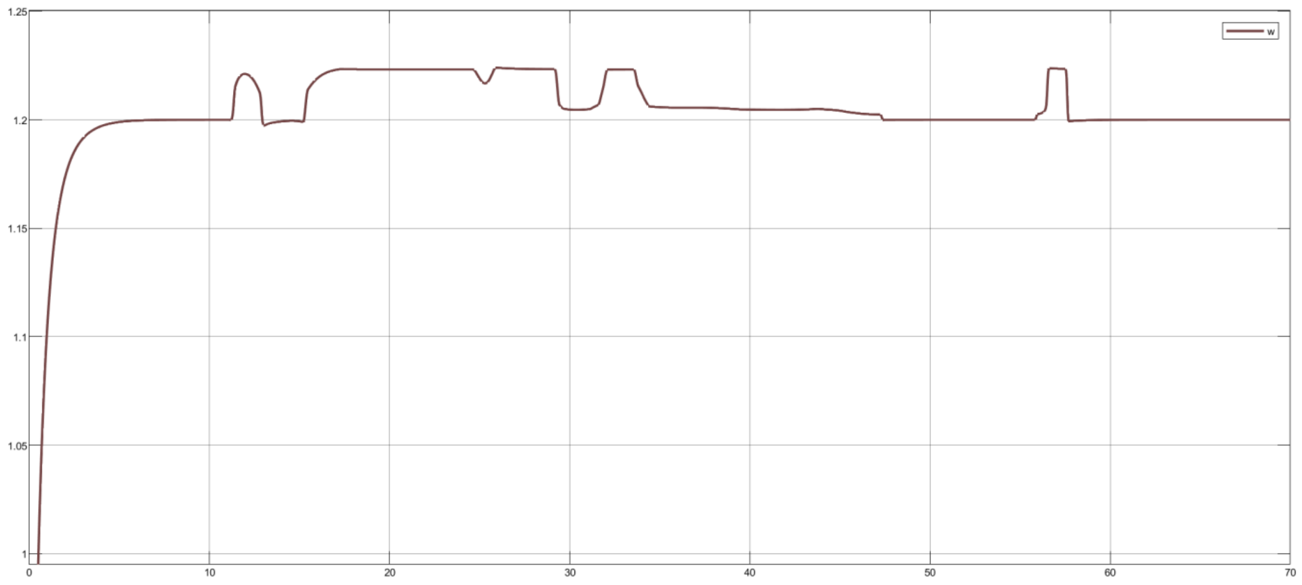
Our goal is to maximize a specific w(q) function, representing the manipulability measure of the SCARA, compatibly with the primary task:

$$\omega(q) = \sqrt{\det(J(q)J^T(q))} = \sqrt{2a_1^2\cos(\vartheta_1)^2 + 2a_1a_2\cos(\vartheta_1)\cos(\vartheta_1 + \vartheta_2) + 2a_2^2\cos(\vartheta_1 + \vartheta_2)^2}$$
$$= \sin^2\theta_2$$

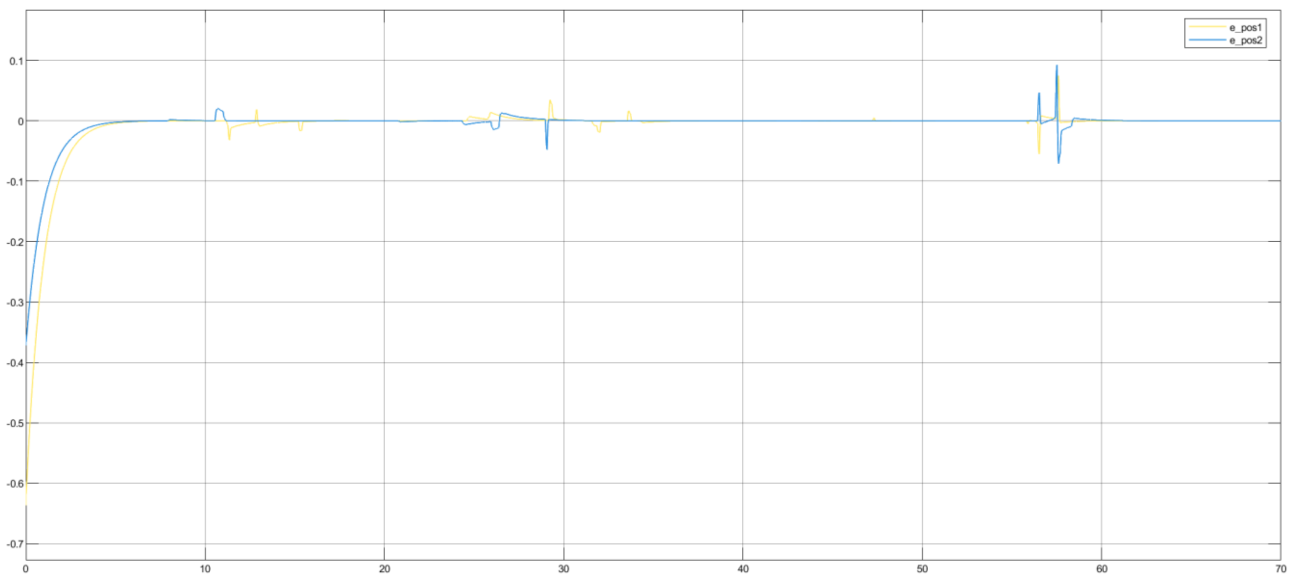
While still satisfying the differential kinematics constraint $\dot{x}_e = J_A(q)\dot{q}$, the resulting joint space trajectory will steer clear of singular configurations:

$$\vartheta_2 = 0, \pi$$

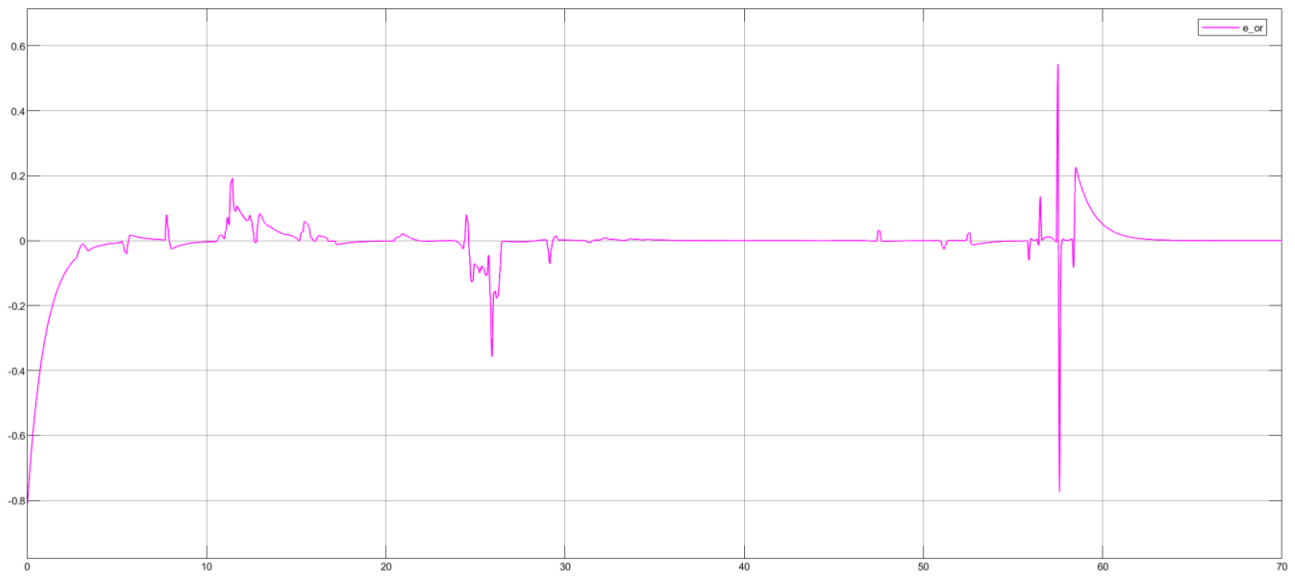
The next figure shows the manipulability mesure:



The next figure shows the position error:



The next figure shows the orientation error:



5.4-Second order CLIK

If the desired trajectory is specified in terms of position, velocity and acceleration a second-order inverse kinematics algorithm is needed.
Differentiating the differential kinematics equation gives:

$$\ddot{x}_e = J_A(q)\ddot{q} + \dot{J}_A(q,\dot{q})\dot{q}$$

Which can be inverted as:

$$\ddot{q} = J_A(q)^{-1}\left(\ddot{x}_e - \dot{J}_A(q,\dot{q})\dot{q}\right)$$

A double numerical integration gives q, but it's still affected by the drift problem. Similarly as done for the first order algorithms, let's define:

$$\ddot{e} = \ddot{x}_d - \ddot{x}_e = \left(\ddot{x}_d - J_A(q)\ddot{q} - \dot{J}_A(q,\dot{q})\dot{q}\right)$$

Choosing $\ddot{q}$ as:

$$\ddot{q} = J_A(q)^{-1}\left(\ddot{x}_d + K_D\dot{e} + K_P e - \dot{J}_A(q,\dot{q})\dot{q}\right)$$

Gives:

$$\ddot{e} = K_D\dot{e} + K_P e$$

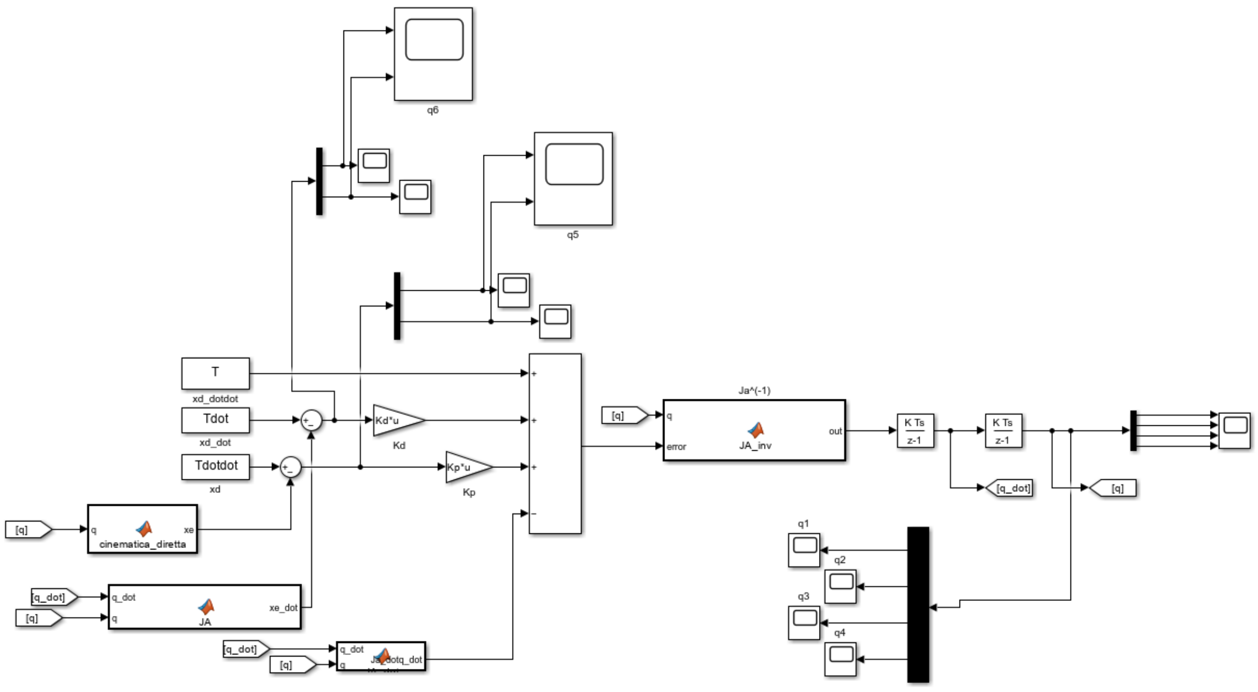
Therefore the error system is asymptotically stable and the error converges to zero for KD, KP positive definite.
K_P was chosen as:

$$K_P = \begin{bmatrix} 50 & 0 & 0 & 0 \\ 0 & 50 & 0 & 0 \\ 0 & 0 & 50 & 0 \\ 0 & 0 & 0 & 50 \end{bmatrix}$$

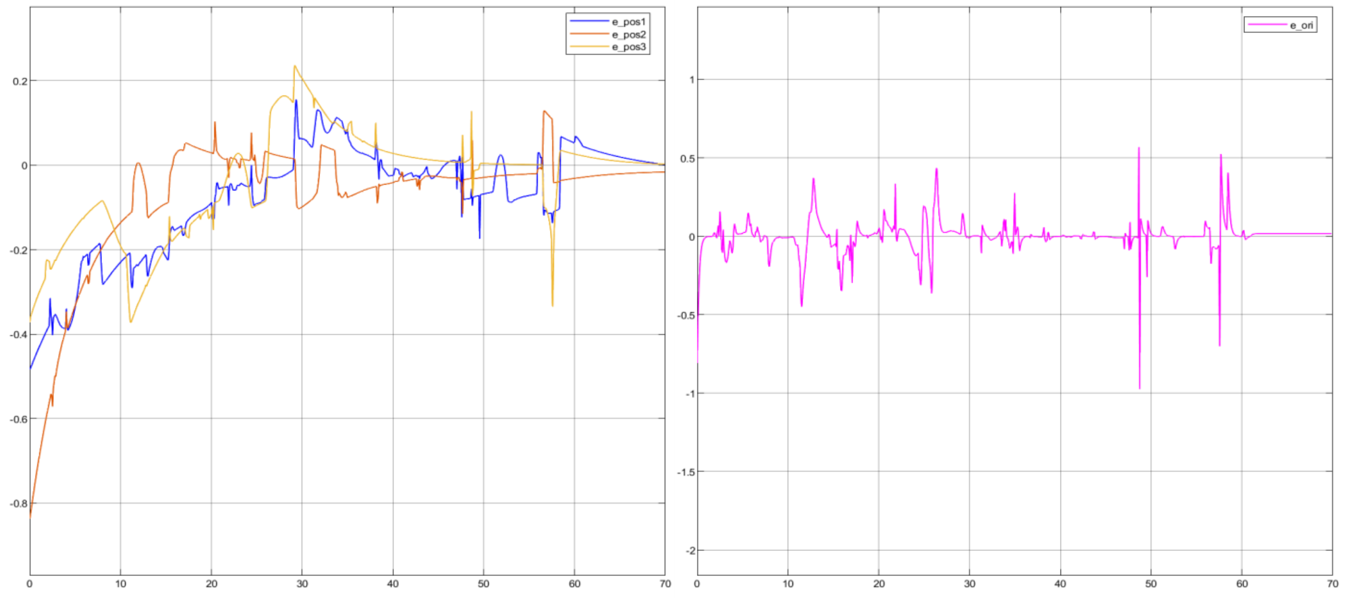
K_D was chosen as:

$$K_P = \begin{bmatrix} 150 & 0 & 0 & 0 \\ 0 & 150 & 0 & 0 \\ 0 & 0 & 150 & 0 \\ 0 & 0 & 0 & 150 \end{bmatrix}$$

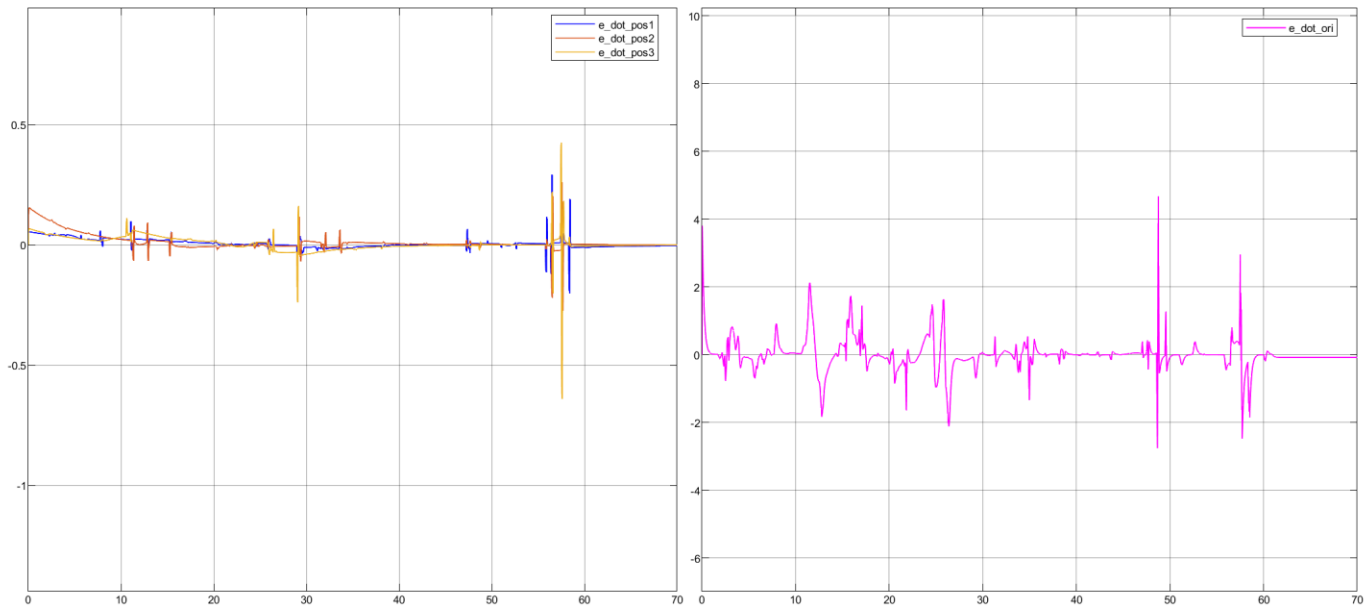
The next figure shows the scheme block:



The figure shows the position error:



The figure shows the velocity error:



6-Motion control

The inverse kinematics algorithms gave the time sequence of the joints positions, velocities and accelerations needed to make the end effector follow the desired trajectory, now it’s time to compute the generalized torques to be applied at the joints; this operation goes under the name of motion control

Some of motion control techniques analyzed belongs to joint space, and for this reason kinematic inversion is necessary: in this way we can earn reference variables starting from the trajectory created in the operational space. Second order CLIK algorithm is the chosen one, and we will apply the following control techniques:

- Robust control
- Adaptive control
- Operational space inverse dynamics control, with the adoption of an integral action to recover the steady-state error to an uncompensated load

6.1-Dynamic model

The dynamic model gives the relationship between the torques applied to the joints and the motion of the structure, hence the first step for motion control is to determine the dynamic model of the manipulator.
The model can be written in a compact form as:

$$B(q)\ddot{q} + C(q,\dot{q})\dot{q} + F_v\dot{q} + g(q) = \tau$$

Where:

- B(q) is the inertia matrix computed as:

$$\textstyle\sum_{i=1}^{n} ml_i J_P^{T(li)} J_P^{(li)} + J_O^{T(li)} R_I I_{li}^{i} J_O^{(li)} + m_{mi} J_P^{T(mi)} J_P^{(mi)}\ J_O^{T(mi)} R_I I_{mi}^{i} J_O^{(mi)}$$

- $C(q, q)$ accounts for the Coriolis and the centrifugal effects and it’s not uniquely defined, in this case it was computed using the Christoffel symbols of the first type:

$$C_{ij} = \sum_{k=1}^{n} c_{ij} q_k$$

With:

$$c_{ijk} = \frac{1}{2}\left(\frac{\partial b_{ij}}{\partial q_k} + \frac{\partial b_{ik}}{\partial q_j} - \frac{\partial b_{jk}}{\partial q_i}\right)$$

- Fv is a diagonal matrix that accounts for the viscous friction of the motors:

$$F_v = \begin{bmatrix} {k_{r1}}^2 f_{m1} & 0 & 0 & 0 \\ 0 & {k_{r2}}^2 f_{m2} & 0 & 0 \\ 0 & 0 & {k_{r3}}^2 f_{m3} & 0 \\ 0 & 0 & 0 & {k_{r4}}^2 f_{m4} \end{bmatrix};$$

With:

- k_{ri} gear reduction ratio of motor i
- f_{mi} viscous friction coefficient of motor i

- (q) is the vector of the moments generated by gravity on each joint at the current configuration. In the SCARA manipulator only the third joint is effected by gravity therefore the vector is:

$$F_v = \begin{bmatrix} 0 \\ 0 \\ -9.8(m_{l3} + m_{pl}) \\ 0 \end{bmatrix}$$

With:

$$m_{pl} = 3$$

6.2-Robust control

It applies mostly when we can't have a perfect knowledge about our dynamc model.

In this case of imperfect compensation, it is reasonable to assume a control vector expressed by:

$$u = \hat{B}(q)y + \hat{n}(q, \dot{q})$$

Where $\hat{B}$ and $\hat{n}$ represent the adopted computational model in terms of estimates of the terms in the dynamic model.

Uncertainty on the estimatesi s represented by:

$$\tilde{B} = \hat{B} - B$$

$$\tilde{n} = \hat{n} - n$$

Using this kind of nonlinear control law u, gives:

$$y = \ddot{q}_d + K_D\dot{\tilde{q}} + K_P\tilde{q} + w$$

Where:

$$z = D^T Q \xi$$

Given the position:

$$\xi = [\tilde{q} \quad \dot{\tilde{q}}]^T$$

Where:

$$\tilde{q} = q_d - q$$

$$D = \begin{bmatrix} 0 \\ I \end{bmatrix}$$

$$\text{H}=\begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix}$$

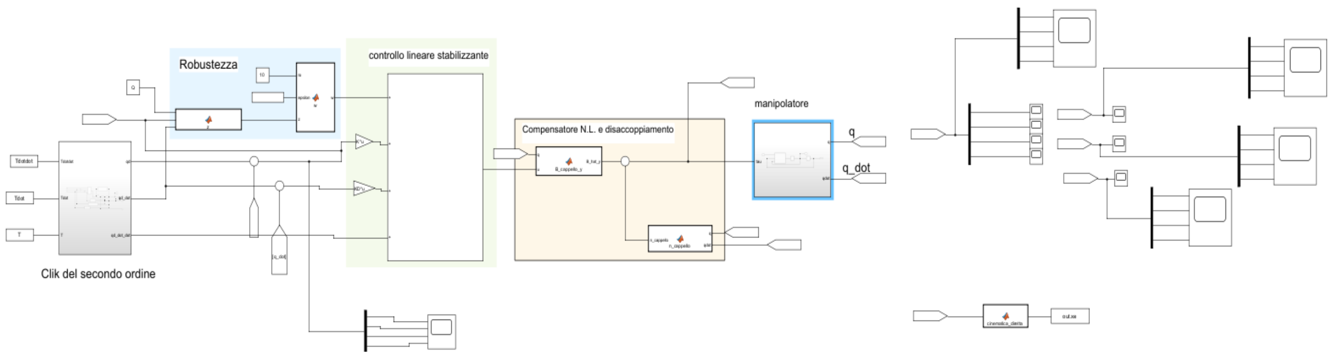
Appling Lyapuanov formula the solution is:

$$w = \frac{\rho}{\|z\|} z \quad ; \; \rho > 0 \;\; ; \; \|z\| > \varepsilon \text{ else } w = \frac{\rho}{\varepsilon} z$$

To sum up, this control law will show three components:

1. The term that ensures an approximate compensation of non linear effects and joint decoupling
2. The term that introduces a linear feedforward action and linear feedback action which stabilizes the error system dynamics
3. The term that represents the robust contribution that counteracts the indeterminacy

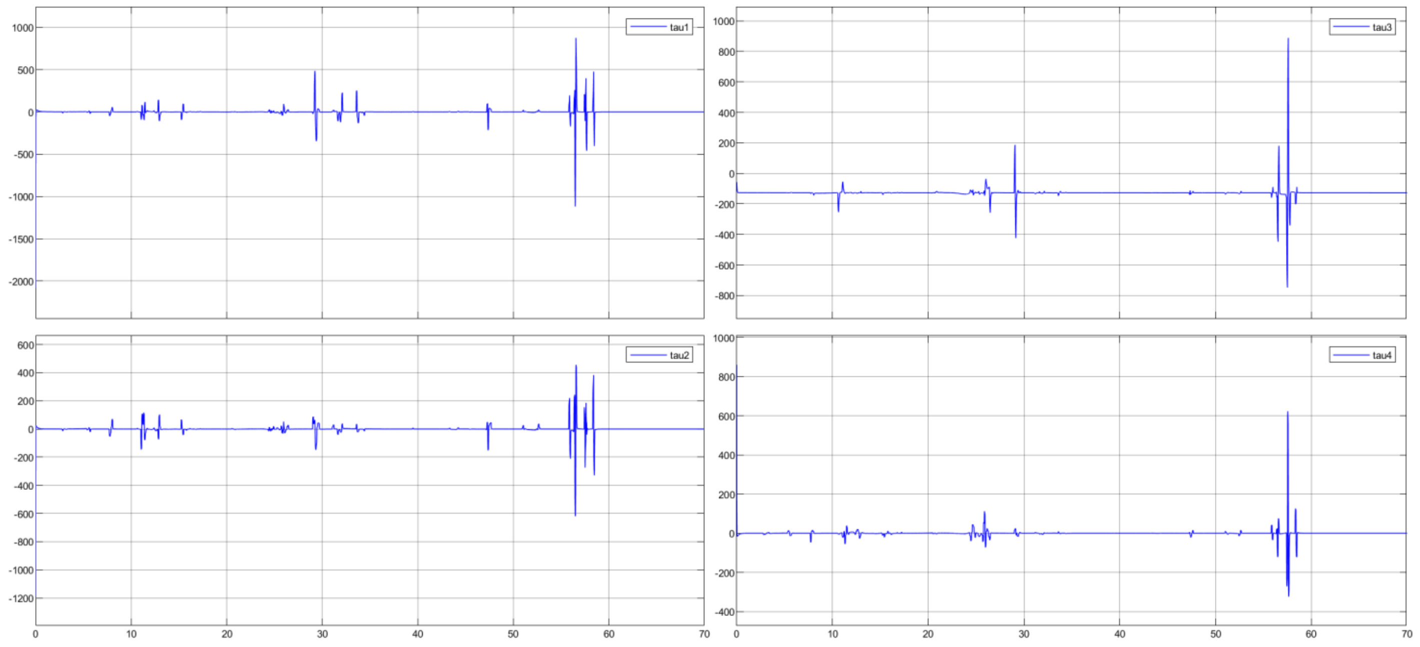
The figure shows the robust control scheme:


Robustezza
controllo lineare stabilizzante
Compensatore N.L. e disaccoppiamento
manipolatore
q
q_dot
Clik del secondo ordine

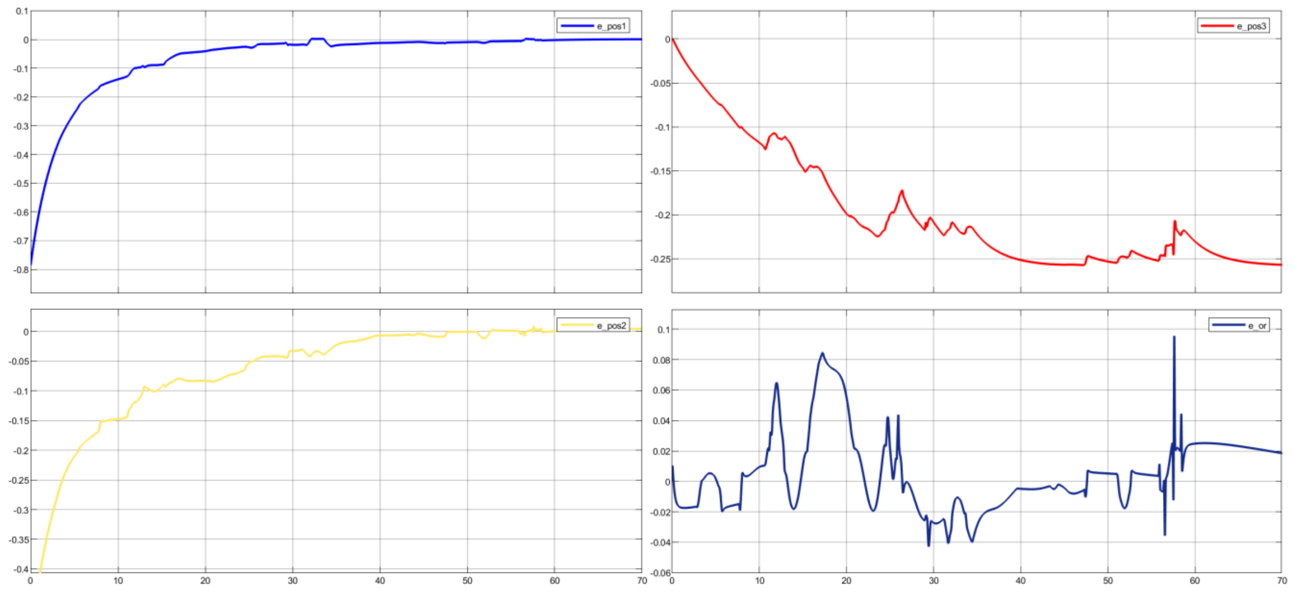
We used:

$$K_D = \begin{bmatrix} 150 & 0 & 0 & 0 \\ 0 & 150 & 0 & 0 \\ 0 & 0 & 150 & 0 \\ 0 & 0 & 0 & 150 \end{bmatrix} \quad K_P = \begin{bmatrix} 50 & 0 & 0 & 0 \\ 0 & 50 & 0 & 0 \\ 0 & 0 & 50 & 0 \\ 0 & 0 & 0 & 50 \end{bmatrix} \quad \rho = 10 \quad \varepsilon = 0.005$$

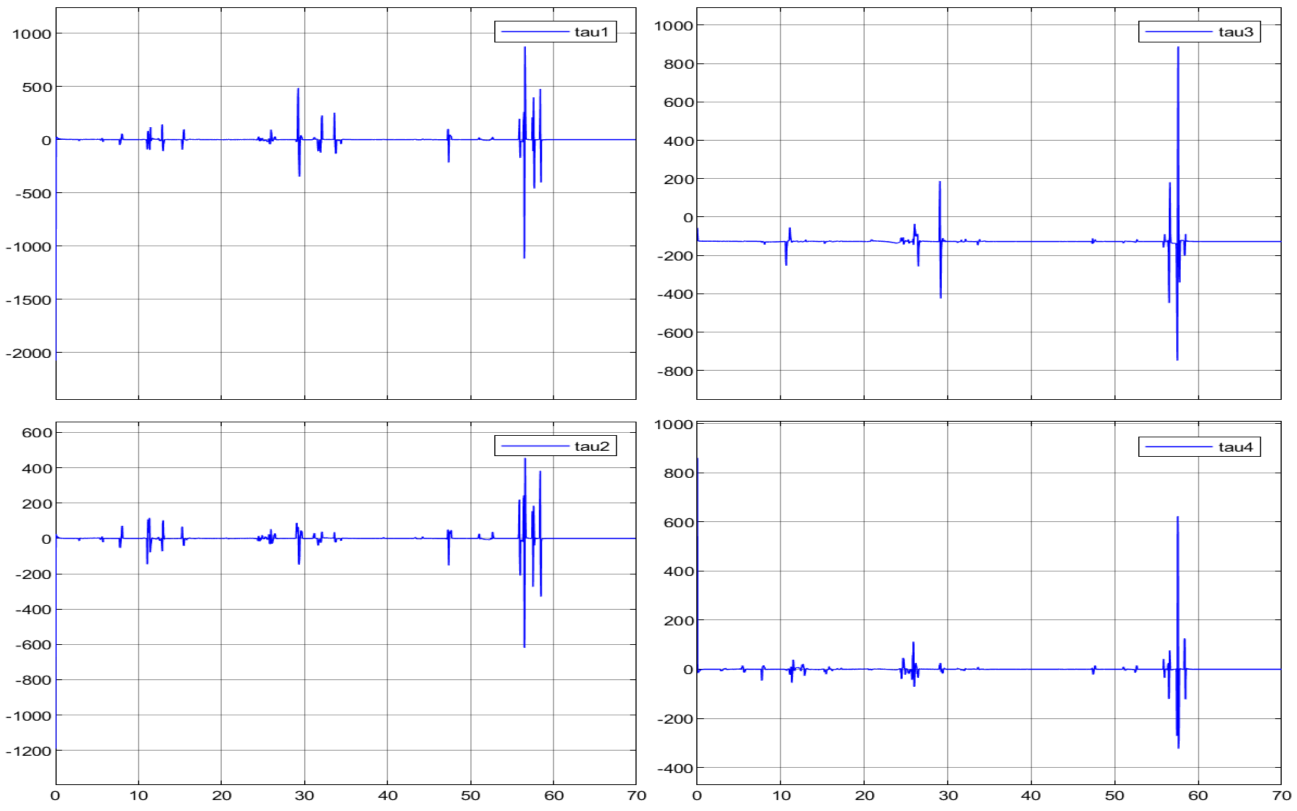
The figure shows the joint acceleration:

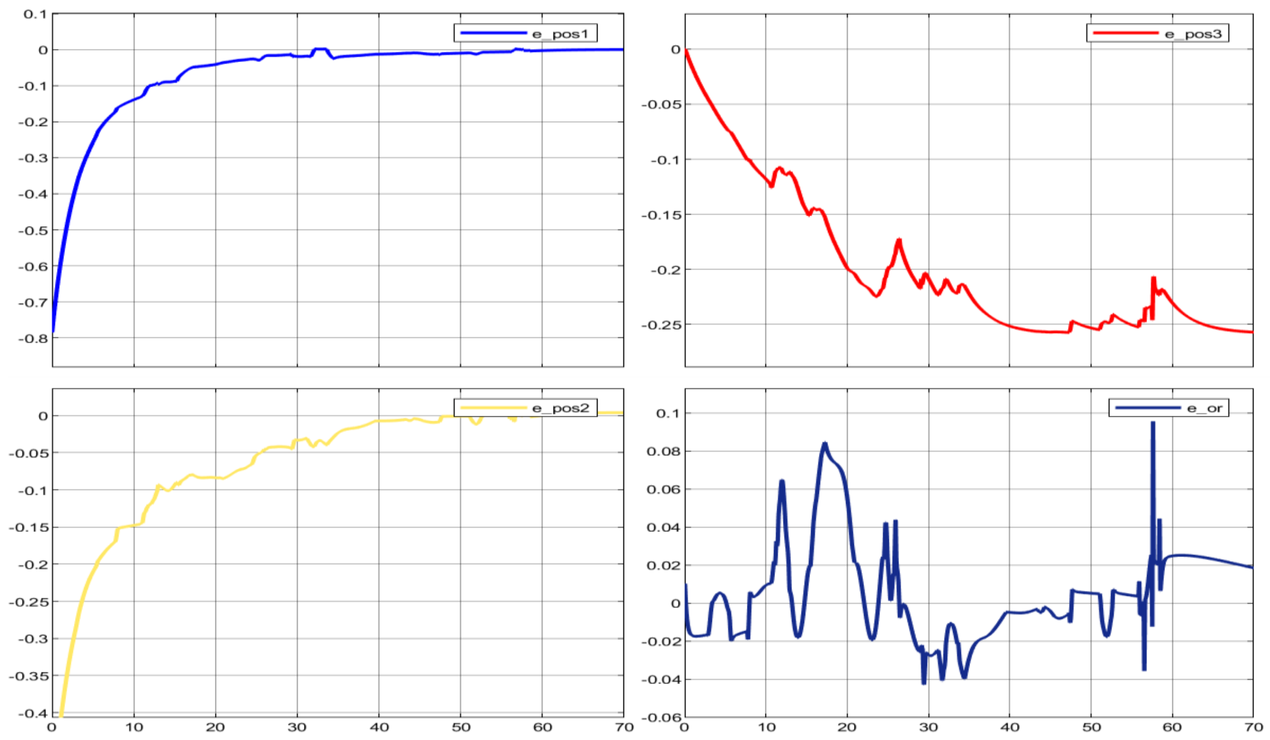


The figure shows the position and orientation error:

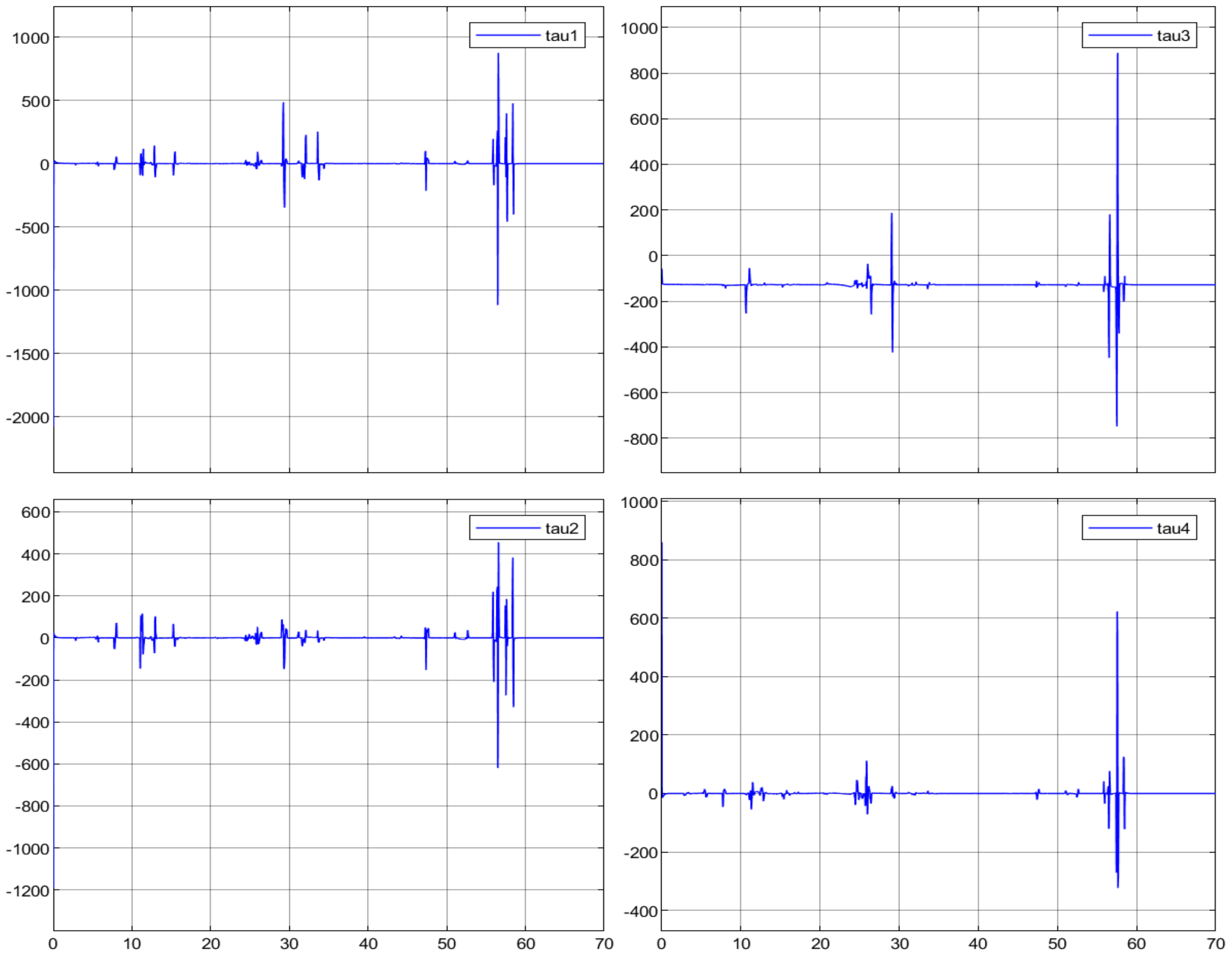


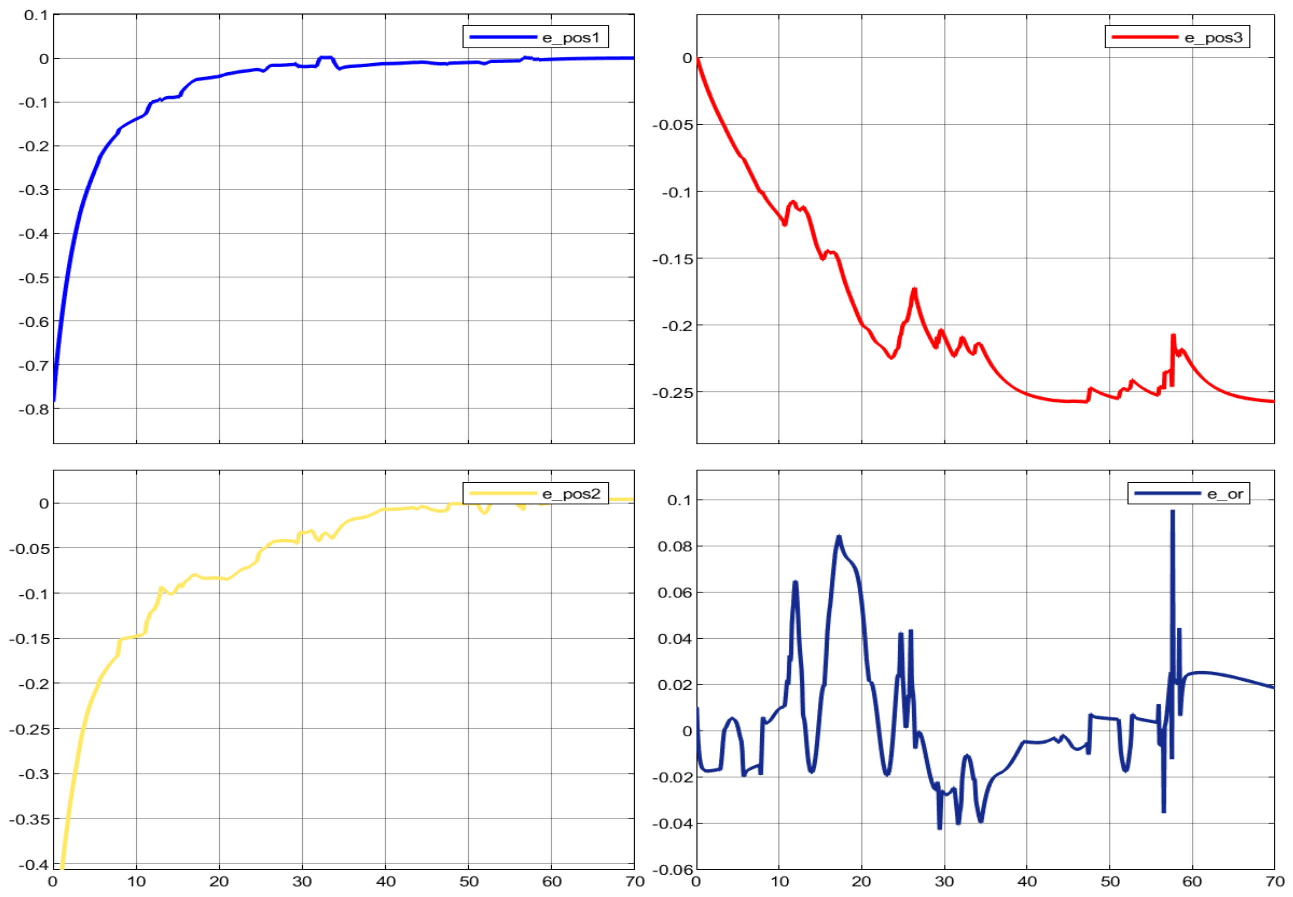
Now we set the ε parameter equals to 0.001, these are the corresponding graphs:





For the last simulation we choose $\varepsilon = 0.01$:





6.3-Adaptive control

This type of control can be applied when the computational model employed for computing inverse dynamics typically has the same structure as that of the true manipulator dynamic model, but parameter estimate uncertainty does exist. The possibility of finding adaptive control laws is ensured by the property of linearity in the parameters of dynamic model of a manipulator:

$$\tau = Y(q,\dot{q},\ddot{q})\pi = B(q)\ddot{q} + C(q,\dot{q})\dot{q} + F_v\dot{q} + g(q)$$

where:

- π is a (px1) vector of constant dynamic parameters;
- $Y(q,\dot{q},\ddot{q})$ is an (nxp) matrix which is function of joint positions, velocities and accelerations;

the choice:

$$\dot{q}_r = \dot{q}_d + \Lambda\tilde{q}$$

$$\sigma = \dot{q}_r - \dot{q}$$

With Λ a positive definite (usually diagonal) (4x4) matrix, allows to use the control law:

$$u = Y(q,\dot{q},\dot{q}_r,\ddot{q}_r)\hat{\pi} + K_D\sigma = \hat{B}(q)\ddot{q}_r + \hat{C}(q,\dot{q})\dot{q}_r + \hat{F}_v\dot{q}_r + \hat{g}(q) + K_D\sigma$$

With a parameter adaptive law:

$$\dot{\hat{\pi}} = K_\pi^{-1}Y^T(q,\dot{q},\dot{q}_r,\ddot{q}_r)\sigma$$

It can be demonstrate that under this control law, the trajectories of the manipulator described by the model shown above, globally asymptotically converge to $\sigma = 0$, $\tilde{q} = 0$ which implies convergence to zero of $\tilde{q}$ and $\dot{\tilde{q}}$ and boundedness of $\hat{\pi}$.

For this simulation the following choices were made

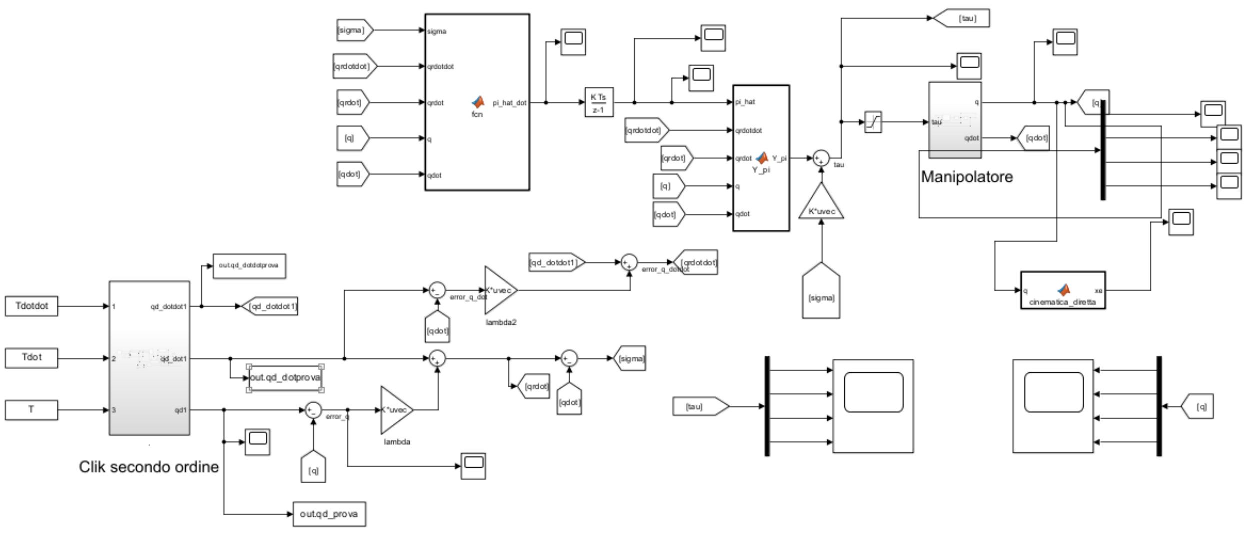
$$\pi = [m_{l1}\ I_{l1}\ I_{m1}\ F_{m1}\ m_{l2}\ I_{l2}\ I_{m2}\ F_{m2}\ \ m_{l3}\ I_{m3}\ \ F_{m3}\ \ I_{l4}\ I_{m4}\ F_{m4}\ ]^T$$

$$\Lambda = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$K_D = 1250$$

K_π= diag([500 500 500 500 500 500 500 500 500 50 500 0.01 500 500 500])

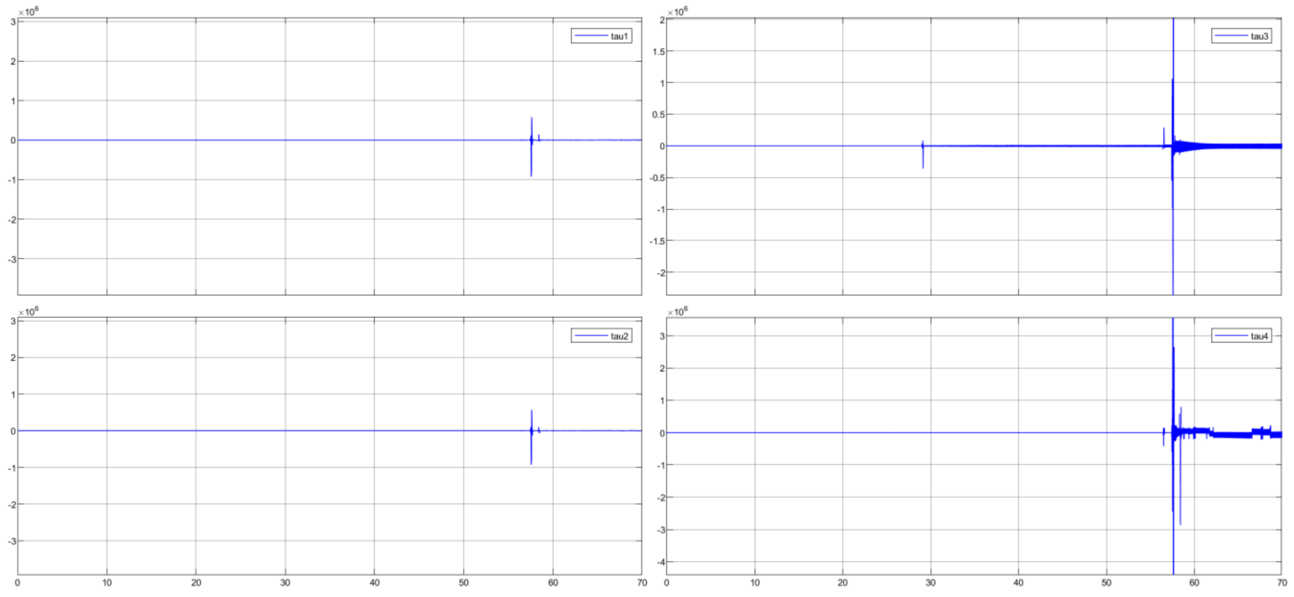
The next figure shows the adaptive control scheme

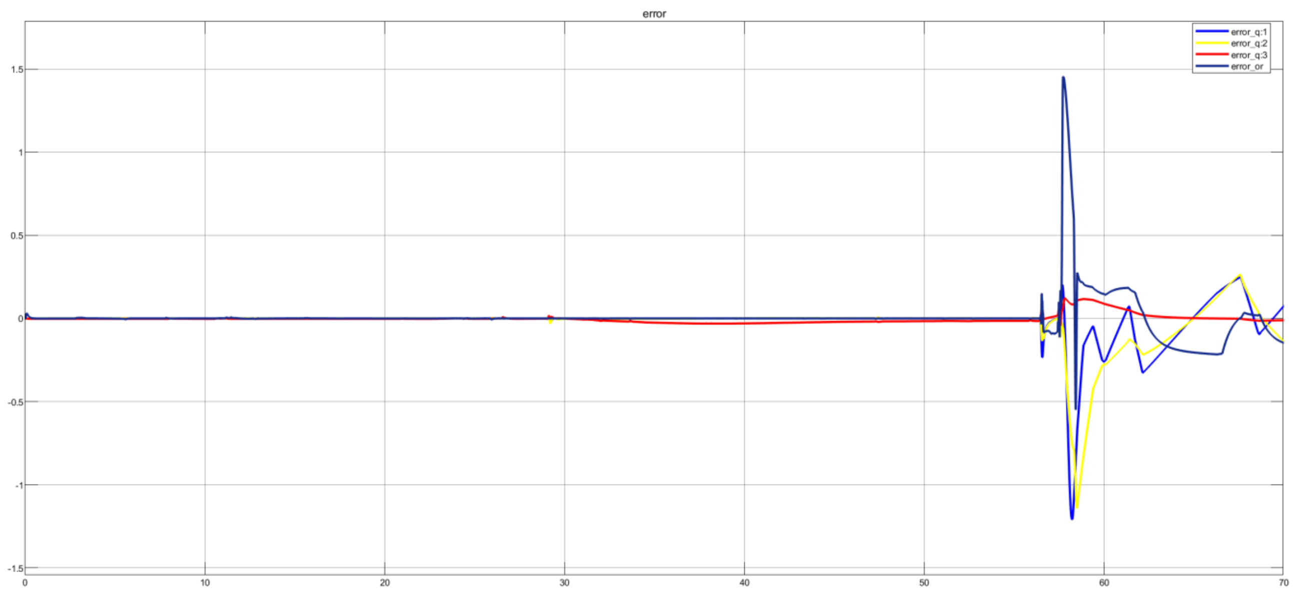


The above control law is formed by 3 different contributions:

1. The term that describes a control action of inverse dynamics type which ensures an approximate compensation of non linear effects and joint decoupling.
2. The term that introduces a stabilizing linear control action of PD type on the tracking error
3. The vector of parameter estimates $\hat{\boldsymbol{\pi}}$ that is updated by an adaptive law of gradient type, so as to ensure asymptotic compensation of the terms in the manipulator dynamic model.

The figure shows the joint acceleration





6.4-Operational space control

They consist in a different approach: control schemes are developed directly in the operational space, so we don't need an inverse kinematics algorithm: references are trajectories and its time derivative.

In inverse dynamics control, model is in the form:

$$B(q)\ddot{q} + n(q,\dot{q}) = u$$

The choice of the inverse dynamics linearizing control:

$$u = B(q)y + n(q,\dot{q})$$

Leads to the system of double integrators:

$$\ddot{q} = y$$

Second-order differential equation is in the form:

$$\ddot{x}_e = J_A(q)\ddot{q} + \dot{J}_A(q,\dot{q})\dot{q}$$

It suggests the choice of the control law:

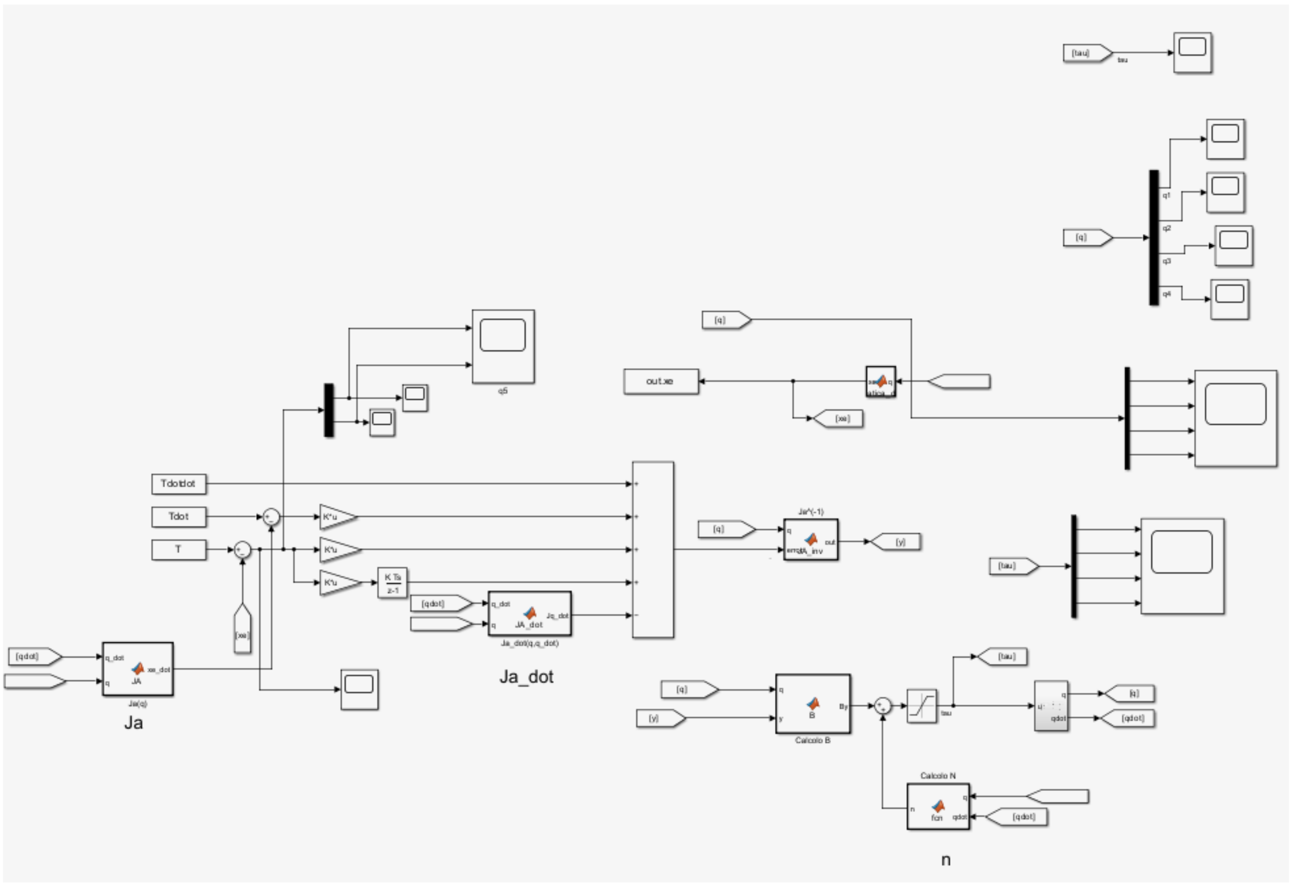
$$\ddot{q} = J_A(q)^{-1}\left(\ddot{x}_d + K_D\dot{\tilde{x}} + K_P\tilde{x} - \dot{J}_A(q,\dot{q})\dot{q}\right)$$

With K_D e K_P positive definite (diagonal) matrices, the system representing theoperational space error dynamics is:

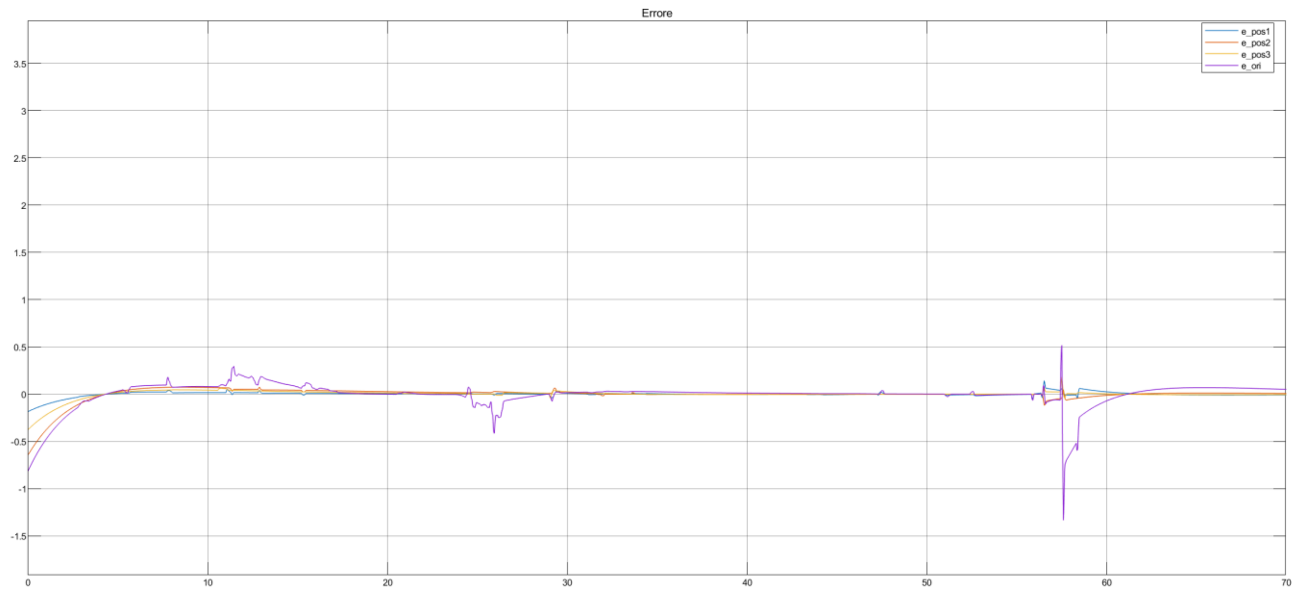
$$\ddot{\tilde{x}} + K_D\dot{\tilde{x}} + K_P\tilde{x} = 0$$

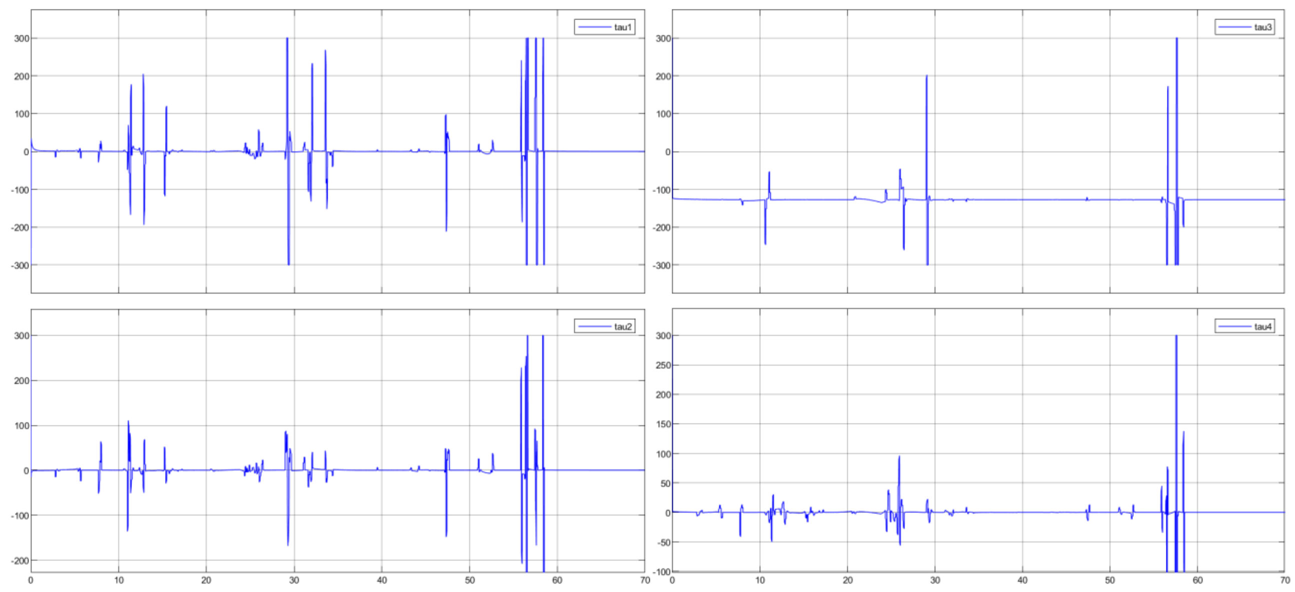
And converges to zero.

The following parameters have been chosen for this simulation



The figure shows the position and orientration error:





CONCLUSIONS

This project start with a manipulability analysis, in velocity and force. Through the respective ellipsoids, we found out that as links that belongs to the reduced system aline themselves, manipulability measure decrease: in fact configurations where arm are completely stretched or overlying are singularity configurations.

After, we realized a trajectory respecting all the assigned constraints.

Then, CLIK algorithms have been implemented, in order to actuate the kinematics inversion.

First 2 of them have been realized for a full rank Jacobian, and have similar steady state behavior, while as regard for the transient ones, the transpose Jacobian scheme , in spite of its computational simplicity, shows a not so good behavior. After relaxing the x -axis operational space component, we introduced Jacobian pseudo-inverse scheme, that let a bigger manipulability measure, but with a delusional trajectory built.

At the end, 3 additional and different control methods have been proposed. Robust and adaptive controls are scheme developed in joint space: a kinematics inversion operation, through a second order algorithm, has been necessary, in order to generate the reference inputs. On the other hand, the operational space inverse dynamics control, that can’t allow in a simple way singularities and redundancy management, works in a optimal way when manipulator model is known with precision, but lacks in robustness.