Advance Control System

Master degree in Computer Engineering for Robotics and Smart Industry

Valentini Michel VR472456

November 2022

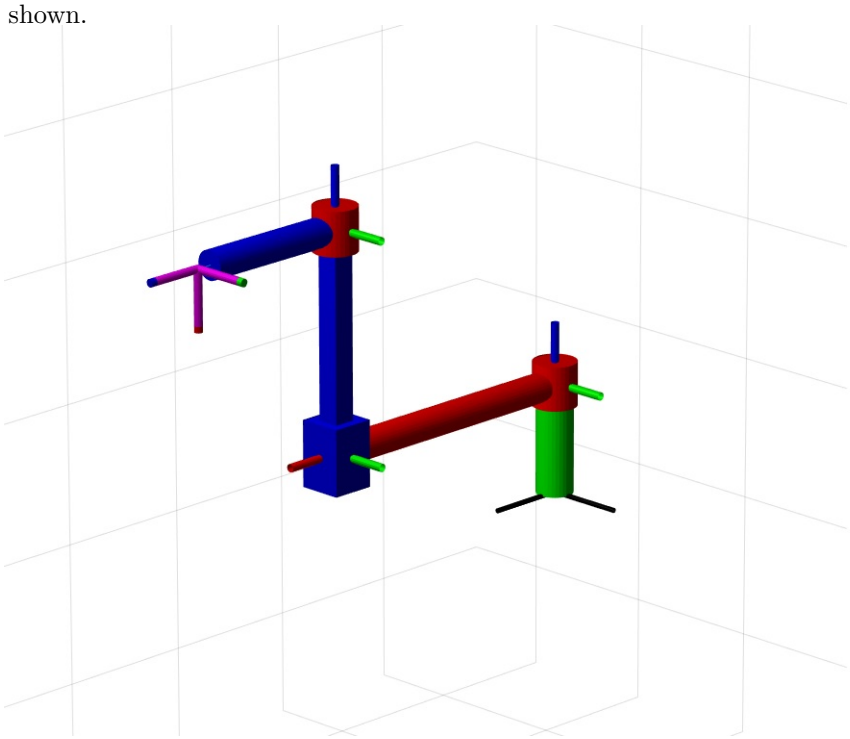
Contents

1 Assignment 1

Description: Compute the DH Table, direct and inverse Kinematics, differential Kinematics of the assigned robot.

1.1 Robot Model

The following image represent the model of my RPR robot where the frame of Denavit Hartenberg convention are shown.



1.2 DH Table

	a	α	d	θ
0	0	0	d_0	0
1	a_1	0	0	θ_1
2	0	0	$q_2 + d_2$	0
3	a_3	0	q_3	0

Table 1: DH Table

1.3 Direct Kinematic

The following rotational matrices are the ones needed describe the change of coordinates from the frames $\Sigma 0$, $\Sigma 1$, $\Sigma 2$, ΣE to the base frame ΣB.

$$T_0^B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_B \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_1^0 = \begin{bmatrix} cos(q1) & -sin(q1) & 0 & a1 \cdot cos(q1) \\ sin(q1) & cos(q1) & 0 & a1 \cdot sin(q1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_2^1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d2 + q2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_3^2 = \begin{bmatrix} cos(q3) & -sin(q3) & 0 & a3 \cdot cos(q3) \\ sin(q3) & cos(q3) & 0 & a3 \cdot sin(q3) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_E^3 = \begin{bmatrix} cos(pi/2) & 0 & -sin(pi/2) & 0 \\ 0 & 1 & 0 & 0 \\ -sin(pi/2) & 0 & cos(pi/2) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Moltiplication matrix :

$$T_1^B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_2^B = \begin{bmatrix} cos(q1) & -sin(q1) & 0 & a1 \cdot cos(q1) \\ sin(q1) & cos(q1) & 0 & a1 \cdot sin(q1) \\ 0 & 0 & 1 & d0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_3^B = \begin{bmatrix} cos(q1) & -sin(q1) & 0 & a1 \cdot cos(q1) \\ sin(q1) & cos(q1) & 0 & a1 \cdot sin(q1) \\ 0 & 0 & 1 & d0 + d2 + q2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_E^B = \begin{bmatrix} cos(q1+q3) \cdot cos(\frac{\pi}{2}) & -sin(q1+q3) & cos(q1+q3) \cdot sin(\frac{\pi}{2}) & a3 \cdot cos(q1+q3) + a1 \cdot cos(q1) \\ sin(q1+q3) \cdot cos(\frac{\pi}{2}) & cos(q1+q3) & sin(q1+q3) \cdot sin(\frac{\pi}{2}) & a3 \cdot sin(q1+q3) + a1 \cdot sin(q1) \\ -sin(\frac{\pi}{2}) & 0 & cos(\frac{\pi}{2}) & d0 + d2 + q2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

1.4 Inverse Kinematics

$$s1 = \frac{(a1 + a3 \cdot c3) \cdot Py - a3 \cdot s3 \cdot Px)}{Px^2 + Py^2}$$

$$c1 = \frac{(a1 + a3 \cdot c3) \cdot Px - a3 \cdot s3 \cdot Py)}{Px^2 + Py^2}$$

$$s3 = \sqrt{(1 - c3^2)}$$

$$c3 = \frac{(Px^2 + Py^2 - ((a1)^2 + (a3)^2))}{2 \cdot a1 \cdot a3}$$

$$q1 = atan2(s1, c1)$$

$$q2 = Pz - (d0 + d2)$$

$$q3 = atan2(\pm s3, c3)$$

1.5 Geometric Jacobian

$$J = \begin{bmatrix} -a1 \cdot sin(q1) - a3 \cdot cos(q1) \cdot sin(q3) - a3 \cdot cos(q3) \cdot sin(q1) & 0 & -a3 \cdot cos(q1) \cdot sin(q3) - a3 \cdot cos(q3) \cdot sin(q1) \\ a1 \cdot cos(q1) + a3 \cdot cos(q1) \cdot cos(q3) - a3 \cdot sin(q1) \cdot sin(q3) & 0 & a3 \cdot cos(q1) \cdot cos(q3) - a3 \cdot sin(q1) \cdot sin(q3) \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

1.6 Analitical Jacobian

$$J = \begin{bmatrix} -a3 \cdot sin(q1+q3) - a1 \cdot sin(q1) & 0 & -a3 \cdot sin(q1+q3) \\ a3 \cdot cos(q1+q3) + a1 \cdot cos(q1) & 0 & a3 \cdot cos(q1+q3) \\ 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

2 Assignment 2

We describe first of all the value of length and mass of links:

m1 = 0.3; m2 = 0.2; m3 = 0.1;

L0 = 0.15; L1 = 0.4; L2 = 0.3; L3 = 0.24;

The follow vectors rappresents the position of the center of mass (COF) for each link:

$$p^1_{l\,1} = \begin{bmatrix} \frac{-L1}{2} \\ 0 \\ 0 \end{bmatrix} \quad p^2_{l\,2} = \begin{bmatrix} 0 \\ 0 \\ \frac{-L2}{2} \end{bmatrix} \quad p^3_{l\,3} = \begin{bmatrix} \frac{-L3}{2} \\ 0 \\ 0 \end{bmatrix}$$

2.1 Kinetic Energy

The kinetic Energy is computed as $\mathcal{T}(q,\dot{q}) = \frac{1}{2}\dot{q}^T B(q)\dot{q}$ with

$$B = \sum_{i=1}^{n} m_{l_i}(J_P^{l_i})^T(J_P^{l_i}) + (J_O^{l_i})^T R_i I^i_{l_i} R_i^T J_O^{l_i}$$

- $I^i_{l_i}$ is the inertia matrix of the link **i**. It is equal to $I^i_{l_i} = I_C + m(r^T r I_{3x3} - rr^T)$, where I_C is the inertia matrix respect to a referance frame Mc with origin on the center of mass. The matrix changes respectively if the link is prismatic or revolute. the i_{th} frame as r=$p^i_{l_i}$.

  $$I_c = \begin{bmatrix} I_c(xx) & 0 & 0 \\ 0 & I_c(yy) & 0 \\ 0 & 0 & I_c(zz) \end{bmatrix}$$

- The robot is compesed by two revolute joint 1-3 and one prismatic joint 2.

$$I^1_{l_1} = \tfrac{1}{12}m_1 \begin{bmatrix} s_1^2 + L_1^2 & 0 & 0 \\ 0 & s_1^2 + L_1^2 & 0 \\ 0 & 0 & 2s_1^2 \end{bmatrix} + m_1 \begin{bmatrix} \frac{L_1^2}{4} & 0 & 0 \\ 0 & \frac{L_1^2}{4} & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$I^2_{l_2} = \tfrac{1}{2}m_2 \begin{bmatrix} r_i^2 + r_o^2 & 0 & 0 \\ 0 & L_2^2 + 3r_i^2 + 3r_o^2 & 0 \\ 0 & 0 & L_2^2 + 3r_i^2 + 3r_o^2 \end{bmatrix} + m_2 \begin{bmatrix} 0 & 0 & 0 \\ 0 & \frac{L_2^2}{4} & 0 \\ 0 & 0 & \frac{L_2^2}{4} \end{bmatrix}$$

$$I^3_{l_3} = \tfrac{1}{12}m_3 \begin{bmatrix} s_3^2 + L_3^2 & 0 & 0 \\ 0 & s_3^2 + L_3^2 & 0 \\ 0 & 0 & 2s_3^2 \end{bmatrix} + m_3 \begin{bmatrix} \frac{L_3^2}{4} & 0 & 0 \\ 0 & \frac{L_3^2}{4} & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

We need to transform the position of the center of masses reference to the base frame and then we can compute the partial Jacobians.

$$p^B_{l_i} = R^B_i p^i_{l_i} + t^B_i$$

$$p^B_{l_1} = R^B_1 p^1_{l_1} + t^B_1 = \begin{bmatrix} cos(q1) & -sin(q1) & 0 \\ sin(q1) & cos(q1) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -\frac{L_1}{2} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} a_1 \cdot cos(q1) \\ a1 \cdot sin(q1) \\ d_0 \end{bmatrix} = \begin{bmatrix} a1 \cdot cos(q1) - \frac{L1 \cdot cos(q1)}{2} \\ a1 \cdot sin(q1) - \frac{L1 \cdot sin(q1)}{2} \\ d0 \end{bmatrix}$$

$$p^B_{l_2} = R^B_2 p^2_{l_2} + t^B_2 = \begin{bmatrix} cos(q1) & -sin(q1) & 0 \\ cos(q1) & cos(q1) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ -\frac{L_2}{2} \end{bmatrix} + \begin{bmatrix} a1 \cdot cos(q1) \\ a1 \cdot sin(q1) \\ d0 + d2 + q2 \end{bmatrix} = \begin{bmatrix} a1 \cdot cos(q1) \\ a1 \cdot sin(q1) \\ d0 - \frac{L_2}{2} + d2 + q2 \end{bmatrix}$$

$$p^B_{l_3} = R^B_3 p^3_{l_3} + t^B_3 = \begin{bmatrix} cos(q1+q3) \cdot cos(piM) & -sin(q1+q3) & cos(q1+q3) \cdot sin(piM) \\ sin(q1+q3) \cdot cos(piM) & cos(q1+q3) & sin(q1+q3) \cdot sin(piM) \\ -sin(piM) & 0 & cos(piM) \end{bmatrix} \begin{bmatrix} -\frac{L_3}{2} \\ 0 \\ 0 \end{bmatrix} +$$

$$\begin{bmatrix} a3 \cdot cos(q1+q3) + a1 \cdot cos(q1) \\ a3 \cdot sin(q1+q3) + a1 \cdot sin(q1) \\ d0 + d2 + q2 \end{bmatrix} = \begin{bmatrix} a3 \cdot cos(q1+q3) + a1 \cdot cos(q1) - \frac{L3 \cdot cos(q1+q3) \cdot cos(piM)}{2} \\ a3 \cdot sin(q1+q3) + a1 \cdot sin(q1) - \frac{L3 \cdot sin(q1+q3) \cdot cos(piM)}{2} \\ d0 + d2 + q2 + \frac{L3 \cdot sin(piM)}{2} \end{bmatrix}$$

Now we can then compute the partial Jacobians. Each column of the partial jacobian is described as follow:

- if joint j is prismatic $\begin{bmatrix} j^{l_i}_{P_j} \\ j^{l_i}_{O_j} \end{bmatrix} = \begin{bmatrix} z_{j-1} \\ 0 \end{bmatrix}$

- if joint j is revolute $\begin{bmatrix} j_{P_j}^{l_i} \\ j_{O_j}^{l_i} \end{bmatrix} = \begin{bmatrix} z_{j-1} \times (p_{l_i} - p_{j-1}) \\ z_{j-1} \end{bmatrix}$

Link 1: $\begin{bmatrix} j_P^{l_1} \\ j_O^{l_1} \end{bmatrix} = \left[\begin{array}{c|c|c} \frac{(L1 \cdot sin(q1))}{2} - a1 \cdot sin(q1) & 0 & 0 \\ a1 \cdot cos(q1) - \frac{(L1 \cdot cos(q1))}{2} & 0 & 0 \\ 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{array}\right]$

Link 2: $\begin{bmatrix} j_P^{l_2} \\ j_O^{l_2} \end{bmatrix} = \left[\begin{array}{c|c|c} -a1 \cdot sin(q1) & 0 & 0 \\ a1 \cdot cos(q1) & 0 & 0 \\ 0 & 0 & 0 \\ \hline 0 & 1 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{array}\right]$

Link 3: $\begin{bmatrix} j_P^{l_3} \\ j_O^{l_3} \end{bmatrix} = \left[\begin{array}{c|c|c} \frac{(L3 \cdot sin(q1+q3) \cdot cos(piM))}{2} - a1 \cdot sin(q1) - a3 \cdot sin(q1+q3) & 0 & -\frac{(sin(q1+q3) \cdot (2 \cdot a3 - L3 \cdot cos(piM)))}{2} \\ a3 \cdot cos(q1+q3) + a1 \cdot cos(q1) - \frac{(L3 \cdot cos(q1+q3) \cdot cos(piM))}{2} & 0 & \frac{(cos(q1+q3) \cdot (2 \cdot a3 - L3 \cdot cos(piM)))}{2} \\ 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{array}\right]$

At the end we can compute the overall inertia matrix B. We define
$B_i = m_{l_i}(J_P^{l_i})^T(J_P^{l_i}) + (J_O^{l_i})^T R_i I_{l_i}^i R_i^T J_O^{l_i}$ equal at B $= \sum_{i=1}^n B_i$

- $B_1 = m_{l_1}(J_P^{l_1})^T(J_P^{l_1}) + (J_O^{l_1})^T R_1 I_{l_1}^1 R_1^T J_O^{l_1} =$
$\begin{bmatrix} \frac{(m1 \cdot (2 \cdot L1^2 - 2 \cdot L1 \cdot a1 + 2 \cdot a1^2 + 3 \cdot ra1^2))}{2} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
- $B_2 = m_{l_2}(J_P^{l_2})^T(J_P^{l_2}) + (J_O^{l_2})^T R_2 I_{l_2}^2 R_2^T J_O^{l_2} =$
$\begin{bmatrix} \frac{(m2 \cdot (L2^2 + 12 \cdot a1^2 + s2^2))}{12} & 0 & 0 \\ 0 & m2 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
- $B_3 = m_{l_3}(J_P^{l_3})^T(J_P^{l_3}) + (J_O^{l_3})^T R_3 I_{l_3}^1 R_3^T J_O^{l_3} =$ Too big to plot here.
- B = also is too big to visualise

With this we finally can compute $\mathcal{T}(q, \dot{q}) = \frac{1}{2}\dot{q}^T B(q)\dot{q}$ which result in a scalar value.
$\mathcal{T}(q, \dot{q})$ it is too big to be represented here. but is possible to visualize on matlab.

2.2 Potential energy

The potential energy depends on the gravity term and it is computed as

$\mathcal{U} = \sum_{i=1}^n \mathcal{U}_i = -\sum_{i=1}^n m_{l_i} g_0^T p_{l_i}$ with $g_0 = \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix}$ and $g = -9.8$

- $\mathcal{U}_1 = -d0 \cdot g \cdot m1$
- $\mathcal{U}_2 = -g \cdot m2 \cdot (d0 - L2/2 + d2 + q2)$
- $\mathcal{U}_3 = -g \cdot m3 \cdot (d0 + d2 + q2 + (L3 \cdot sin(piM))/2)$

Total potential energy is given by:

$$\mathcal{U} = -1.0 \cdot d0 \cdot g \cdot m1 - 1.0 \cdot g \cdot m3 \cdot (d0 + d2 + q2 + 0.5 \cdot L3 \cdot sin(piM)) - 1.0 \cdot g \cdot m2 \cdot (d0 - 0.5 \cdot L2 + d2 + q2)$$

As expected only the second joint affects the potential energy as it is the only one that changes along the z direction.

3 Assignment 3

3.1 Dynamic Model

After computing kinetic and potential energy we can use the Lagrangian approach to compute the dynamic model:

$$\mathcal{L}(q, \dot{q}) = \mathcal{T}(q, \dot{q}) - \mathcal{U}(q)$$

- The torque is given by $\tau = \frac{d}{dt}(\frac{\partial \mathcal{L}}{\partial \dot{q}})^T - (\frac{\partial \mathcal{L}}{\partial q})^T$
- The dynamic model can be described as

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

Where we're not taking into account frictions, and B matrix has already been computed for the kinetic energy.

Remains to calculate $C(q, \dot{q})$.

$$\sum_{j=1}^{n} c_{ij}(q)\dot{q}_j = \sum_{j=1}^{n}\sum_{k=1}^{n} \frac{1}{2}(\frac{\partial b_{ij}}{\partial q_k} + \frac{\partial b_{ik}}{\partial q_j}) - \frac{\partial b_{jk}}{\partial q_i})\dot{q}_k\dot{q}_j$$

The result of matrix $C(q, \dot{q})$ is too big to plot here, but is possible to visualise in matlab.

$g = \begin{bmatrix} 0 \\ -g \cdot m2 - g \cdot m3 \\ 0 \end{bmatrix}$ which is actually constant for this robot.

4 Assignment 4

4.1 RNE formulation

The Newton-Euler algorithm is a recursive function in the form

$\tau = NE(q, \dot{q}, \ddot{q}, g_0)$

The equivalence with Lagrangian can be checked with the following relations:

$g(q) = NE(q, 0, 0, g_0)$

$C(q, \dot{q})\dot{q} = NE(q, \dot{q}, 0, 0)$

$B(q) = \begin{bmatrix} B_1(q) & ... & B_n(q) \end{bmatrix}$ with $B_i(q) = NE(q, 0, e_i, 0)$

Just to see how it differs we can compare the values of the matrices in the configuration

$q = \begin{bmatrix} \frac{\pi}{3} \\ -0.05 \\ -\frac{\pi}{3} \end{bmatrix}$, $\dot{q} = \begin{bmatrix} 0.1 \\ 0.1 \\ 0.1 \end{bmatrix}$

- $C_{lag} = \begin{bmatrix} 0.0002494 \\ 0 \\ 6.235e-6 \end{bmatrix}$, $C_{rne} = \begin{bmatrix} 0.0002356 \\ 0 \\ 4.157e-5 \end{bmatrix}$
- $G_{lag} = \begin{bmatrix} 0 \\ 2.943 \\ 0 \end{bmatrix}$, $G_{rne} = \begin{bmatrix} 0 \\ 2.943 \\ 0 \end{bmatrix}$
- $B_{lag} = \begin{bmatrix} 0.1156 & 0 & 0.01166 \\ 0 & 0.3 & 0 \\ 0.01166 & 0 & 0.00686 \end{bmatrix}$, $B_{RNE} = \begin{bmatrix} 0.1007 & 0 & 0.0087 \\ 0 & 0.3 & 0 \\ 0.0015 & 0 & 0.0087 \end{bmatrix}$
- $TAU_{lag} = \begin{bmatrix} 0.1156 \cdot ddq1 + 0.01166 \cdot ddq3 + 0.0002494 \\ 0.3 \cdot ddq2 + 2.943 \\ 0.01166 \cdot ddq1 + 0.00686 \cdot ddq3 + 6.235e-6 \end{bmatrix}$,

  $TAU_{RNE} = \begin{bmatrix} 0.1007 \cdot ddq1 + 0.0087 \cdot ddq3 + 0.0002356 \\ 0.3 \cdot ddq2 + 2.943 \\ 0.0015 \cdot ddq1 + 0.0087 \cdot ddq3 + 4.157e-5 \end{bmatrix}$

5 Assignment 5

5.1 Dynamic model in operational space

To compute the dynamic model in operational space we use the following relationship for non-redundant manipulator:

$B_A(x) = J_A^{-T} B J_A^{-1}$
$C_A(\dot{x}) = J_A^{-T} C\dot{q} - B_A \dot{J}_A \dot{q}$
$G_A(x) = J_A^{-T} G$
where

$$\dot{J}_A = \begin{bmatrix} -a3 \cdot cos(q1+q3) - a1 \cdot cos(q1) & 0 & -a3 \cdot cos(q1+q3) \\ -a3 \cdot sin(q1+q3) - a1 \cdot sin(q1) & 0 & -a3 \cdot sin(q1+q3) \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

The matrix B_A, C_A, G_A are too big to be plot here, but is possible to visualise in Matlab.

6 Assignment 6

Design the Joint Space PD control law with gravity compensation.
The control law is reported in the following equation:

$$\tau = g(q) + K_p(q_d q) K_d \dot{q}$$

The following simulations are using as parameters: $K_D = \begin{bmatrix} 5 \\ 5 \\ 5 \end{bmatrix}$ $K_P = \begin{bmatrix} 50 \\ 50 \\ 50 \end{bmatrix}$ with a desired position $q_d = \begin{bmatrix} pi \\ -0.2 \\ -pi \end{bmatrix}$

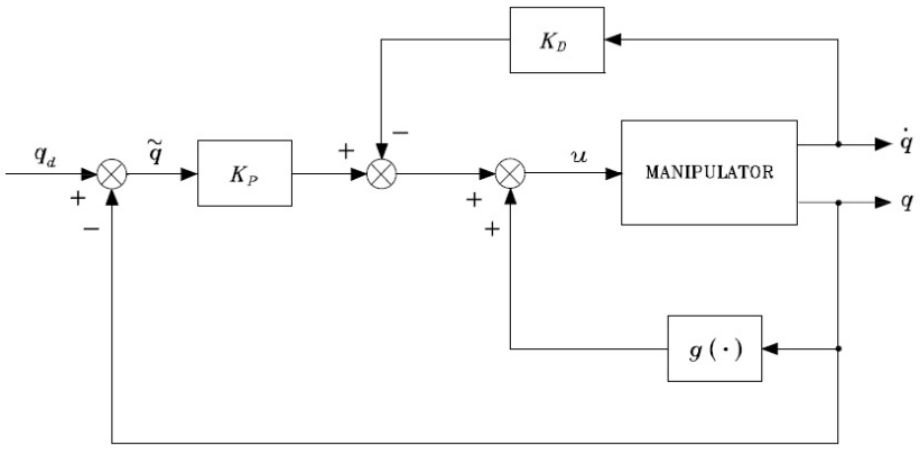


Figure: Joint Space PD control with gravity compensation block scheme

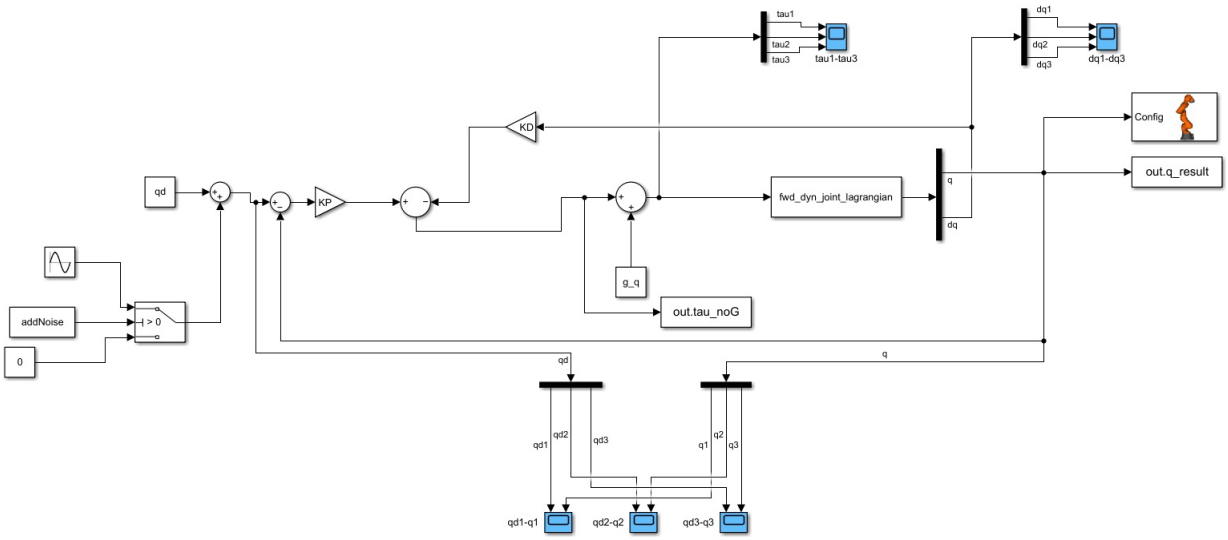


Figure 1: Implementation of the Joint Space PD control law with gravity compensation

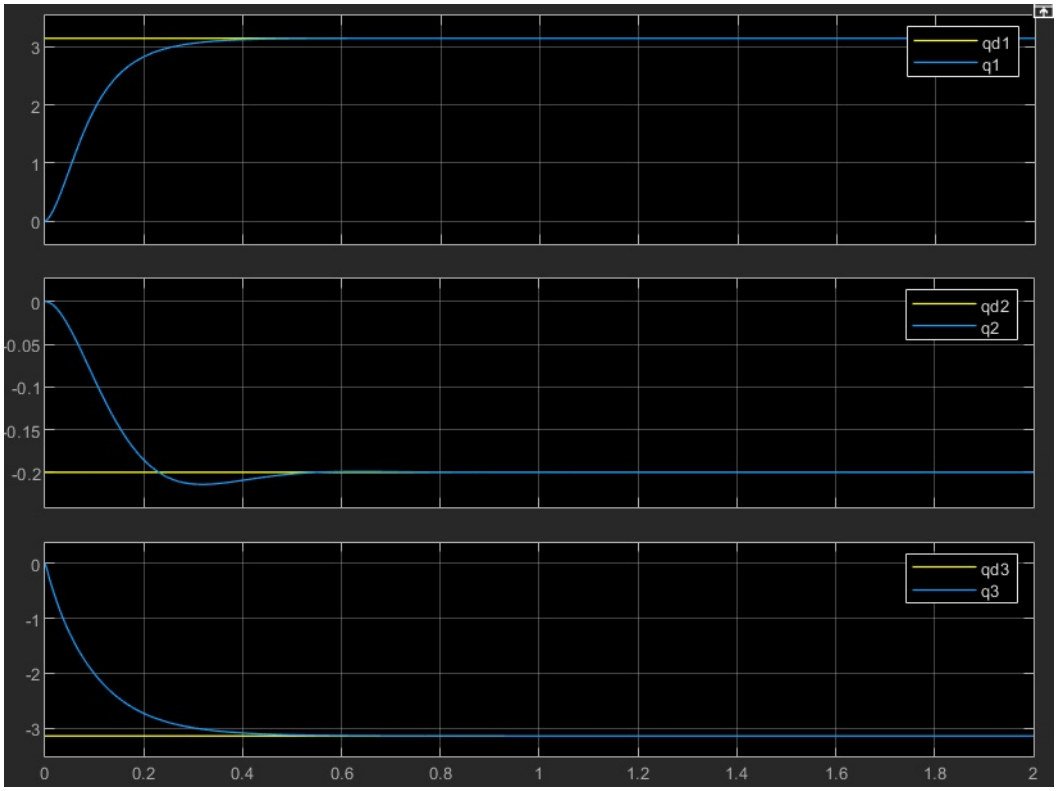


Figure 2: Position Scope : Joint Space in gravity compensation

6.1 What happens if g(q) is not taken into account?

As we can see, the joint 2 which is the only affected by gravity never reaches the desired position.

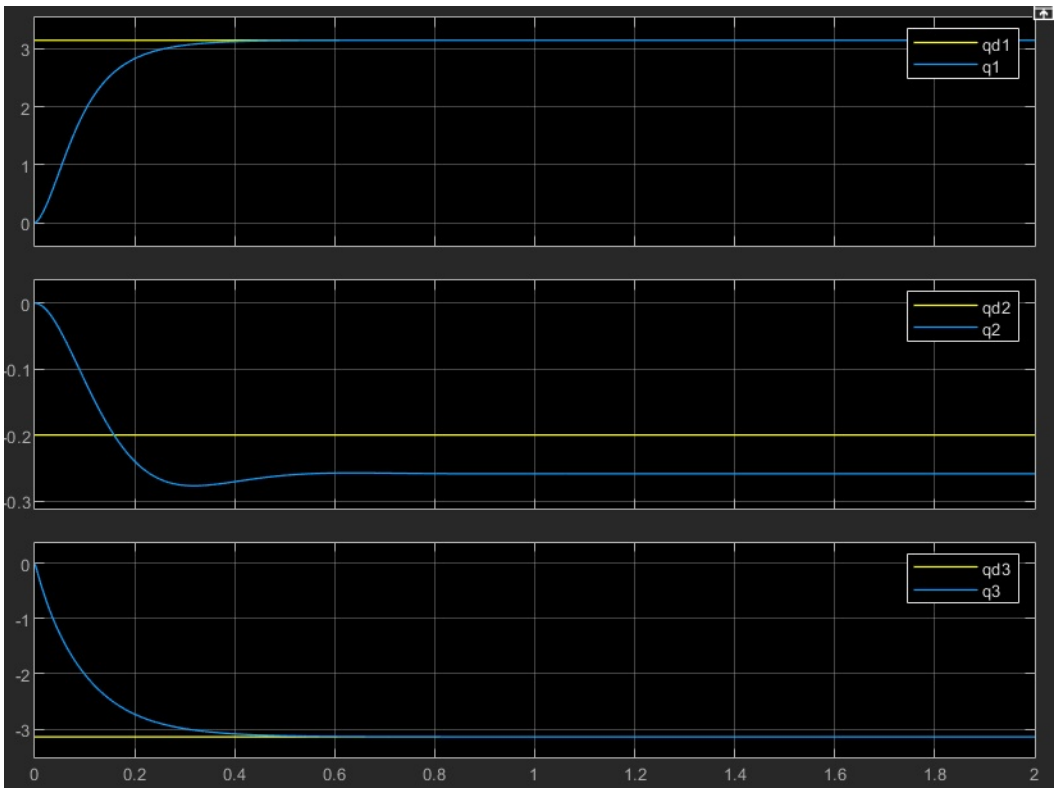


Figure 3: Position Scope : Joint Space in gravity compensation without gravity

6.2 What happens if the gravity term is set constant and equal to $g(q_d)$ within the control law?

As in this case the gravity term is already constant, so a test we can do is set the gravity term with a wrong value

$$g(q_d) = \begin{bmatrix} 0 \\ (-g \cdot m2 - g \cdot m3) \cdot 2 \\ 0 \end{bmatrix}$$

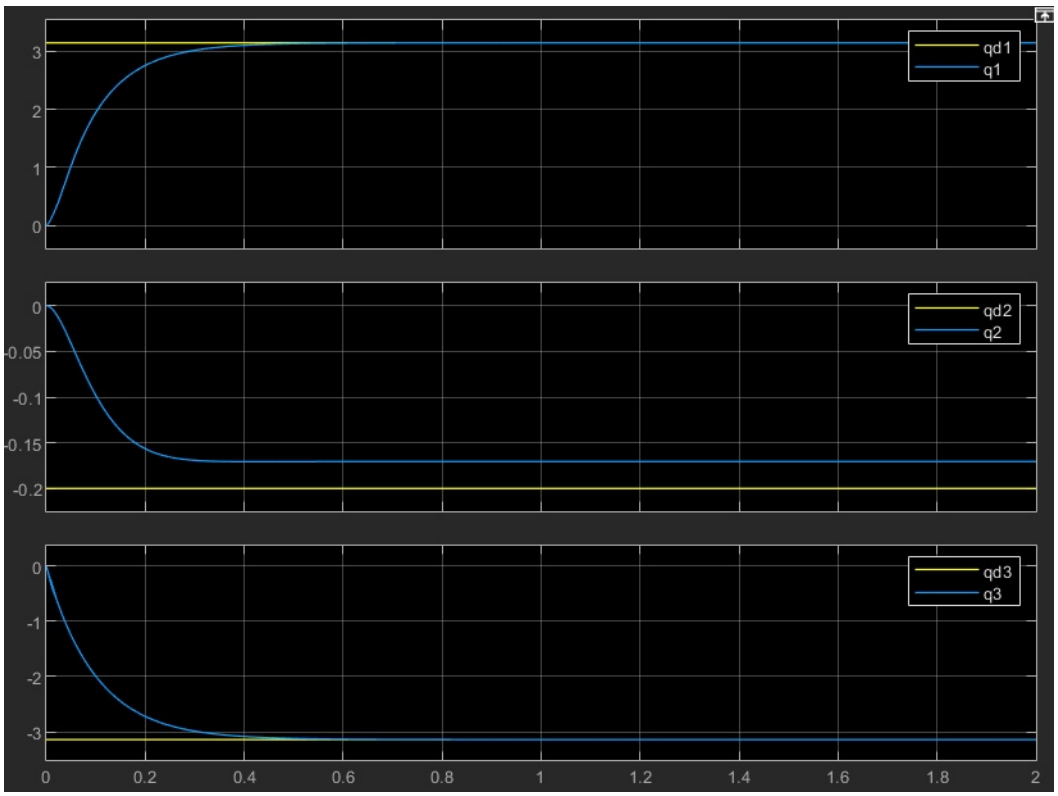


Figure 4: Position Scope : Joint Space in gravity compensation with double G

6.3 What happens if q_d is not constant (e.g. $q_d(t) = \overline{q}_d + \Delta sin(\omega t)$)?

The sine wave has an amplitude of 0.1 with a frequency of 5 rad/s and the result can be seen in where we can notice that the controller is not able to follow the trajectory perfectly.

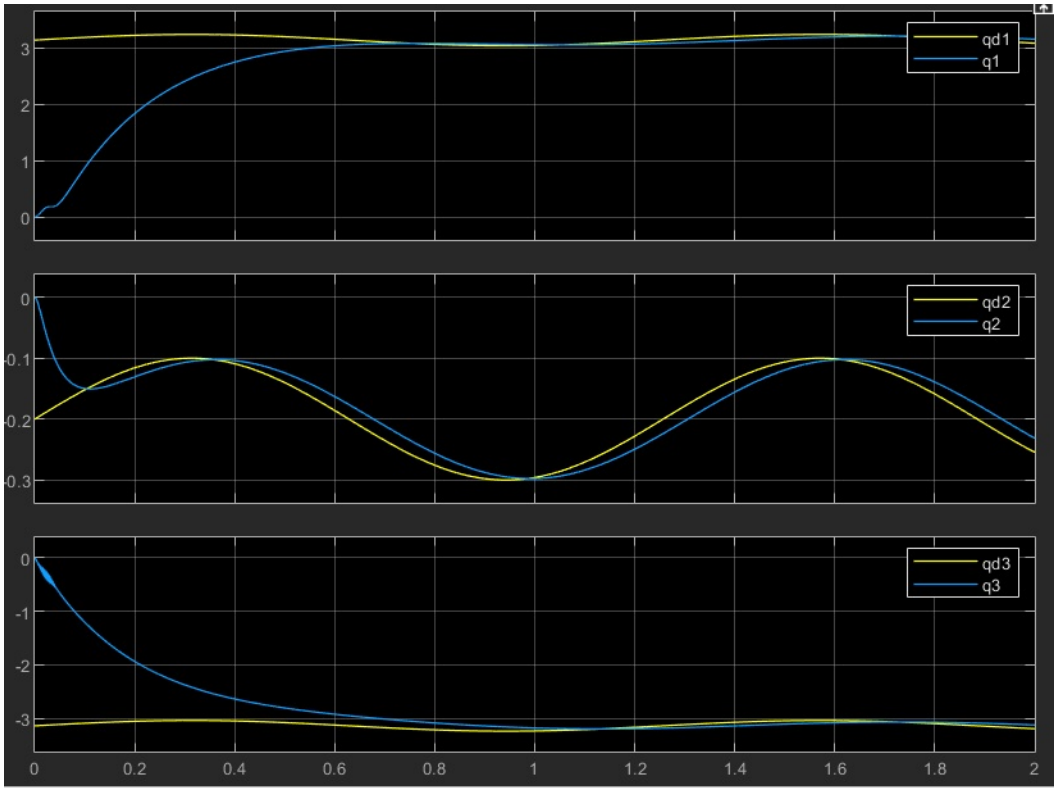


Figure 5: Position Scope : Joint Space in gravity compensation with sine

7 Assignment 7

7.1 Design the Joint Space Inverse Dynamics Control law

The trajectory is given by a "cubicpolytraj" function by matlab. Witch generates a third-order polynomial that achieves a given set of input waypoints with corresponding time points

$$q_i = \begin{bmatrix}0\\0\\0\end{bmatrix} \text{ to } q_1 = \begin{bmatrix}\pi\\-0.2\\\pi\end{bmatrix} \text{ to } q_f = \begin{bmatrix}0\\-0.1\\\pi/2\end{bmatrix} \text{ and the control parameters are } K_D = \begin{bmatrix}20\\100\\20\end{bmatrix}, K_P = \begin{bmatrix}80\\1000\\100\end{bmatrix}.$$

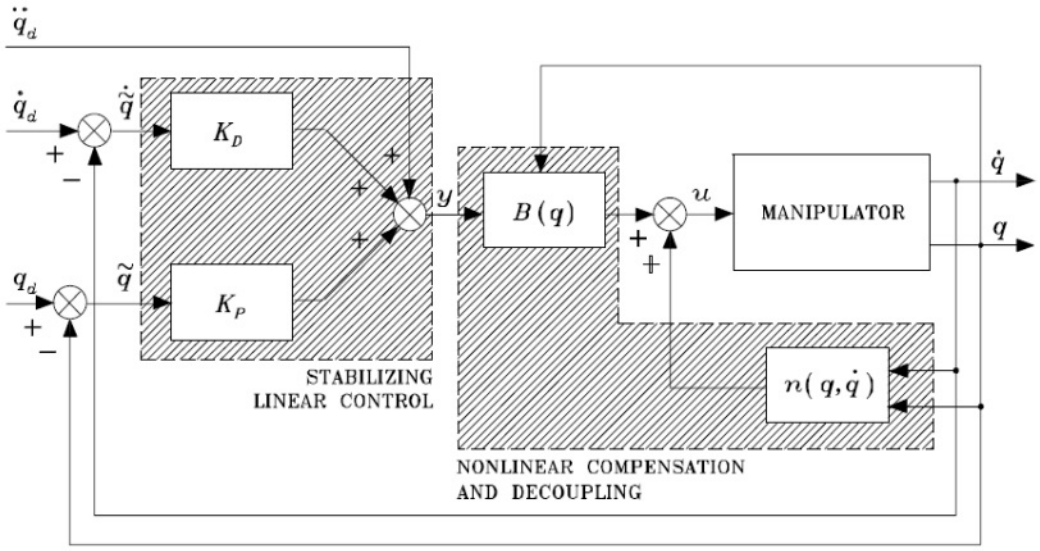


Figure 6: Block scheme : Joint Space Inverse Dynamics

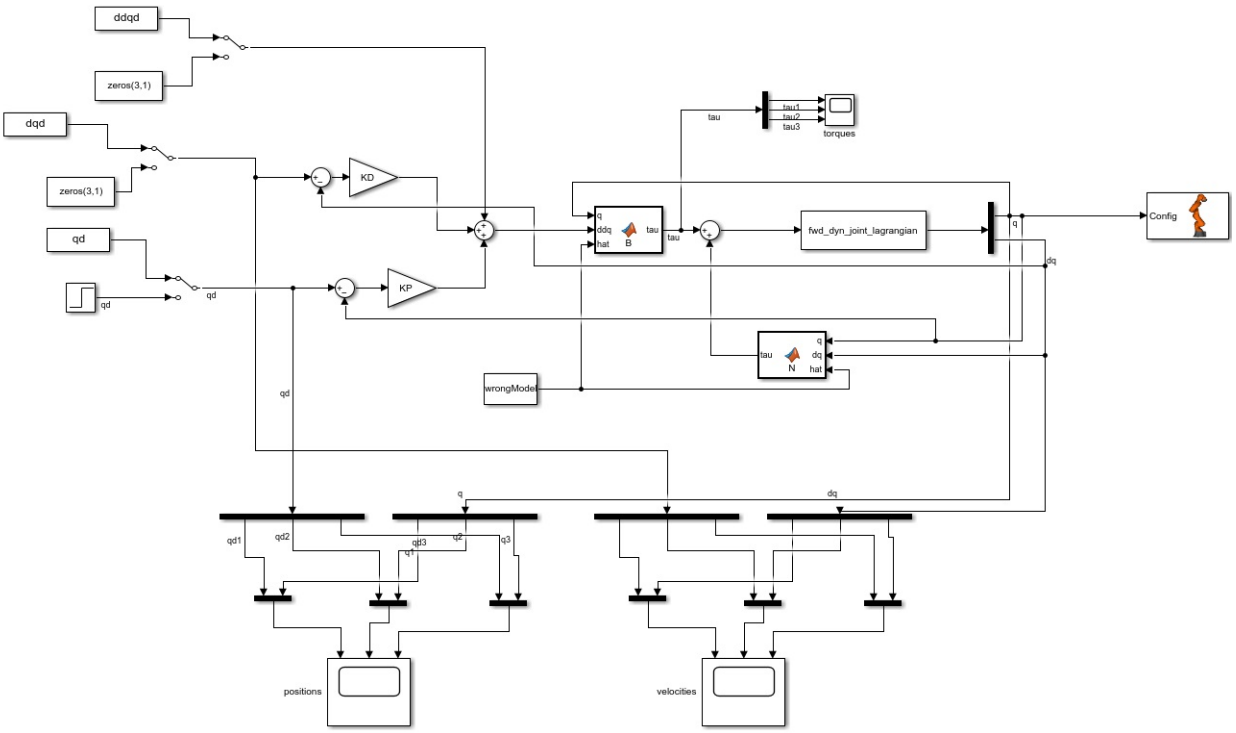


Figure 7: Simulink Model : Joint Space Inverse Dynamics

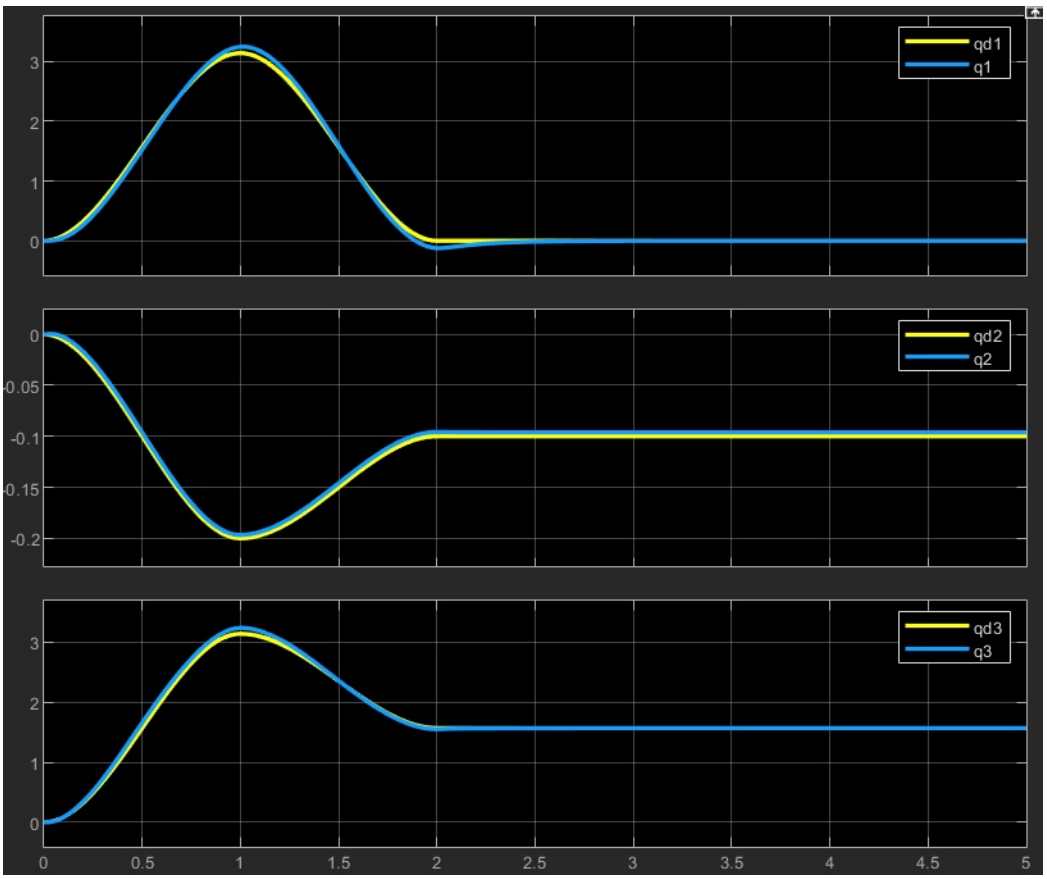


Figure 8: Position Scope : Joint Space Inverse Dynamics

7.2 Check that in the nominal case the dynamic behaviour is equivalent to the one of a set of stabilized double integrators

For this scenario the following values have been used: $K_D = \begin{bmatrix} 20 \\ 100 \\ 20 \end{bmatrix}$, $K_P = \begin{bmatrix} 80 \\ 1000 \\ 100 \end{bmatrix}$ and the result is almost identical to the Figure 8.

7.3 Check the behavior of the control law when the $\hat{B}, \hat{C}, \hat{g}$ used within the controller are different than the "true ones" B;C; g (e.g. slightly modify the masses, the frictions, ...)

The joint with worst result is the one affected by gravity as we can see in Figure 9.

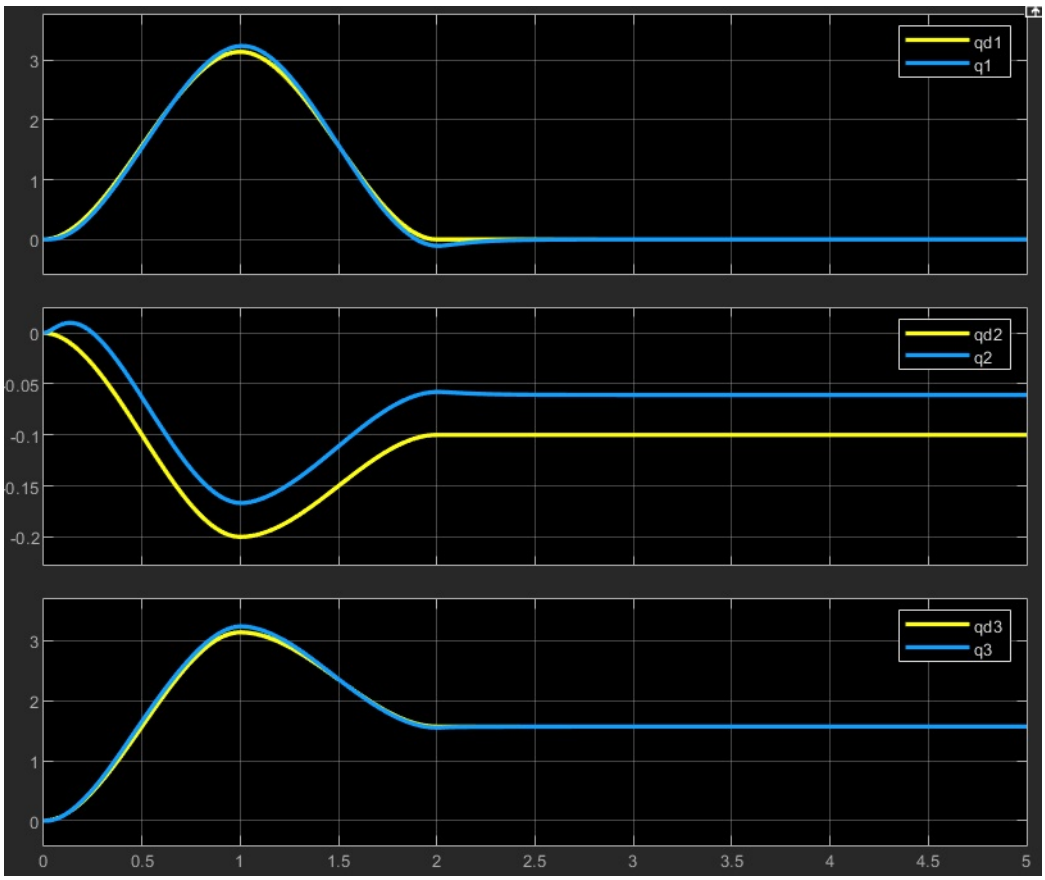


Figure 9: Scope Position : Joint space Inverse Dynamics, simulated with wrong model

7.4 What happens to the torque values when the settling time of the equivalent second order systems is chosen very small?

For having a small settling time high gains are needed, which will result in peaks in torques as we can see in figure 10, while the positions still reach the settling time all at the same moment.

The gains used are: $K_D = \begin{bmatrix} 40 \\ 40 \\ 40 \end{bmatrix}$, $K_P = \begin{bmatrix} 500 \\ 500 \\ 500 \end{bmatrix}$

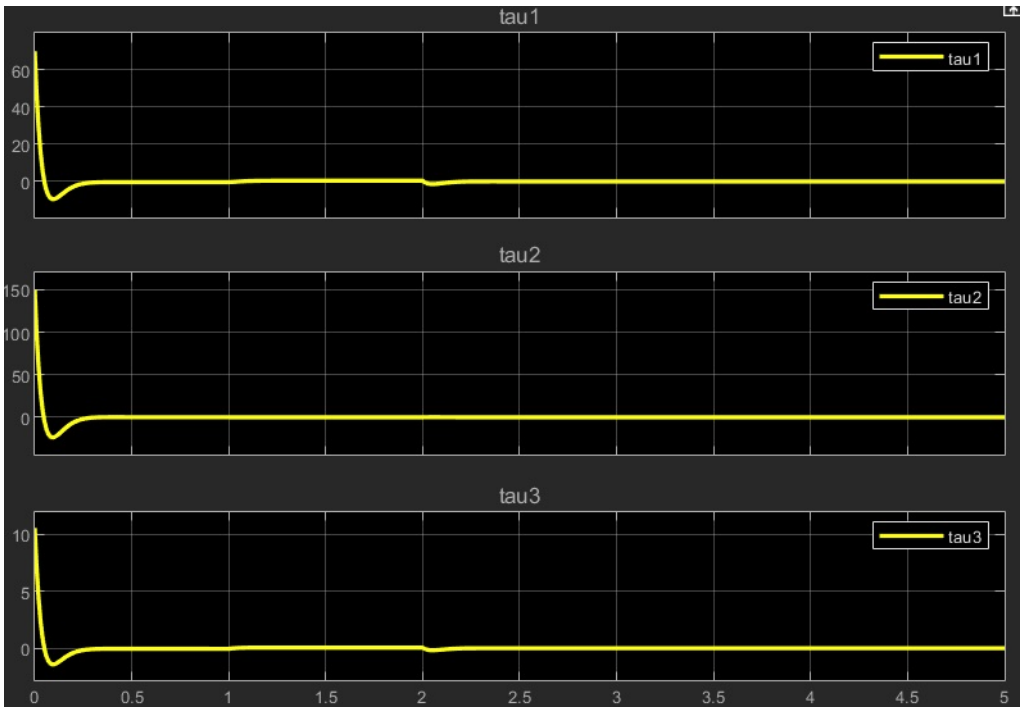


Figure 10: Scope Joint space Inverse Dynamics: Torques with high gains

8 Assignment 8

8.1 Implement in Simulink the Adaptive Control law for the a 1-DoF link under gravity

The schema can be seen in Figure 11.

- Dynamic parameters: $I = 0.3, F = 0.1, G = 2.943$;
- Initial estimate: $\hat{I} = 0.29, \hat{F} = 0.09, \hat{G} = 2.933$;
- Trajectory: $q_d(t) = Asin(\omega t), \ddot{q}_d(t) = square\_wave(\pm A)$
- Control parameters: $K_D = 40, \lambda = 300, K_\theta^- 1 = \begin{bmatrix} 1000 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 100 \end{bmatrix}$

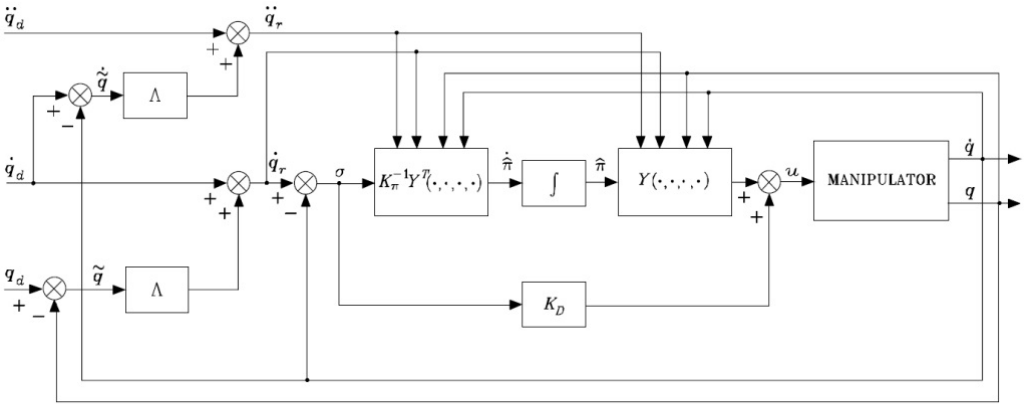


Figure: Joint Space Adaptive Control block scheme

Figure 11:

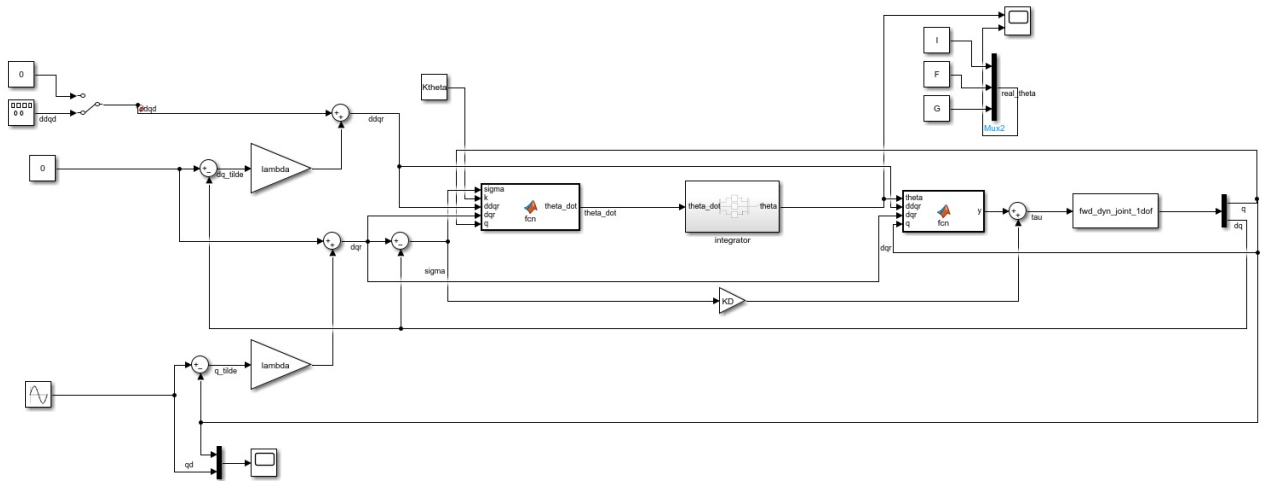


Figure 12: Simulink Adaptive Control law for the a 1-DoF link under gravity

The tracking of the trajectory is well performing as we can see in Figure 13, and by feeding the integrator with the initial estimations as described before we can see in Figure 14 that the values oscillates back and forth around the true value. Moreover the high value in K_θ keeps the value closer, otherwise the amplitude of the oscillation would be much higher. We can also notice that the G paramam is not too well performing as the other, but the different value is very low.

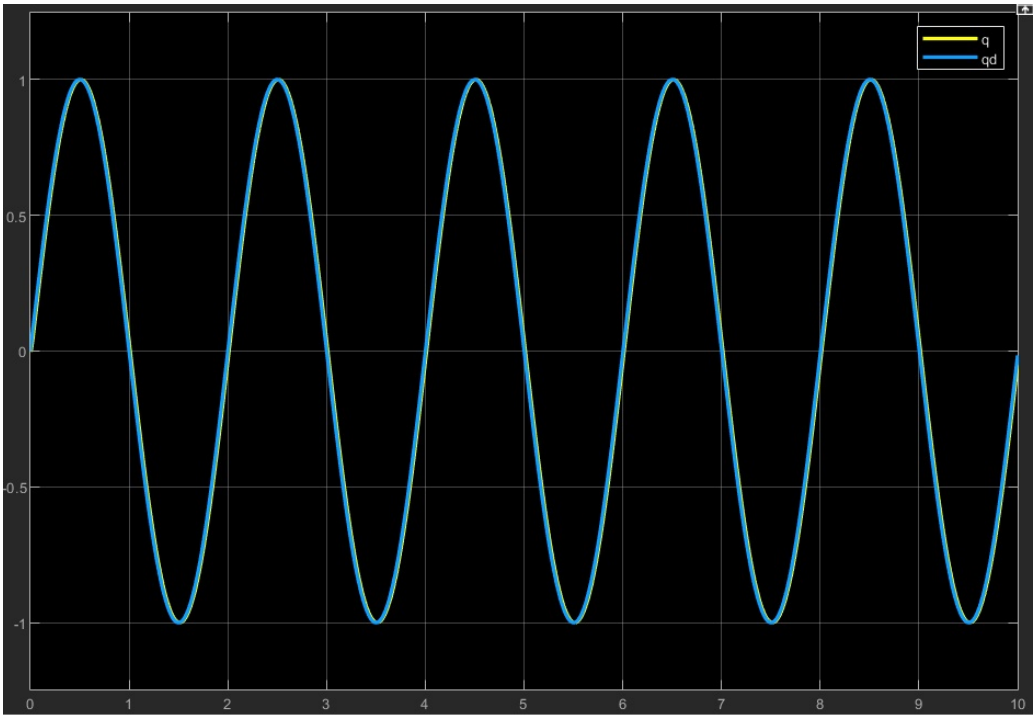


Figure 13: Adaptive control: position

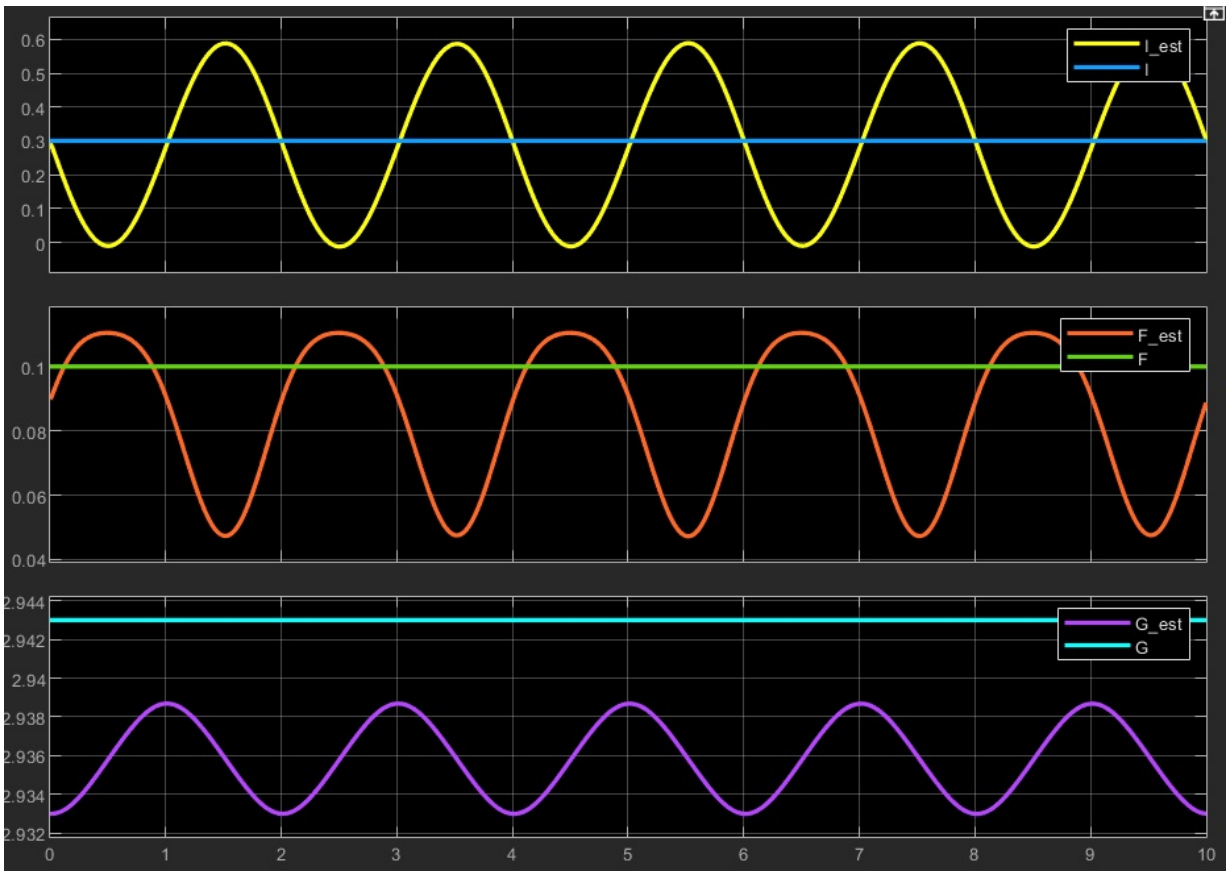


Figure 14: Adaptive control: dynamic parameters evolution

9 Assignment 9

9.1 Design the Operational Space PD control law with gravity compensation

The schema can be seen in figure 15. This time the gains refers to x,y,z directions not on joints and are defined as: $K_D = \begin{bmatrix} 80 \\ 200 \\ 50 \end{bmatrix}$, $K_P = \begin{bmatrix} 1000 \\ 500 \\ 50 \end{bmatrix}$

The gravity compensation case is reported in Figure 16. The other cases resembles the joint space cases: without gravity compensation (or with a wrong g model) the z direction has a steady state error and with a sin wave added to input the tracking is not perfect.

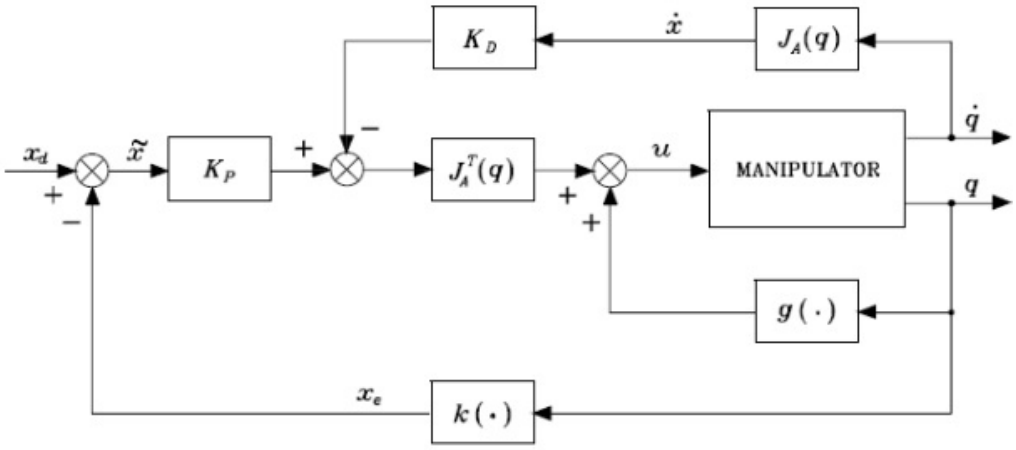


Figure: Operational Space PD control with gravity compensation block scheme.

Figure 15: Operational space PD with gravity compensation

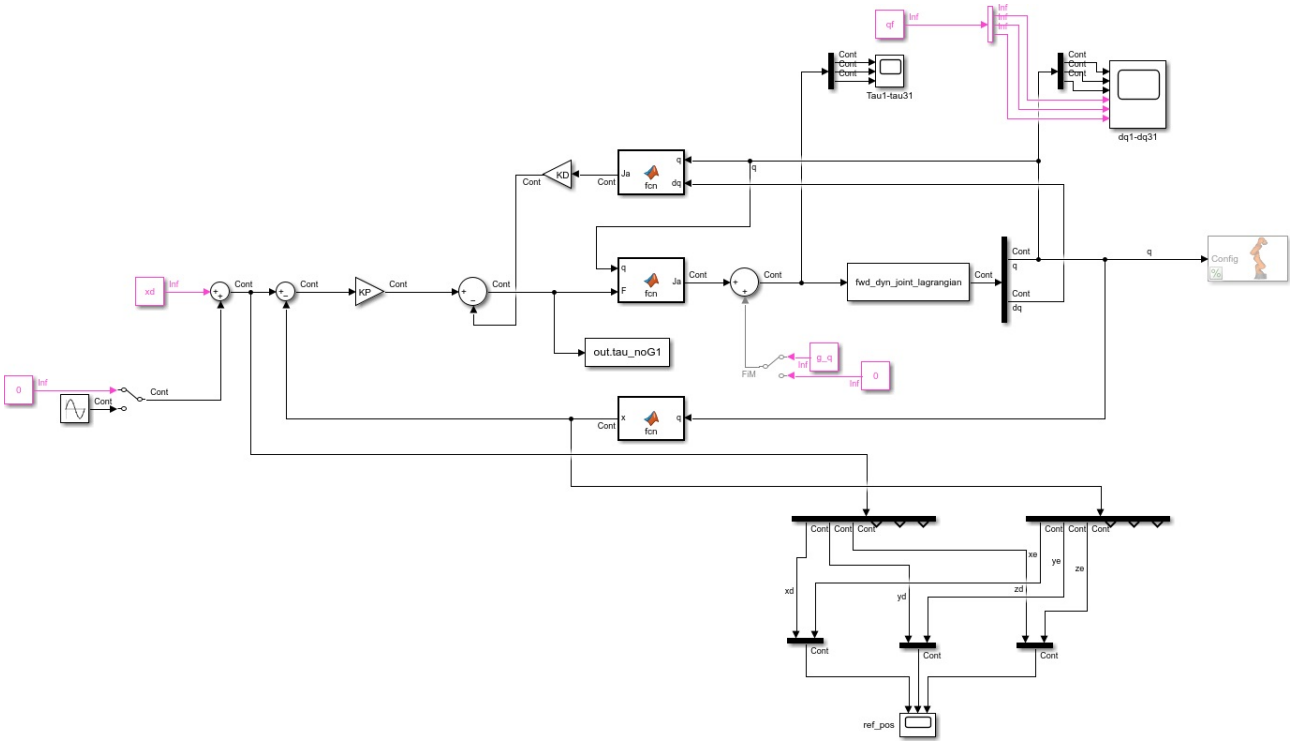


Figure 16: Simulink : Operational space PD with gravity compensation

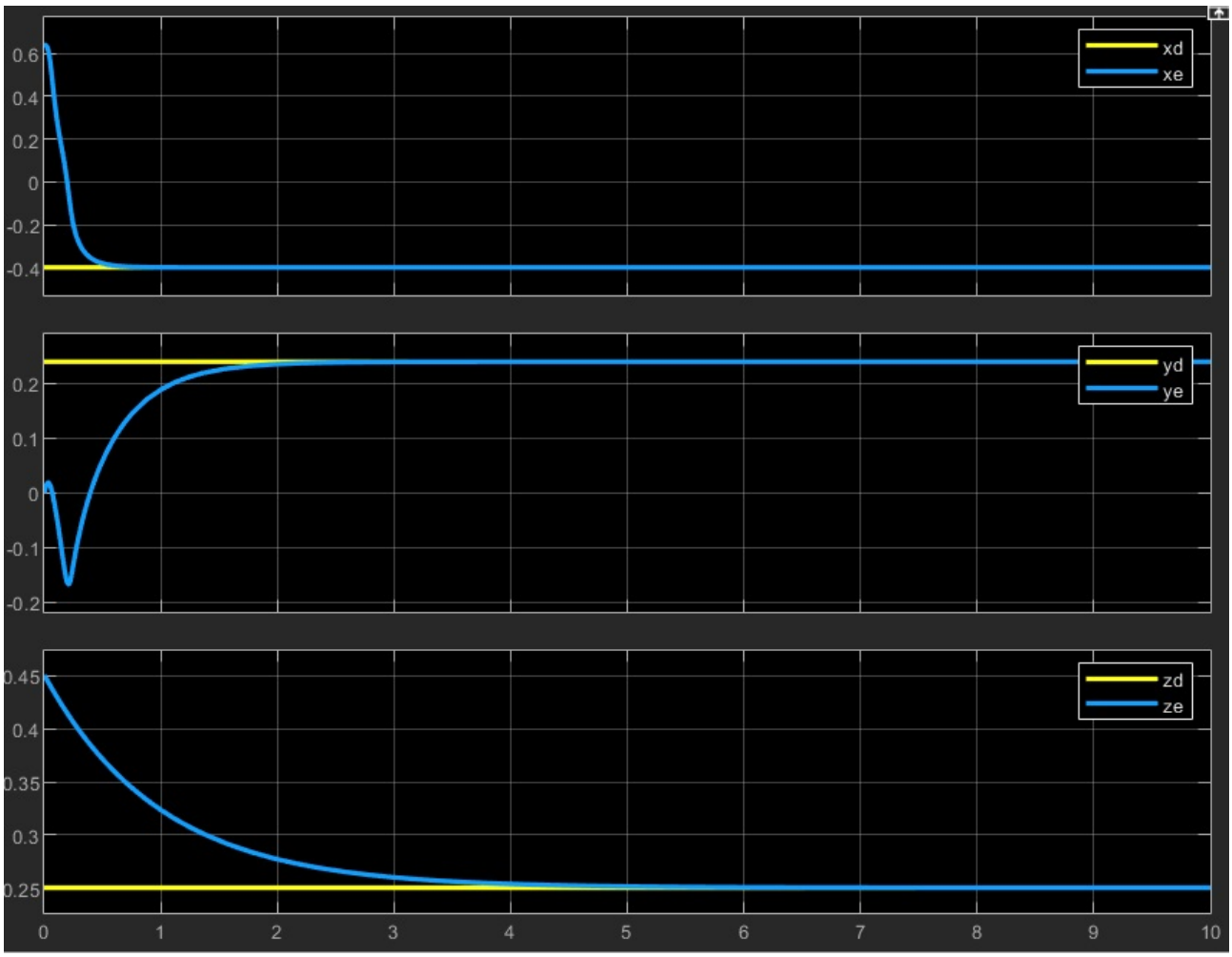


Figure 17: Scope : Operational space PD with gravity compensation

10 Assignment 10

10.1 Design the Operational Space Inverse Dynamics Control law

The schema can be seen in . The trajectory is again given by cubicpolytraj function by matlab. The gains used are: $K_D = \begin{bmatrix} 250 \\ 150 \\ 250 \end{bmatrix}$, $K_P = \begin{bmatrix} 1000 \\ 500 \\ 500 \end{bmatrix}$

It's worth noticing that as the gains are referring to operational space, the joint torques are still low as we can see in

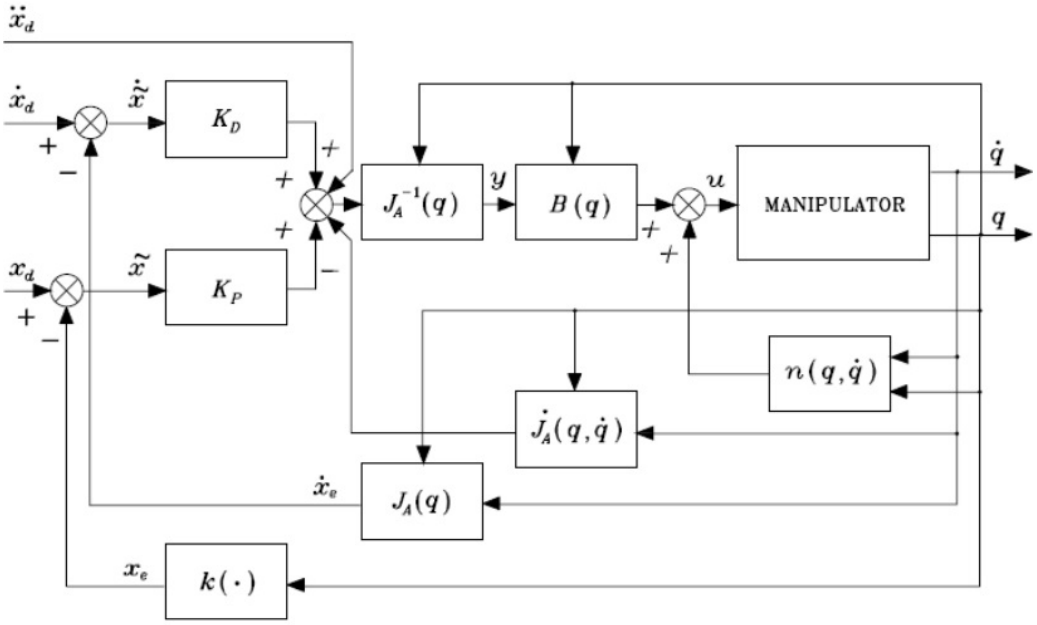


Figure: Operational Space Inverse Dynamics block scheme

Figure 18:

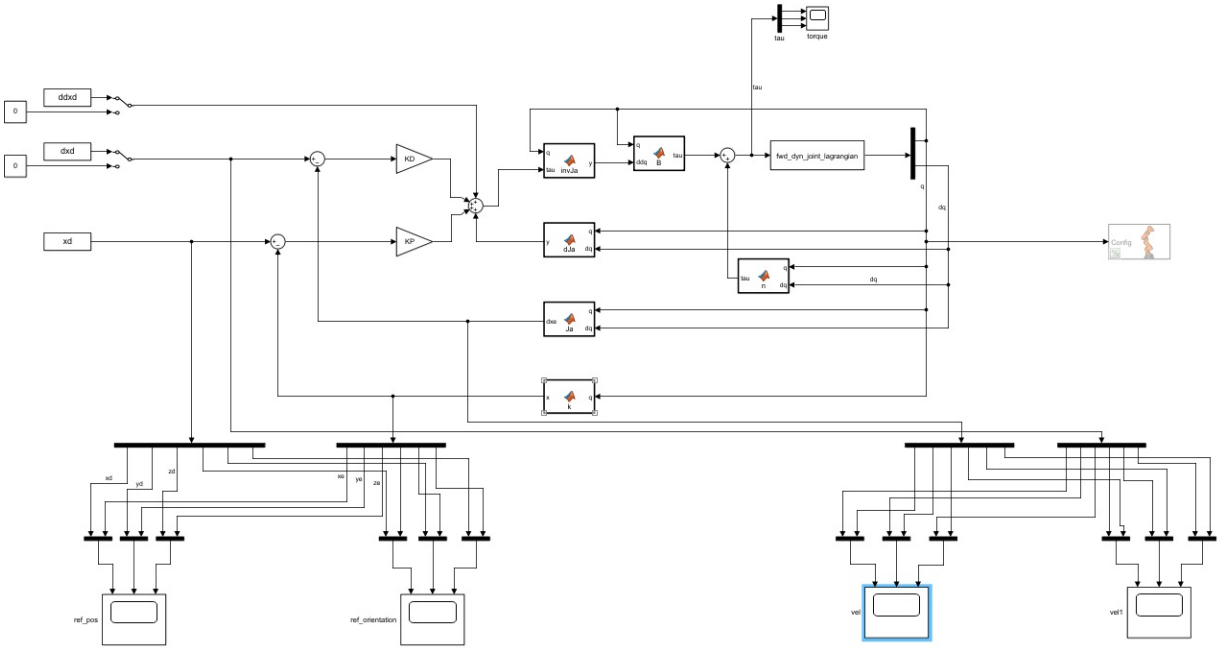


Figure 19: Simulink : Operational Space Inverse Dynamics

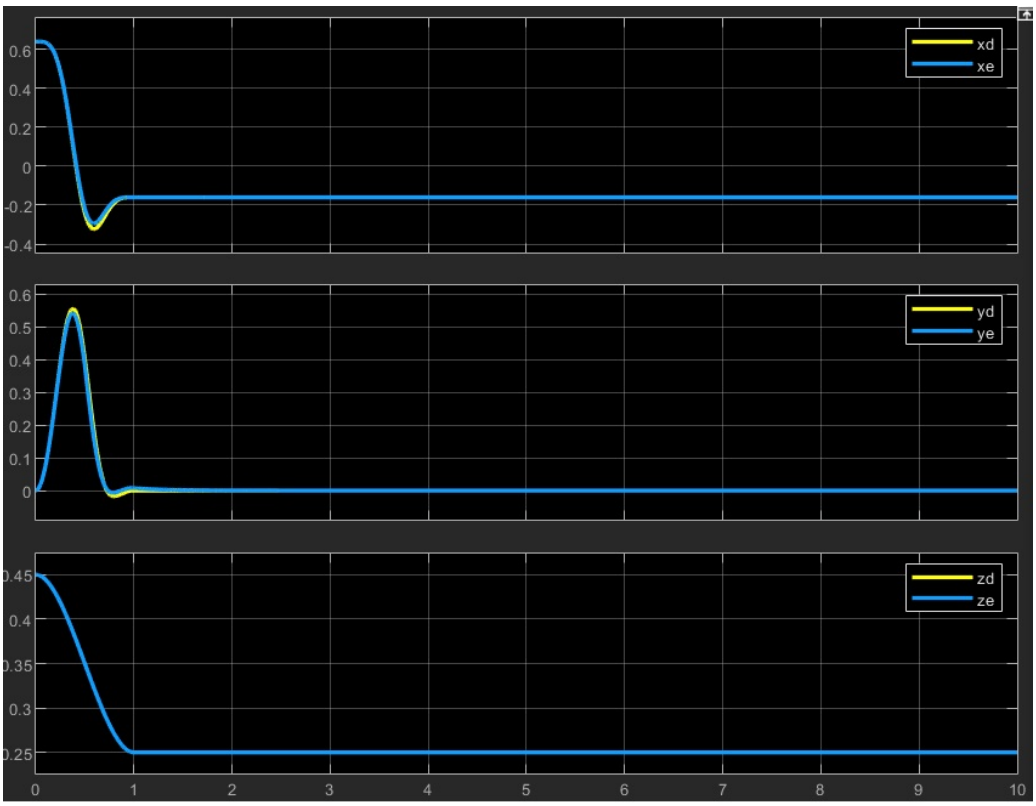


Figure 20: Scope position: Operational Space Inverse Dynamics

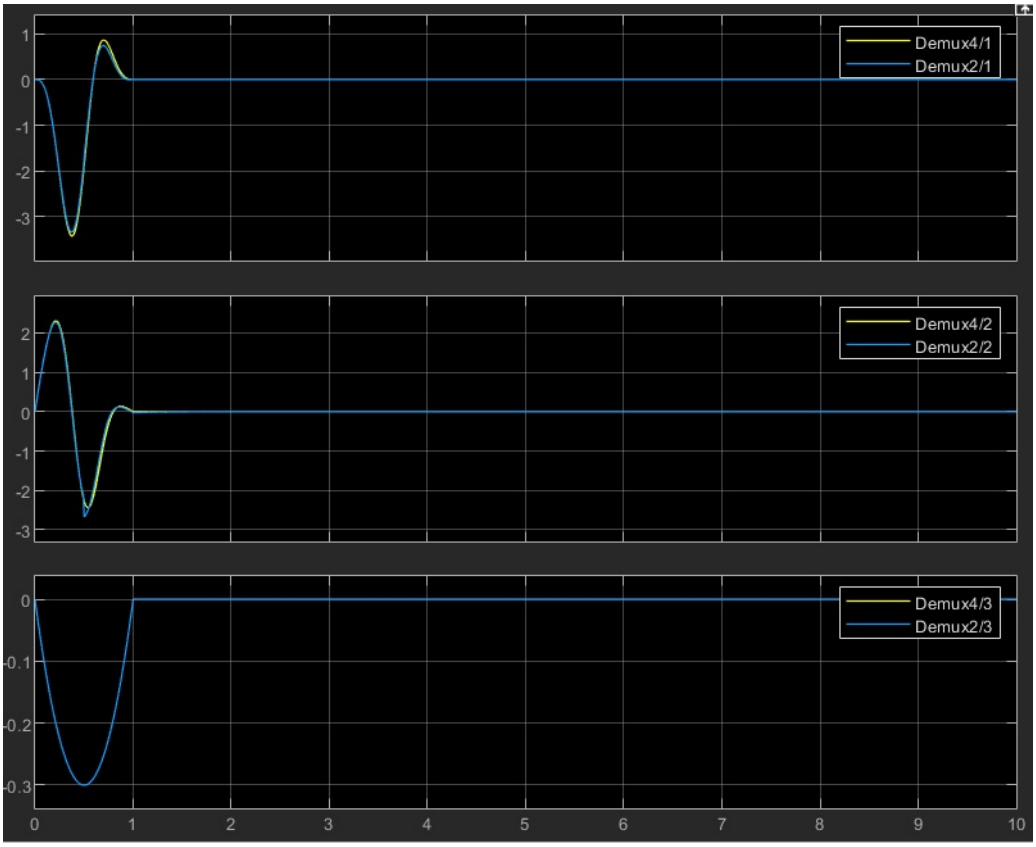


Figure 21: Scope Velocity: Operational Space Inverse Dynamics

11 Assignment 11

11.1 Study the compliance control

The environment is simulated as a plane to [0.52;0.2;0.45].

In the following sections the different scenarios are presented for different values of stiffness K of the environment.

11.2 Compliance Control: $K << K_P$

In this case we have $K = 1$ and as we can see in Figure 22 the desired position is almost reached.

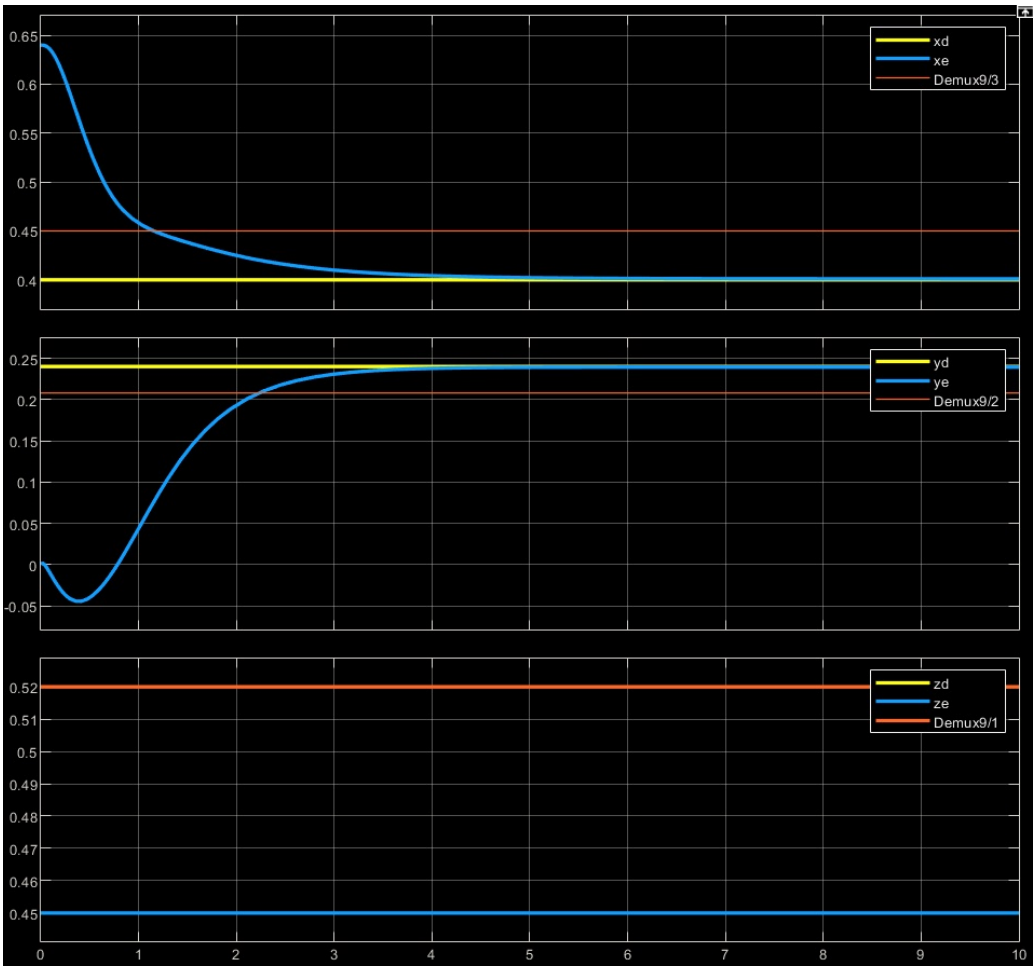


Figure 22: Compliance Control: $K << K_P$

11.3 Compliance Control: $K >> K_P$

In this case we have $K = 100$ and as we can see in Figure 23 the end effector stops when get in contact with the environment and cannot move forward.

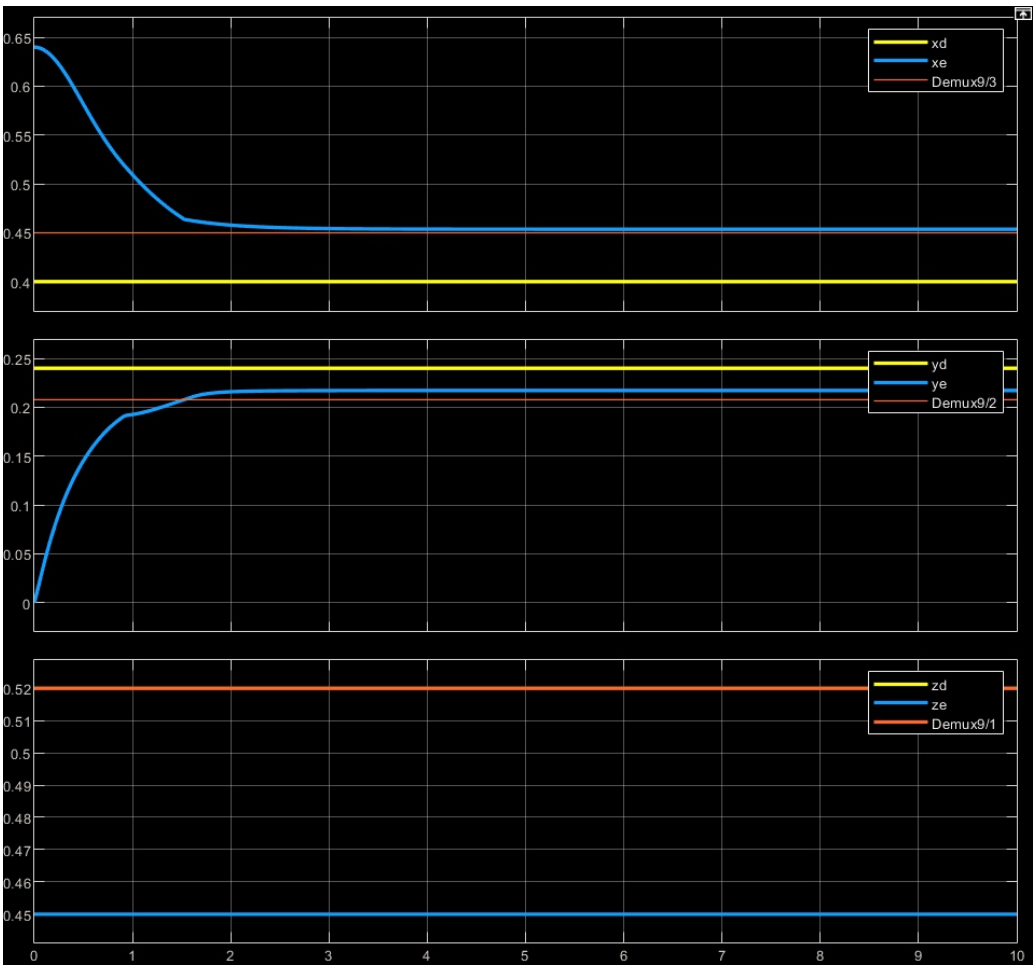


Figure 23: Compliance Control: $K >> K_P$

11.4 Compliance Control: $K == K_P$

In this case we have $K = 50$ and as we can see in Figure 24 the end effector exeed the environment while getting some resistance and it stops half way through the desired position.

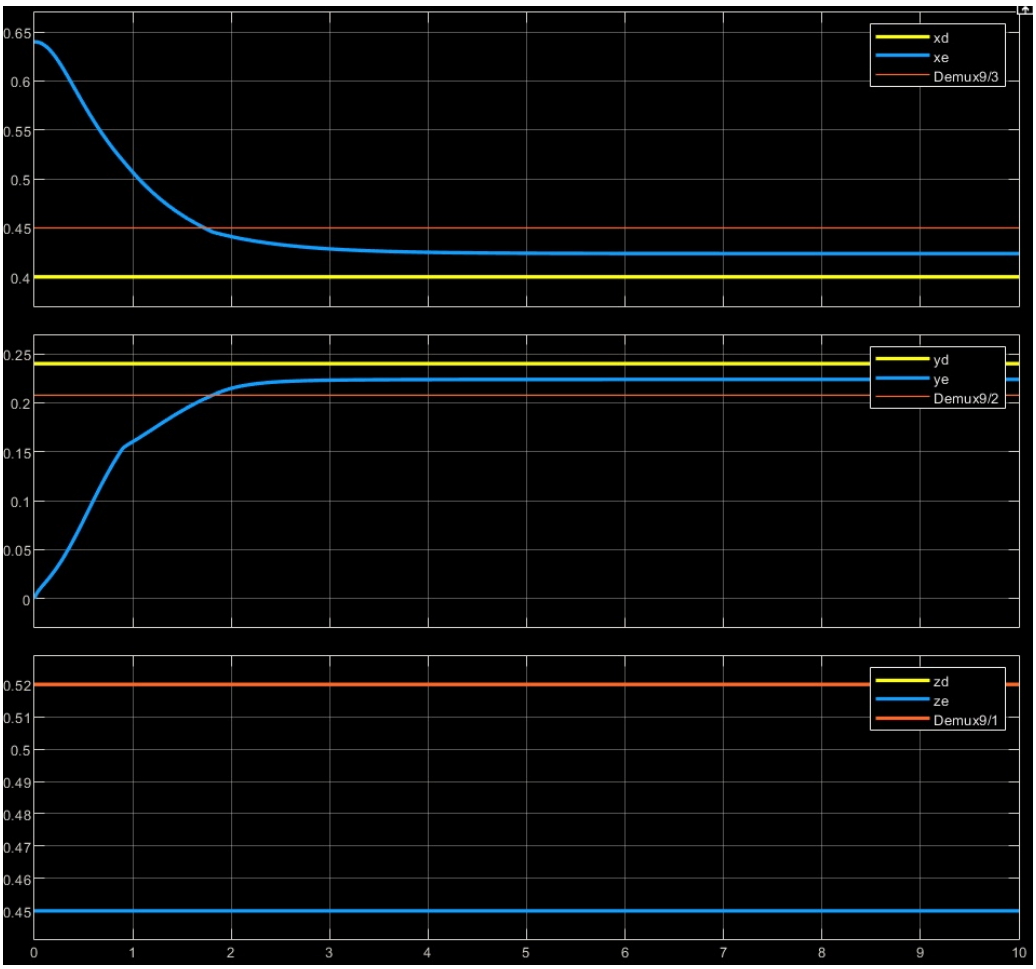


Figure 24: Compliance Control: $K == K_P$

12 Assignment 12

12.1 Implement the impedance control in the operational space

The schema can be seen in Figure 25.

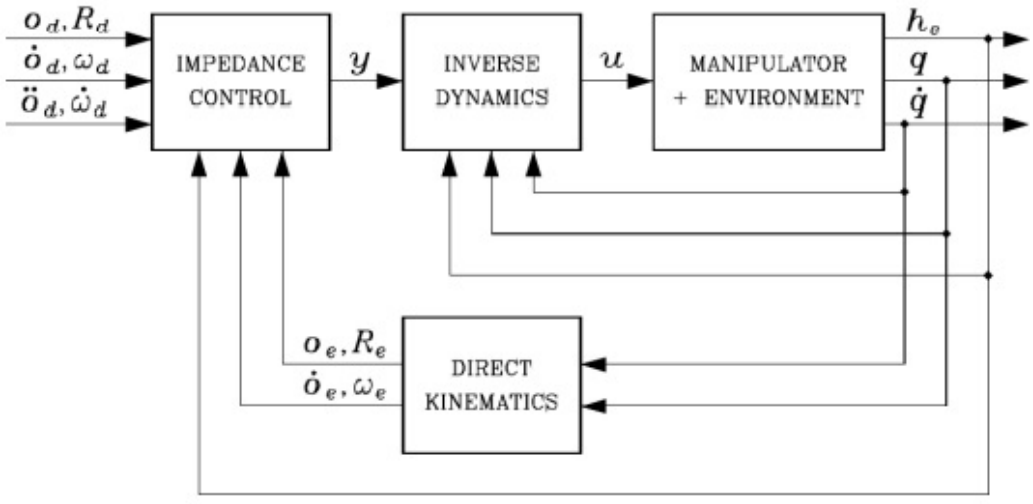


Figure: Block scheme of impedance control.

Figure 25:

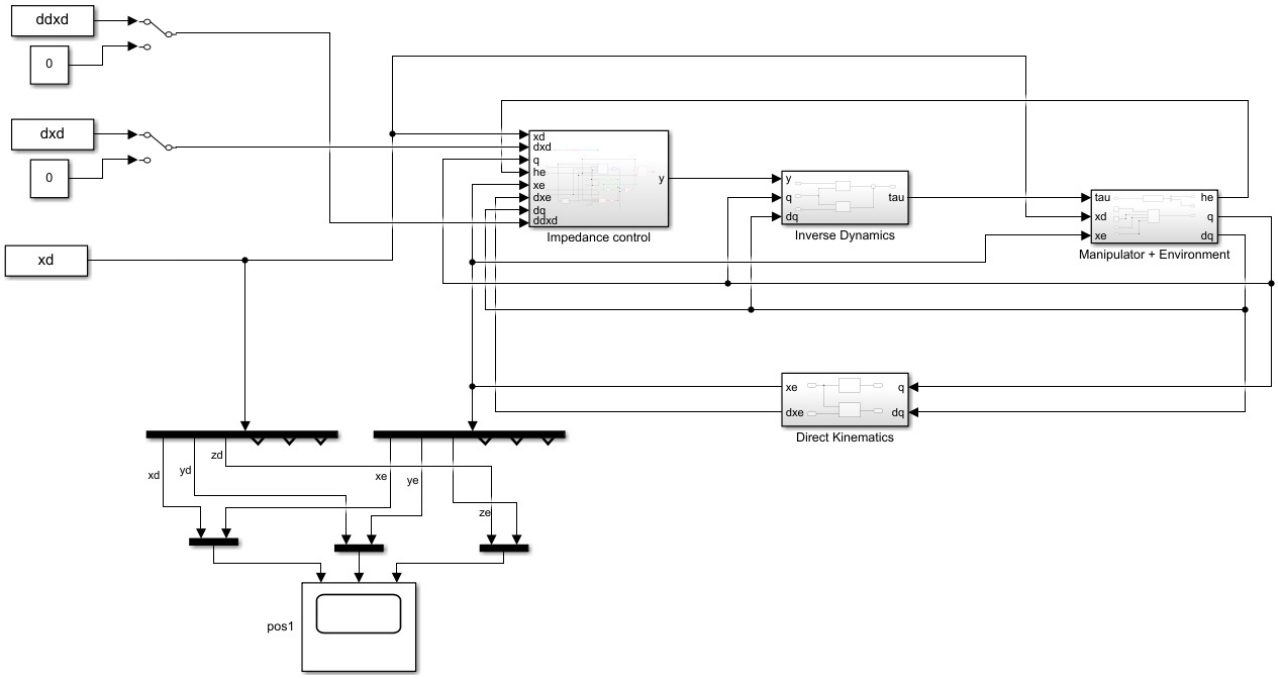


Figure 26: Simulink : Impedance Control

The environment is described as before but this time on the x and y direction we have $K_P = 50$ and $K = 100$. Also we can see that in z graph it start with a overshoot but with a value very low.

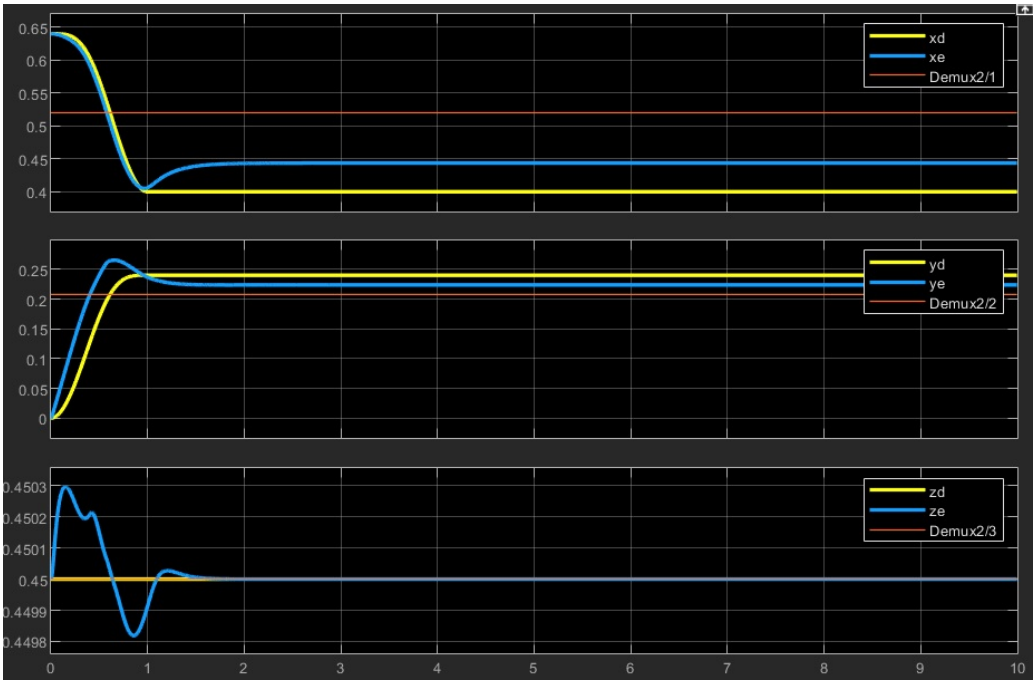


Figure 27: Scope :Impedance Control

13 Assignment 13

13.1 Implement the admittance control in the operational space

The schema can be seen in figure 28.

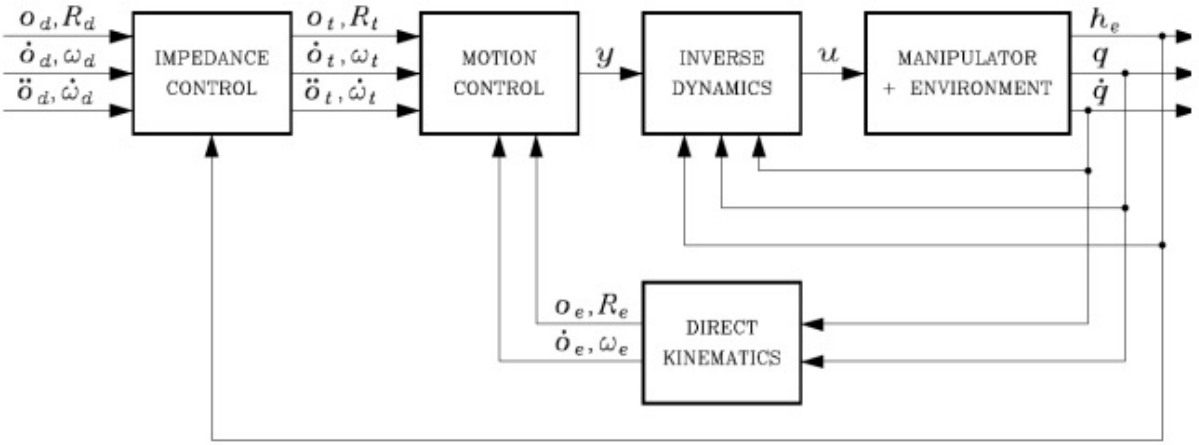


Figure 28: Admittance Control

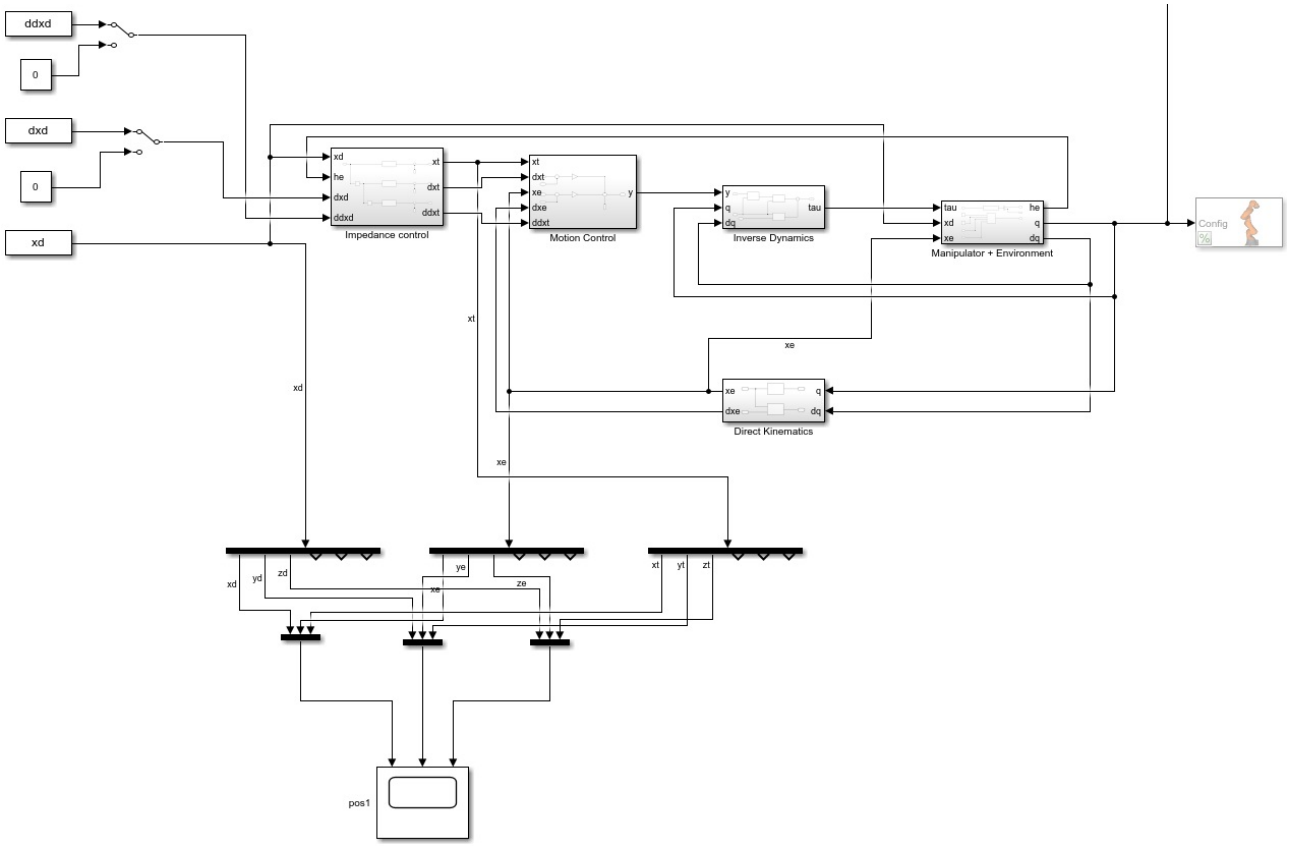


Figure 29: Simulink : Admittance Control

We have same situation as with the impedance about the position of the environment and the trajectory used. In the admittance model we have an admittance block that generates a new reference trajectory x_t based on the desired trajectory and the contact force with the environment. This new trajectory is then fed to a motion control. We can see in Figure 30 that again after the contact the trajectory in $x\&y$ direction is following x_t and no longer x_d. Here the ratio between K_P and K is 50 to 20

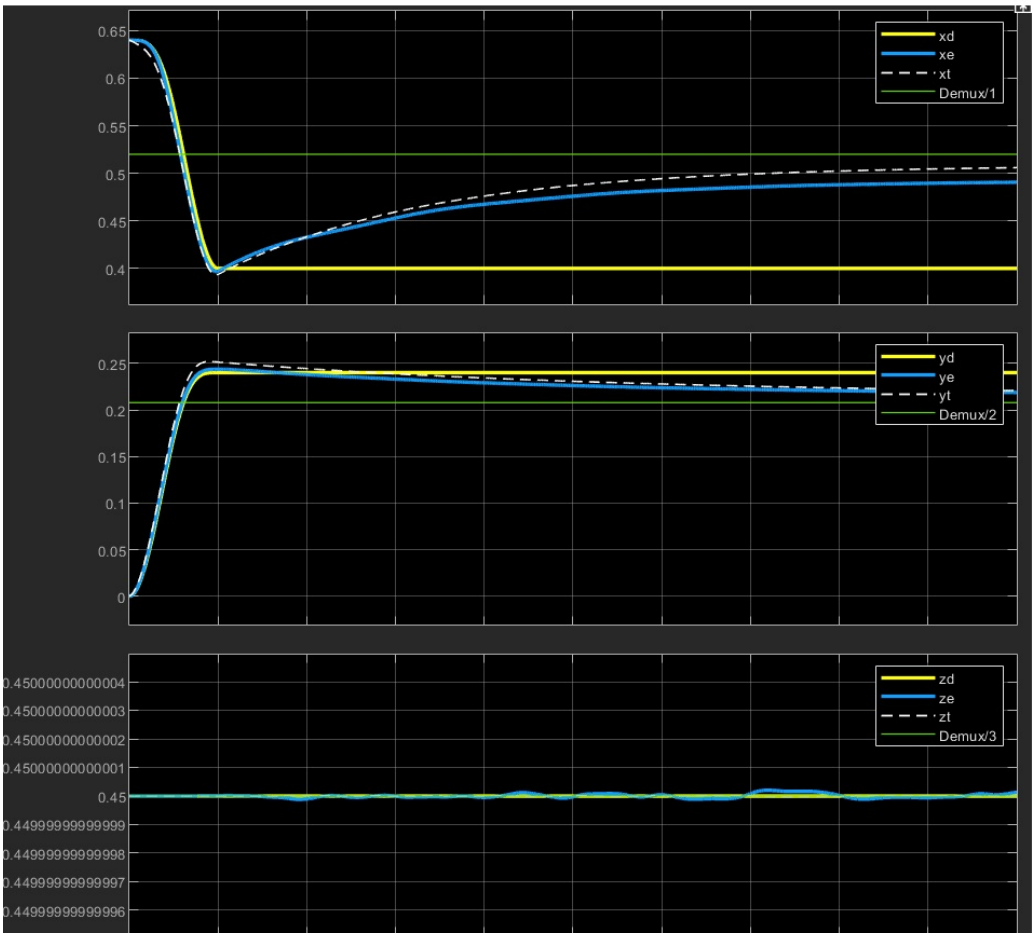


Figure 30: Scope : Admittance Control

14 Assignment 14

14.1 Implement the Force Control with Inner Position Loop

The simplified version of the schema can be seen in Figure 31.

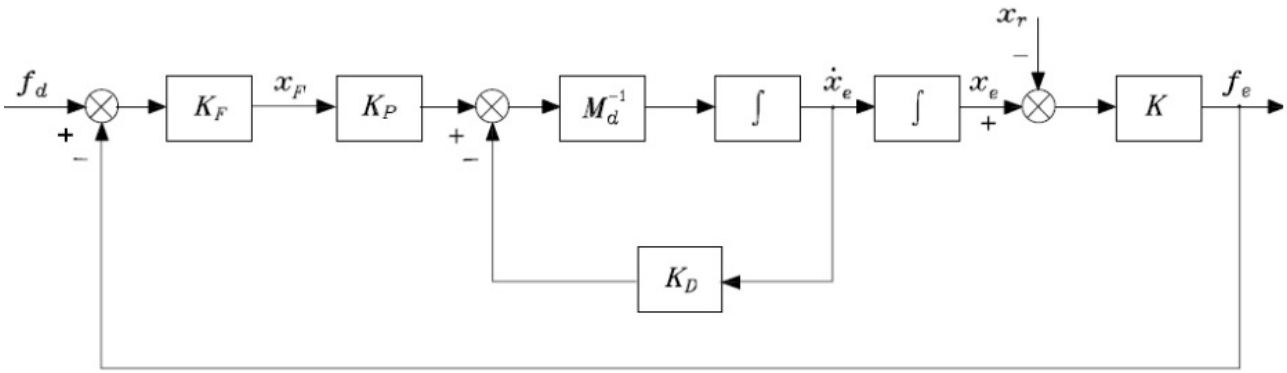


Figure: Block scheme of force control with inner velocity loop.

Figure 31: Force Control: $C_F = K_F$

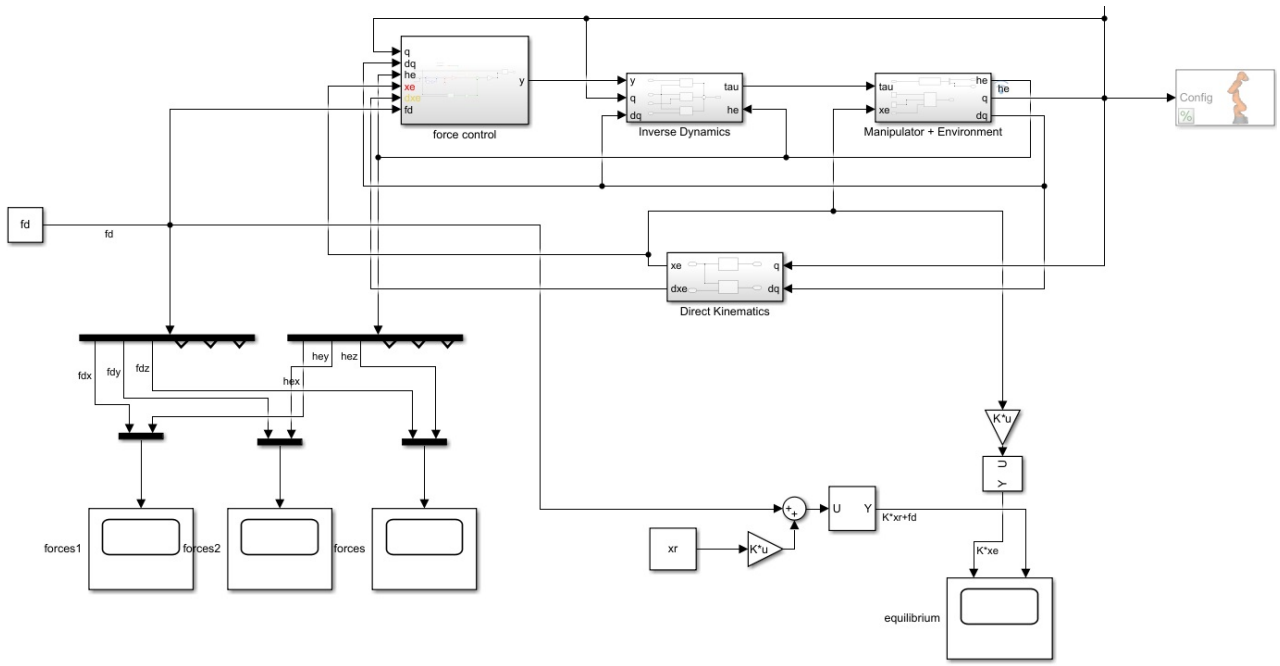


Figure 32: Force control

The controller is only in force we will show in the x and y direction.

Using or not the integral force controller we have always a steady state error as we can see in Figures below. The gain value is $K_F = 5$.

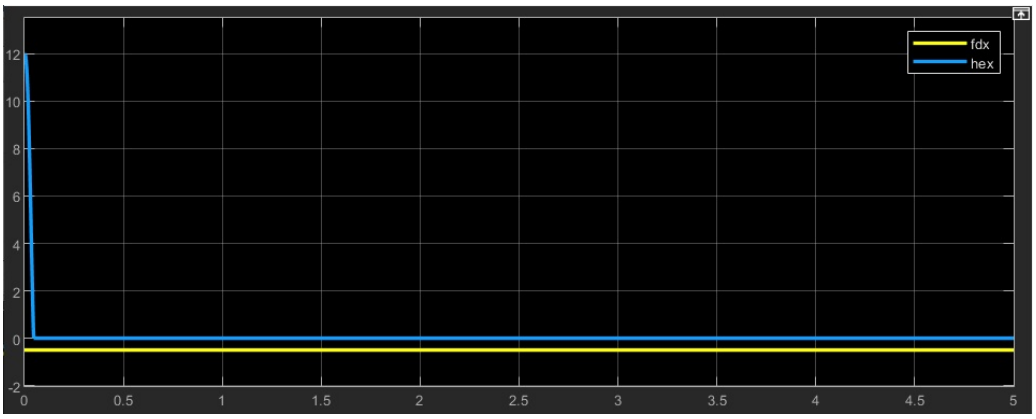


Figure 33: Force Control: Fdx

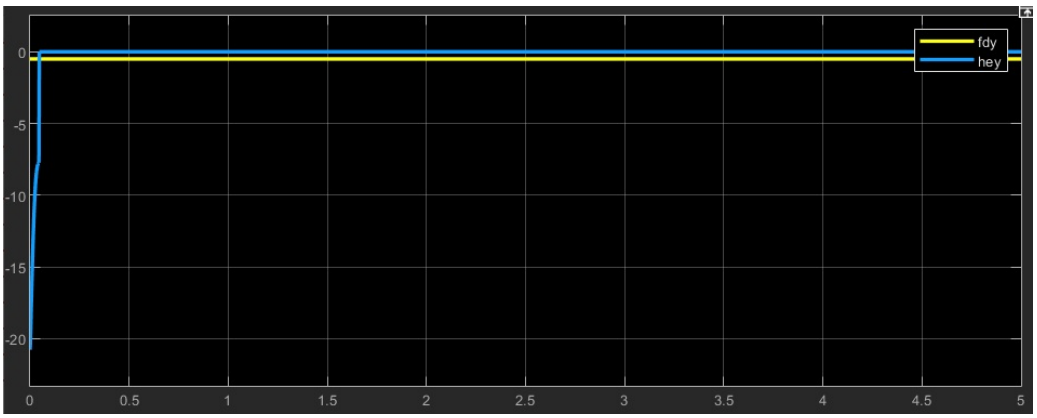


Figure 34: Force Control: Fdy

The proportional gain is still the same while the integral is $K_I = 4$, $C_F = K_F + \frac{K_I}{s}$.

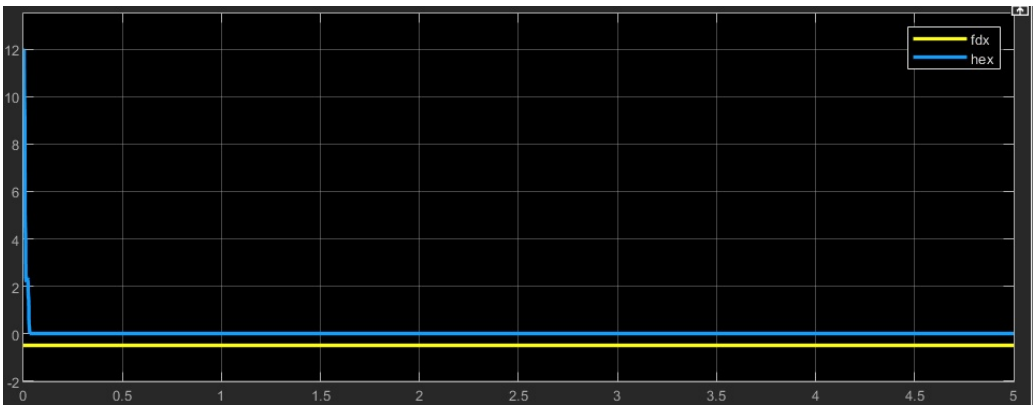


Figure 35: Force Control: Fdx

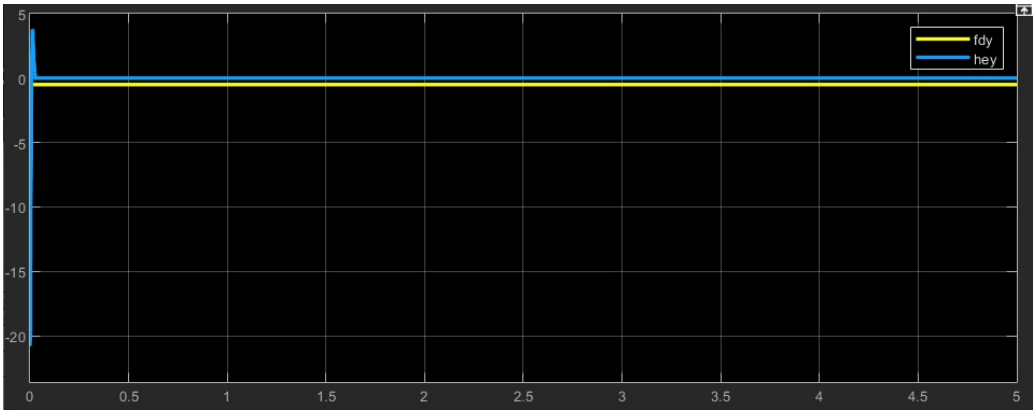


Figure 36: Force Control: Fdy

15 Assignment 15

15.1 Implement the Parallel Force/Position Control

The simplified version of the schema can be seen in Figure 37.

- The gains used are: $K_D = \begin{bmatrix} 80 \\ 80 \\ 80 \end{bmatrix}$, $K_P = \begin{bmatrix} 50 \\ 50 \\ 50 \end{bmatrix}$
- mass desired : $Md = \begin{bmatrix} 0.3 \\ 0.2 \\ 0.1 \end{bmatrix}$
- position $q = \begin{bmatrix} 0 \\ -0.2 \\ -pi/4 \end{bmatrix}$

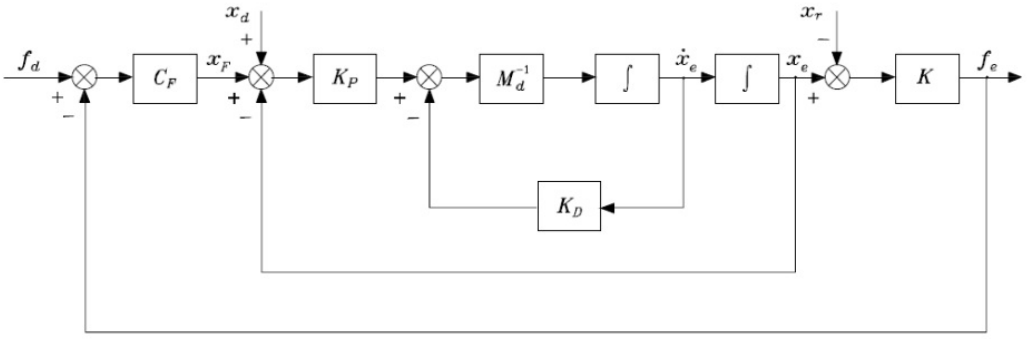


Figure: Block scheme of parallel force/position control.

Figure 37: Block scheme : Parallel Force/Position Control

The environment is still the same but this time the goal is to control the robot in force along the x and y directions and z in position control.

We can see in this exact behaviour where x and y try to reach their desired positions but find the environment and so forced with a desired force of $fd = [-0.5 - 0.50]$. It possible to see the model in Figure 38 and 39, while z reach its position.

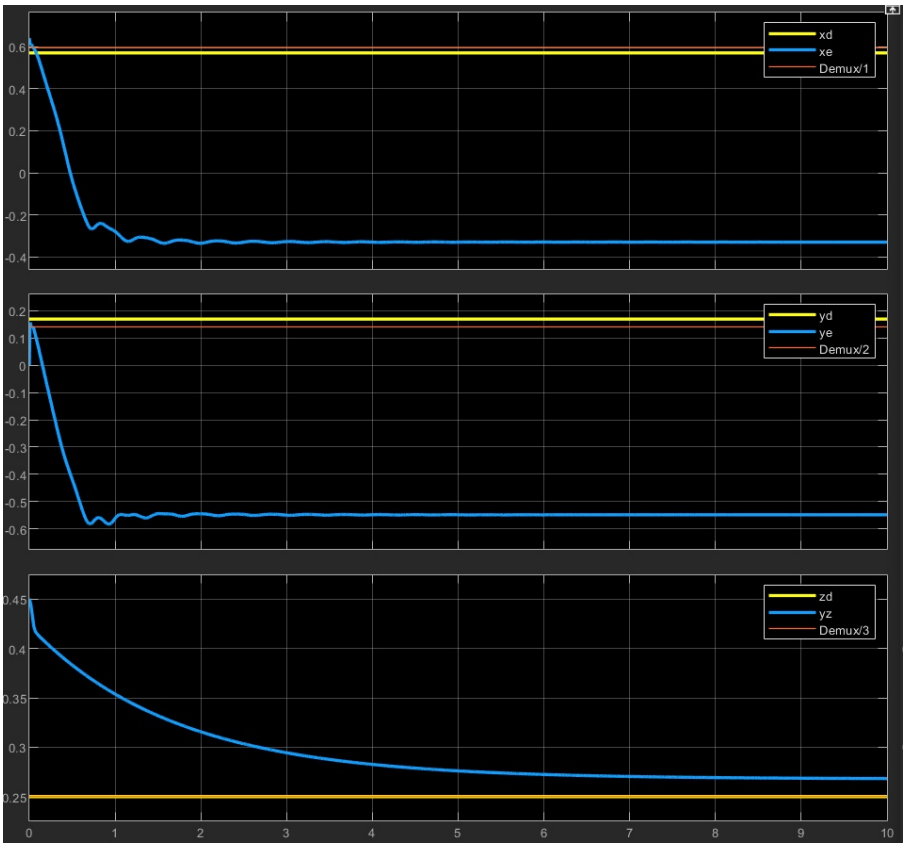


Figure 38: Position Scope : Parallel Force/Position Control

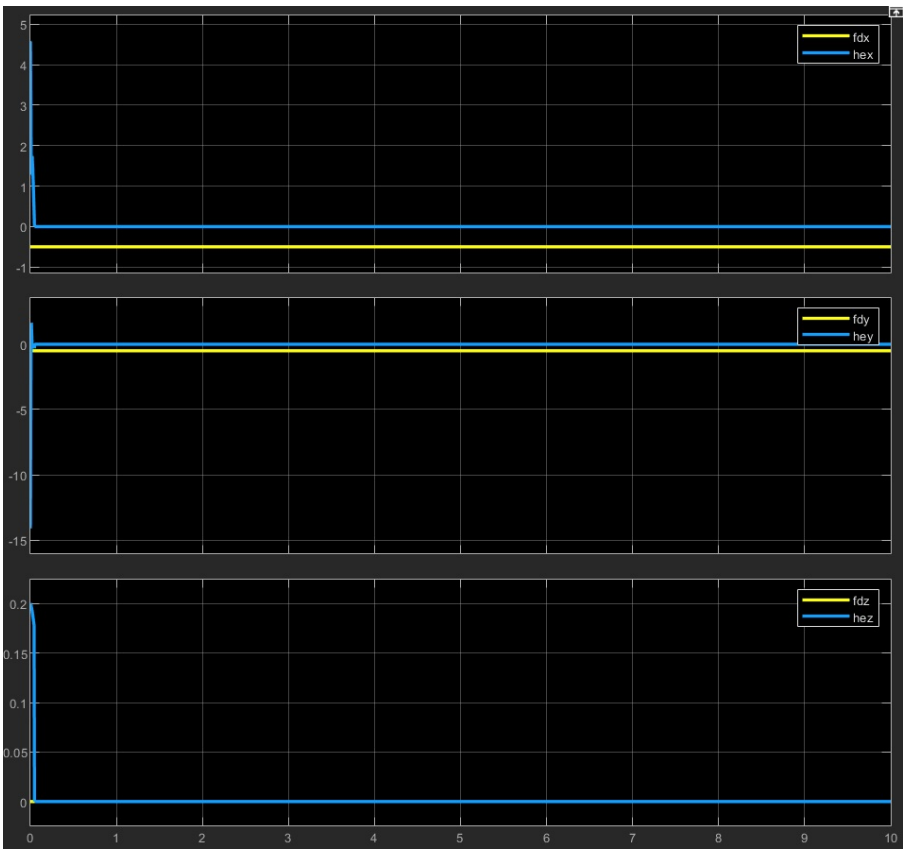


Figure 39: Force Scope : Parallel Force/Position Control

We can test the system using a PI controller,$KI = 4$, while it reaches the desired force control X and Y direction.

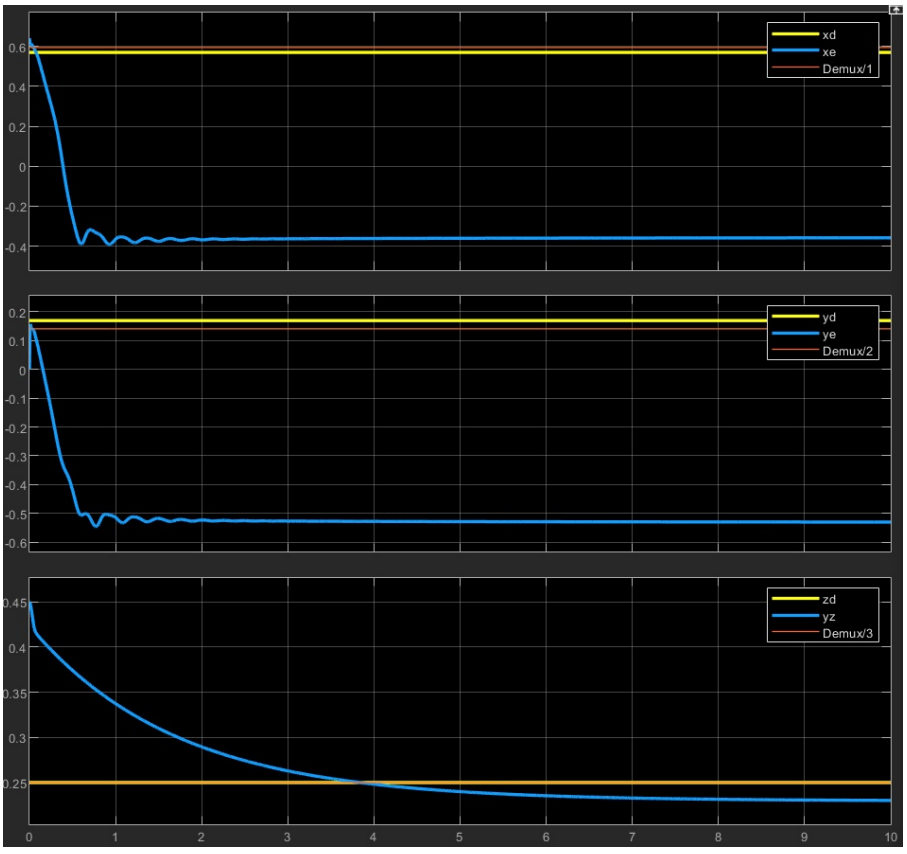


Figure 40: Position Scope with PI: Parallel Force/Position Control

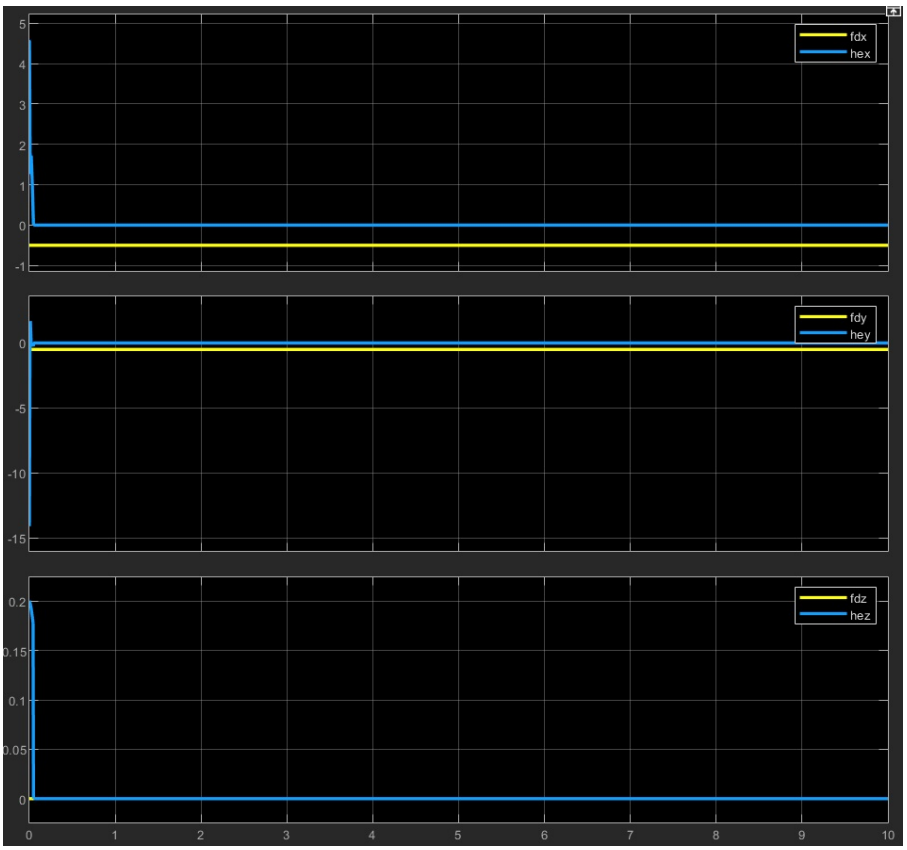


Figure 41: Force Scope with PI: Parallel Force/Position Control