Homework 3: Numerical Simulations

Vitetta Emanuele;
2082149

September 17, 2023

In the delivered folder, there are a few sub-folders. Each of them contains all the necessary code to repeat the simulations. The original given files have been modified or renamed in order to solve all the exercises. Remark: in order too see all the details of the images, you have to zoom the pdf a lot (figures have a great resolution)

1 Control of the SCARA robot

In this section, we are going to analyze two different controllers: the first one is a PD with gravity compensation, while the second one is based on feedback linearization. In all these cases we do not derive the dynamics of the SCARA robot, since they are already available from the numerical experiences.

1.1 PD with gravity compensation - target postion

We implemented a controller following this equation

$$\tau = K_P e + K_D \dot{e} + g(q) \tag{1}$$

where $e = q_d - q$. In this case, as suggested in the handout, we set

$$K_D = K_P = 1000 I_{4\times 4} \tag{2}$$

Below we can see both the Simulink schematics of the controller (see fig. 1) and a plot of the numeric results obtained (see fig. 2). Looking at this second image we can conclude that, starting from the initial condition $q = [\ 0 \quad 0 \quad 0 \quad 0\ ]$ we can achieve the desired position $q = [\ \pi/2 \quad \pi/3 \quad 0.1 \quad -\pi/4\ ]$. We can also notice in fig. 2 that, as expected, all joint velocities go to zero (as well as the accelerations). We can also appreciate the behaviour of the four components of e.

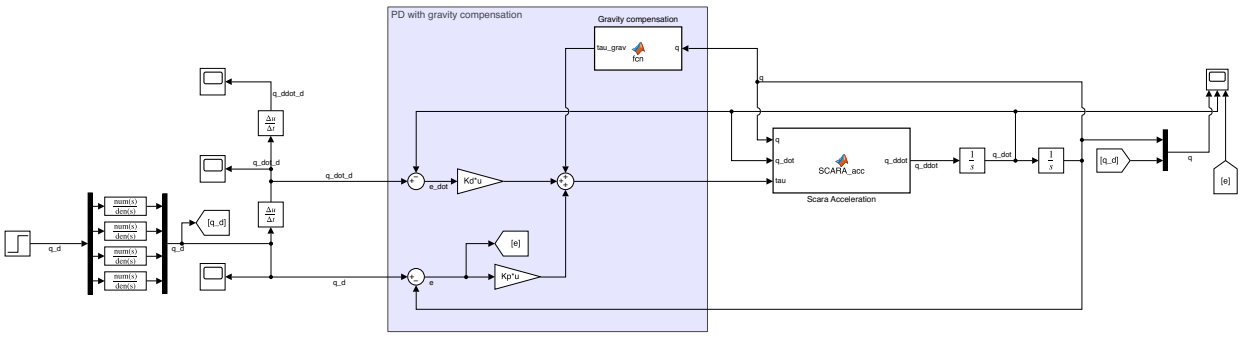

Figure 1: PD controller with gravity compensation

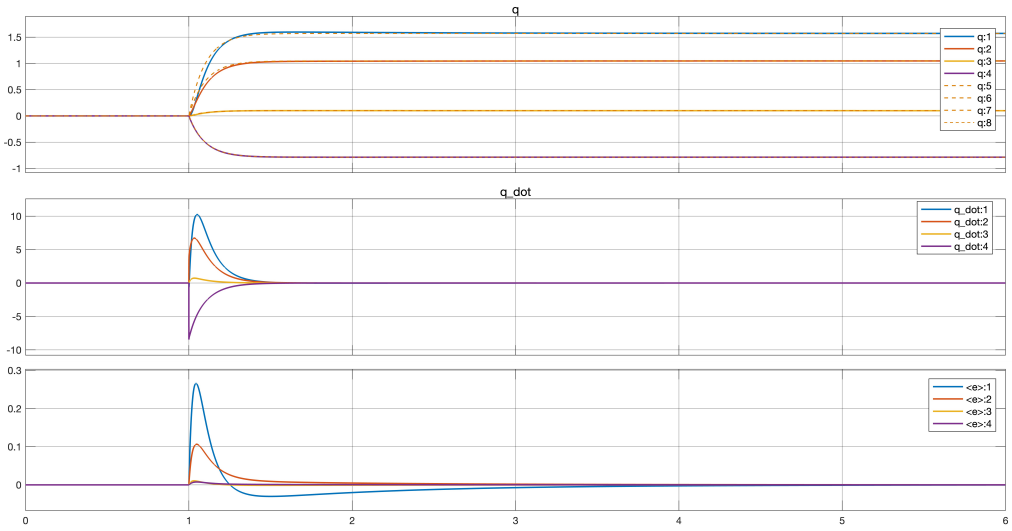

Figure 2: Numerical simulation

1.2 PD with gravity compensation - trajectory tracking

In this part, we used the controller designed in (1) in order to track asymptotically a trajectory made by a sinusoidal signal. In order to achieve satisfactory performances we had to adapt the controller gains K_P and K_D depending on the frequency of the sinusoid we want to track.

The Simulink scheme used for this section is reported in fig. 3. We can first notice that there is a main difference with respect to the one in the previous exercise (see fig. 1). In fact, the discrete derivative has been replaced with a *real derivative*, whose transfer function is given by

$$\frac{\omega_c^2 s}{s^2 + 2\delta\omega_c s + \omega_c^2} \tag{3}$$

where $\delta = 1/\sqrt{2}$ and $\omega_c = 2\pi 50$ rad/s. By using this strategy the simulation runs faster (but doesn't work in real-time, though) and does not require generating a sampled trajectory using *casADi*. The reference is generated, instead, by a Simulink block with a sampling of 1ms.

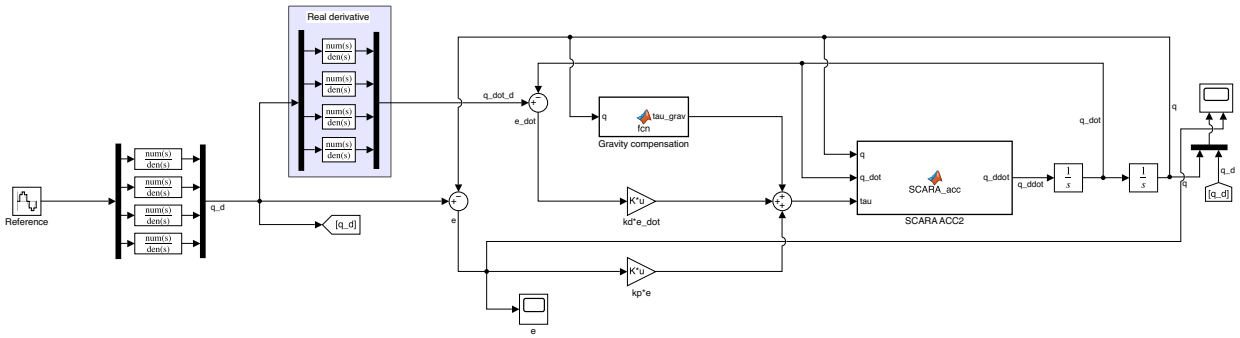

Figure 3: PD controller with gravity compensation

In figures. 4-7 reported below, we can observe the behaviour of the system considering the sinusoidal input. In each figure, we can see both the real trajectory compared to the reference one and the error e. In order to keep the error with an acceptable magnitude, we had to change gains when pushing on the frequency of the input signal. In table 1 we also report the choice of the gains of the controller (each of them multiplies a 4×4 identity matrix). It is clear that this strategy is not practically realizable. For high frequencies gains become unbounded.

ω [rad/s]	K_P	K_D
0.5	2000	4000
3	6000	10000
6	12000	60000
10	40000	120000

Table 1: Gains used in the PD controller

It is possible to observe that computation time is strongly influenced by the magnitude of the gain, in particular by K_D. We can also notice from the figures below that the error never goes to zero, but keeps oscillating in a certain interval. Anyway, such error is very small if compared to values of q but it is not negligible.

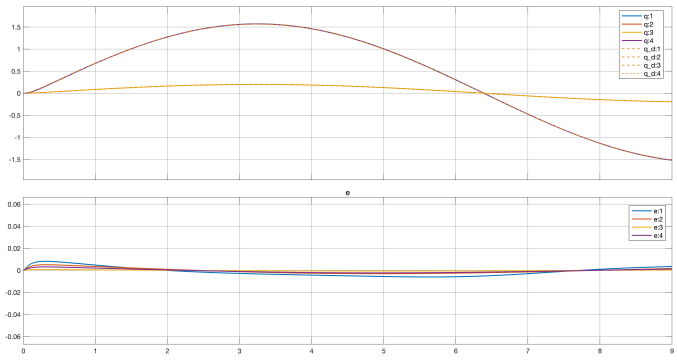

Figure 4: $\omega = 0.5$ [rad/s]

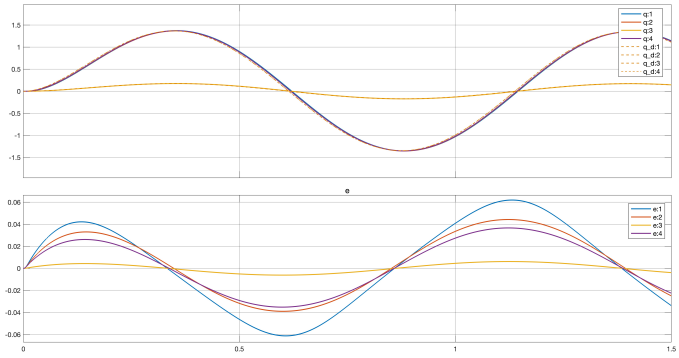

Figure 5: $\omega = 3$ [rad/s]

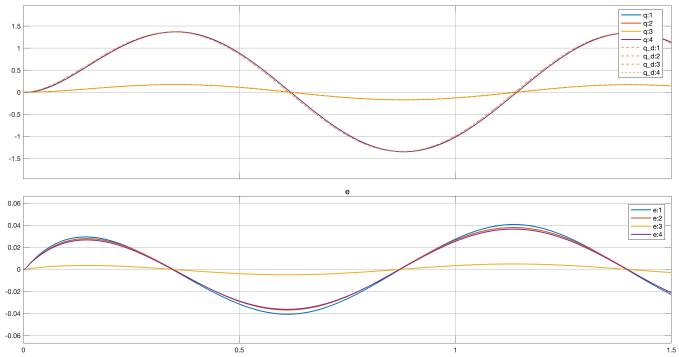

Figure 6: $\omega = 6$ [rad/s]

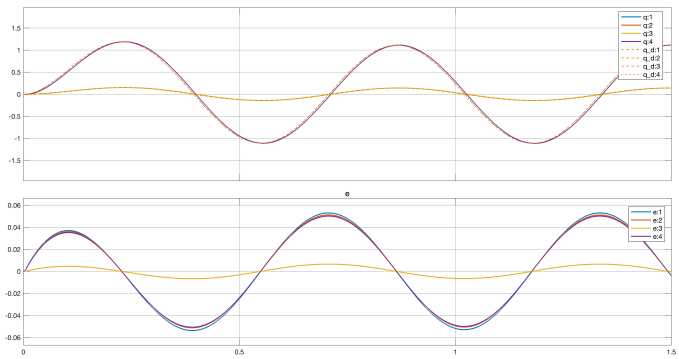

Figure 7: $\omega = 10$ [rad/s]

1.3 Feedback linearization

In this section, we exploit the feedback linearization controller, based on the following expression.

$$\tau = B(q)\left(K_P e + K_D \dot{e} + \ddot{q}_d\right) + C(q, \dot{q}) + g(q) \tag{4}$$

Notice that the expressions of the matrices $B(q)$, $C(q, \dot{q})$ and $g(q)$ were already available. They are used inside few matlab functions to evaluate the various terms appearing in the controller. Notice also that the real derivative defined in (3) has been used twice in this exercise. In fact, we need to differentiate q_d two times in order to get $\ddot{q}_d$.

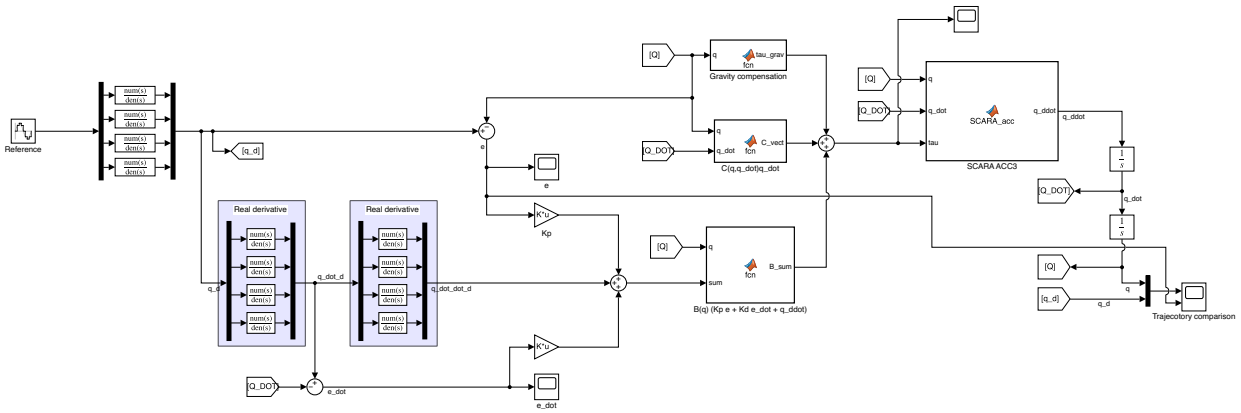

Figure 8: Feedback linearization controller

Now we consider the behaviour of the robot for the same values of ω tested in the chapter before. Furthermore, we consider two different initial conditions for the simulations. We consider

$$q_1(0) = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix} \tag{5}$$

whose results are reported in in figures 9-12, and

$$q_2(0) = \begin{bmatrix} \pi/2 & \pi/4 & 0 & 0 \end{bmatrix} \tag{6}$$

whose results are reported in in figures 13-16. For each ω we used as gains $K_P = K_D = 1000 I_{4\times 4}$. This clearly shows that feedback linearization is a powerful technique when performing trajectory tracking. In fact, using lower gains, we can use a smaller control action, which is more convenient. Notice that the error e is small for all ω. By pushing a little the two gains we can get even better results (but as side effects, computations will be a slight bit slower).

We can just make a few considerations about how the initial condition and ω influence the tracking. If we start from $q_1(0)$, for every ω we can notice that the tracking is almost perfect, with a small error that increases with the oscillating frequency. On the other hand, if we start from $q_2(0)$ the system needs a few oscillations period to start tracking the trajectory with a small error. As well as before, performances are better for a smaller ω.

1.4 Solutions for $q(0) = [\ 0 \quad 0 \quad 0 \quad 0\ ]$

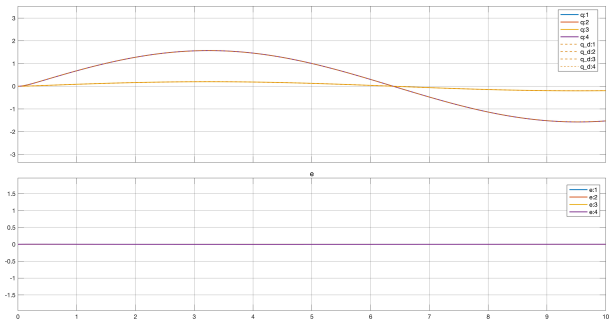

Figure 9: $\omega = 0.5$ [rad/s]

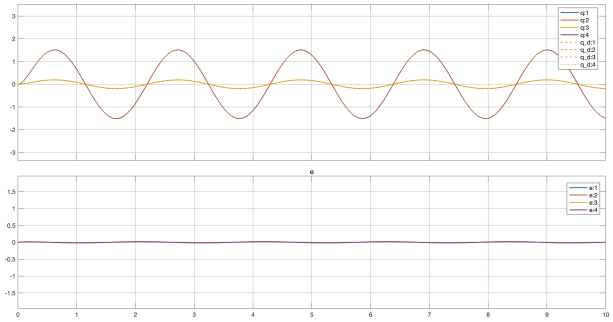

Figure 10: $\omega = 3$ [rad/s]

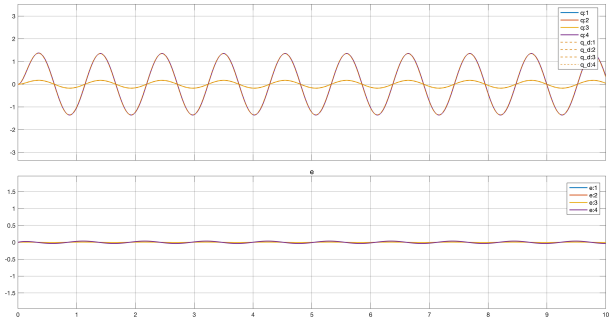

Figure 11: $\omega = 6$ [rad/s]

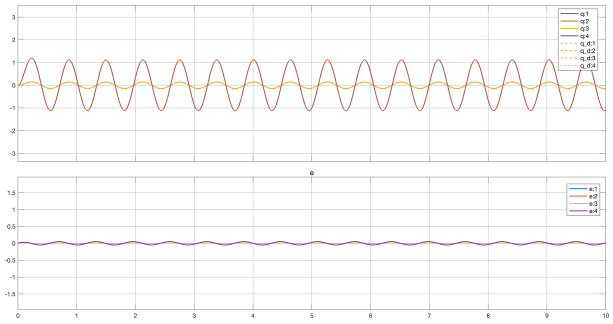

Figure 12: $\omega = 10$ [rad/s]

1.5 Solutions for $q(0) = [\ \pi/2 \quad \pi/4 \quad 0 \quad 0\ ]$

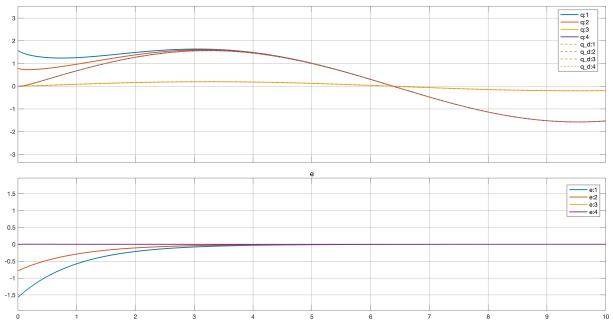

Figure 13: $\omega = 0.5$ [rad/s]

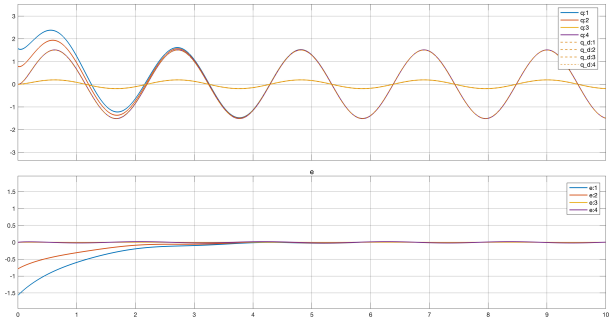

Figure 14: $\omega = 3$ [rad/s]

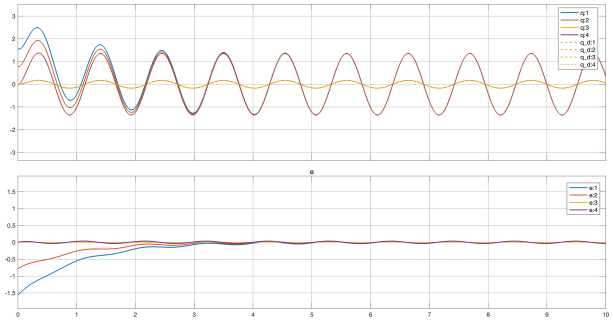

Figure 15: $\omega = 6$ [rad/s]

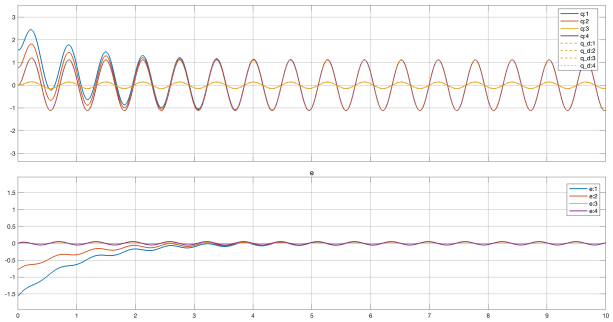

Figure 16: $\omega = 10$ [rad/s]

2 Kinematics derivation using *casADi*

In the submitted folder, you can find four subfolders, each of which corresponds to one of the homework requests. Specifically, in folder *2.1*, you will find all the required files. The file **Solutions.mat** contains the three calculated variables: *POS*, *ANG*, and *TAU*. If you want to evaluate again the solutions, just run *UR_10.mlx* (this is going to overwrite all the solutions).