HOMEWORK #3 ROBOTICS AND CONTROL 1

Davide Peron 2082148

Academic Year 2022/2023

Introduction

In this introductory section I will explain what is included in the *.zip* folder delivered; each point and sub-point of the exercise has its own dedicated folder containing all the necessary files required to run the simulation, to test and eventually reproduce the results presented in this report. In each folder there are also the *casADi* functions used and their compiled files in *Windows*.
It is important to note that some figures may not have clearly visible results due to the quality of the image inserted and the overlapping of some trajectories, so if some results are not clear then the simulation can be rerun to appreciate the result through the scopes in *Simulink*.

1 Control of the SCARA robot

In this section two different control strategy for the SCARA robot will be evaluated; the first set of tests will be conduced using a PD with gravity compensation, both to reach a desired position and to track a sinusoidal trajectory. In the last part a different controller has been developed, namely a feedback linearization controller, and it will be compared to the previous one when tracking sinusoidal trajectories.

1.1 PD controller with gravity compensation: Step

To complete this task we are required to design a PD controller with gravity compensation, and then test it while starting from initial conditions $q = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}$ and reaching the target position $q = \begin{bmatrix} \frac{\pi}{2} & \frac{\pi}{3} & 0.1 & -\frac{\pi}{4} \end{bmatrix}$.
This controller has the following control law:

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

with $e = q_d - q$ and the proportional and derivative gain set both to 1000 as requested in the assignment. The Simulink implementation of the controller can be seen in Figure 1, it has been develop modifying the structure given to produce the desired plots. Another main difference with the one presented in the numerical experience is the use of proportional and derivative gains, in place of the PID blocks, theoretically the result should be identical since both control scheme implement the same control law via different tools.

The next step is to analyze the results obtained while tracking the desired target position; these results are reported in Figure 2 where it is clear that the task can be completed successfully in approximately 4s and it is possible to see, form the last plot, that the error reaches zero, so this controller does a really good job while trying to track a desired target position.

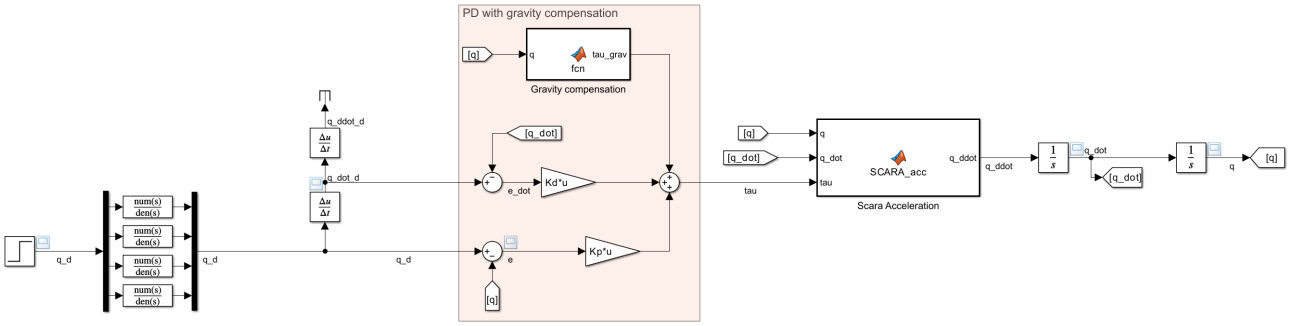

Figure 1: PD controller with gravity compensation: scheme to track a step reference

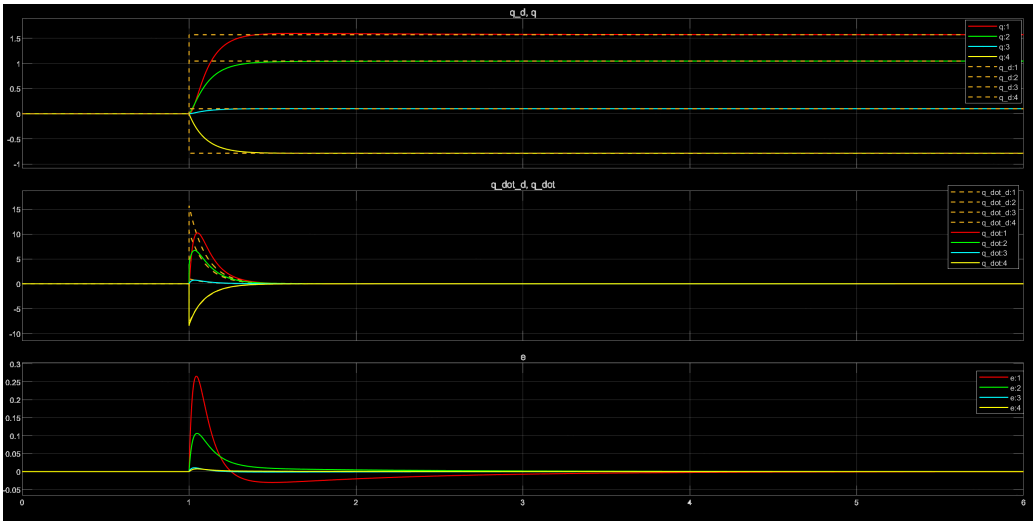

Figure 2: PD controller with gravity compensation: results tracking step reference

1.2 PD controller with gravity compensation: Sinusoids

In this second part It will be possible to see how the controller designed in the previous part behaves when trying to track a sinusoidal reference. As it is possible to see in Figure 3 the derivative block used in the previous scheme has been replaced with a set of derivative filters, namely the real derivative as seen in the Control Engineering Laboratory course since, using the gains instead of the PID blocks, made the simulation impossible to run, namely the system was not able to compute quickly enough with such high gains, however i decided to keep using this control scheme since it provided similar results to the one using a PID block, the response was almost the same but the PID had little more overshoot but a little faster response, so depending on the application, even if the two schemes theoretically should behave the same, one may be better than the other depending on if we need more raw speed or a more accurate approach. The derivative filter implemented has the following transfer function:

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

with $\delta = \frac{1}{\sqrt{2}}$ and $\omega_c = 2\pi 50$ rad/s; this filter has the same behaviour of a derivative block so the results should be really similar to the one using the original scheme but the model was able to simulate the system even though really slowly. The main task of this point was to adapt the proportional and derivative gains

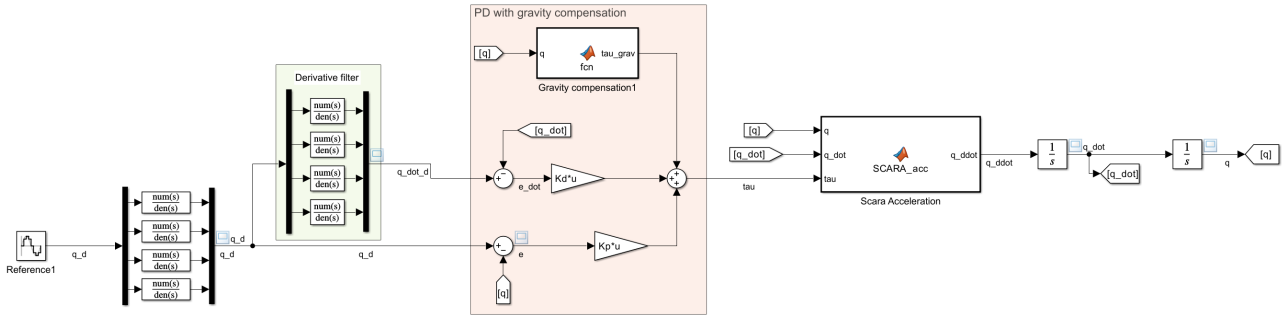

Figure 3: PD controller with gravity compensation: scheme to track a sinusoidal reference

of the controller to obtain satisfactory results when trying to rack sinusoidal signals at different angular velocities; As it is possible to see from Table 1 some of the gain especially at higher angular velocities are not physically implementable, moreover such high gains made the simulations really slow, this makes the controller a bad choice to use in a real time environment.

ω [rad/s]	K_P	K_D
0.5	3000	3000
3	3000	5000
6	3000	10000
10	3000	30000

Table 1: Proportional and derivative gains

We can now analyze the results of the simulation that can be seen from Figure 4 to Figure 7, we can clearly see that the controller is able to track the desired reference independently of the angular velocity of the reference but its clear that it cannot track the signals perfectly since the error (last plot of each Figure) never goes to zero but oscillates with magnitude in the order of 10^{-2} which is small compared to the magnitude of the signal but not null; this error is probably due to computational limitation and delays

since using high gains really renders the simulation difficult to run and computationally expensive.
As we have seen this controller is not enough if we want to track a sinusoidal signal in a real environment since it is too computationally demanding, both for real-time use and not implementable gains, but while using it in these simulation it proved to be reliable and offered good tracking performances within an acceptable limit for most use cases.

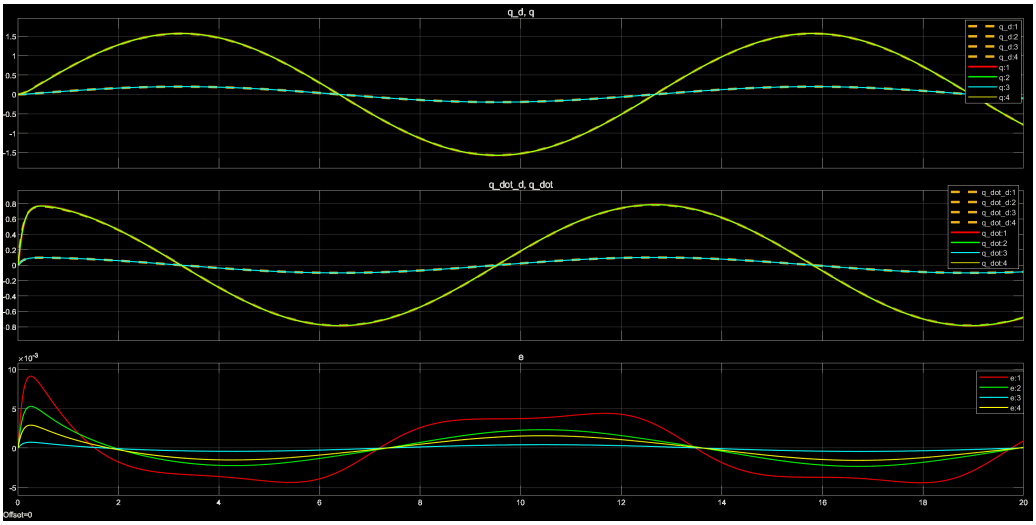

Figure 4: PD controller with gravity compensation: results tracking sinusoidal reference with $\omega = 0.5$rad/s

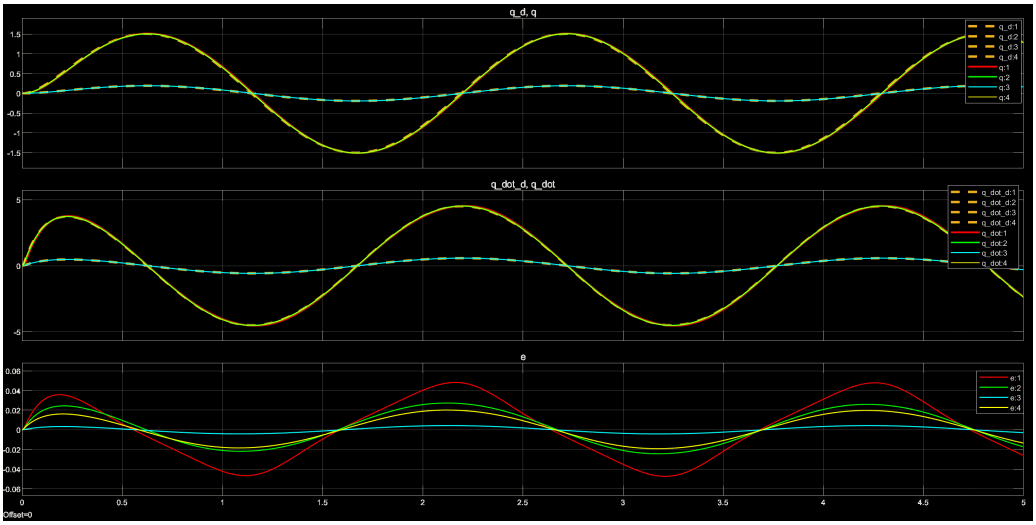

Figure 5: PD controller with gravity compensation: results tracking sinusoidal reference with $\omega = 3$rad/s

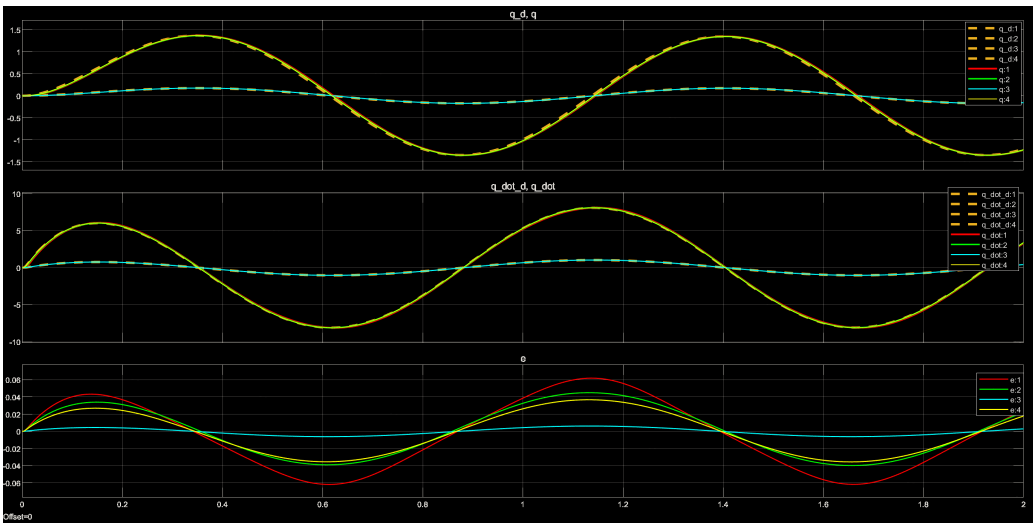

Figure 6: PD controller with gravity compensation: results tracking sinusoidal reference with $\omega = 6\text{rad/s}$

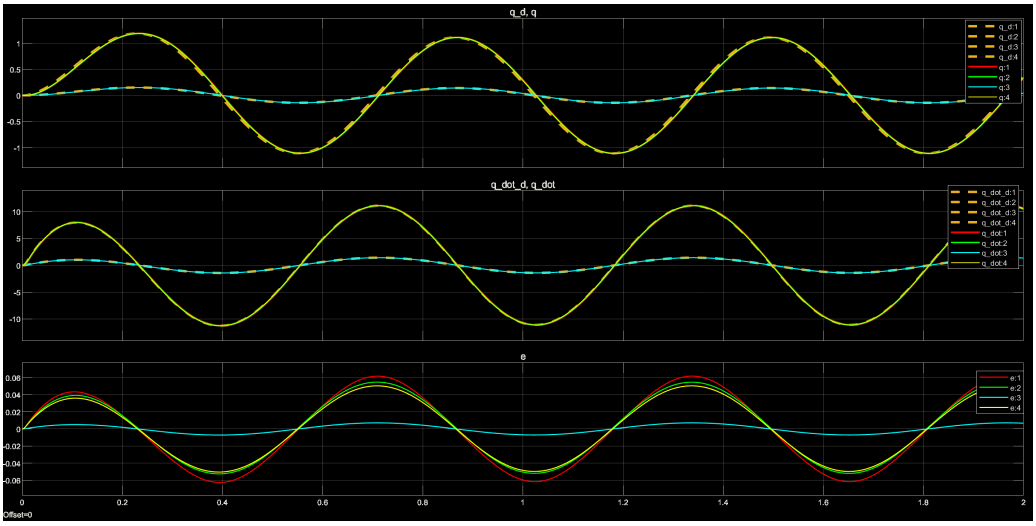

Figure 7: PD controller with gravity compensation: results tracking sinusoidal reference with $\omega = 10\text{rad/s}$

1.3 Feedback linearization controller

In this third part a new controller will be developed and tested with sinusoidal signals; this new controller is the feedback linearization one; to implement this controller we first need to define its control law which is:

$$\tau = B(q)\left(K_P e + K_D \dot{e} + \ddot{q}_d\right) + C(q, \dot{q}) + g(q) \tag{3}$$

The implementation of the controller can be seen in Figure 8, as we can see I used the real derivative filter even in this case for the same reasoning discussed before; through some *Matlab* function that implement the necessary *casADi* function I obtained the desired control law. the test required for this controller were

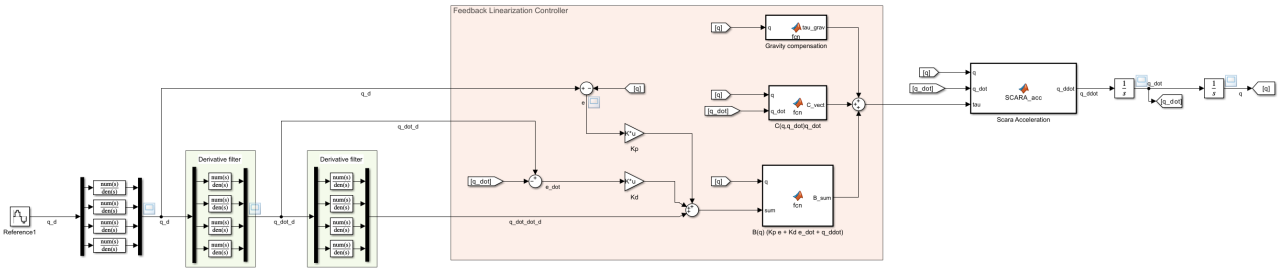

Figure 8: Feedback linearization controller

not only to evaluate its behaviour in tracking a sinusoidal reference at different angular velocity but to test it also with different initial conditions, namely:

$$q_1(0) = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix} \tag{4}$$

$$q_2(0) = \begin{bmatrix} \frac{\pi}{2} & \frac{\pi}{4} & 0 & 0 \end{bmatrix} \tag{5}$$

The results of the test starting from $q_1(0)$ are reported in Figures 9 through 12 instead the results starting from $q_2(0)$ are reported in Figures 13 through 16.

For all the tests I decided to use the basic gains, namely $K_P = K_D = 1000$, and not tune them for each particular application so the results are just indicative of the baseline performances of the controller and can be greatly improved with the right tuning.

Starting from $q_1(0)$ we can see that the behaviour is quite similar to the one we had using the PD with gravity compensation, the errors are the same order of magnitude, and the tracking performances are quite similar; the main difference stands in the computational efforts that is much lower using this controller, indeed it can be used in real time and since it uses, in general, lower gains, so it is much faster that the PD controller but it requires previous knowledge on the system ($B(q)$ and $C(q, \dot{q})$ must be known a priori).

The second set of test has been run using $q_2(0)$ and the results showed the tracking capability of the controller since after a brief transient the system was back at tracking in a very precise way the reference, in this case tuning accurately the proportional and derivative gains could have resulted in a faster transient and so even better tracking capabilities but as said previously I decided to include the baseline performances of the controller. After the transient even in this case, there was a small oscillating error with the same order of magnitude previously observed in the other tests.

An important thing to note about this design is that I still used the gains blocks instead of the PID block, since in this case there was a clear difference between the two schemes, the PID block was much faster in tracking the reference signals, at high angular velocities it took little less than a period to track, but it gave a really high overshoot (over 50%) so for this reason I decided to stick with my architecture that even

though a bit slower it was more predictable and less risky in a real world application.

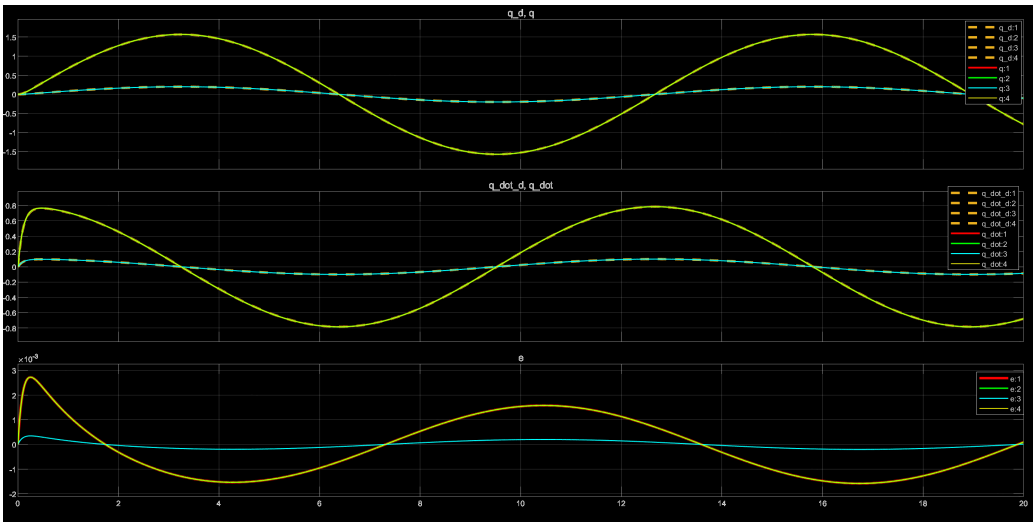

Figure 9: Feedback linearization controller: sinusoidal reference with $\omega = 0.5$rad/s starting from $q_1(0)$

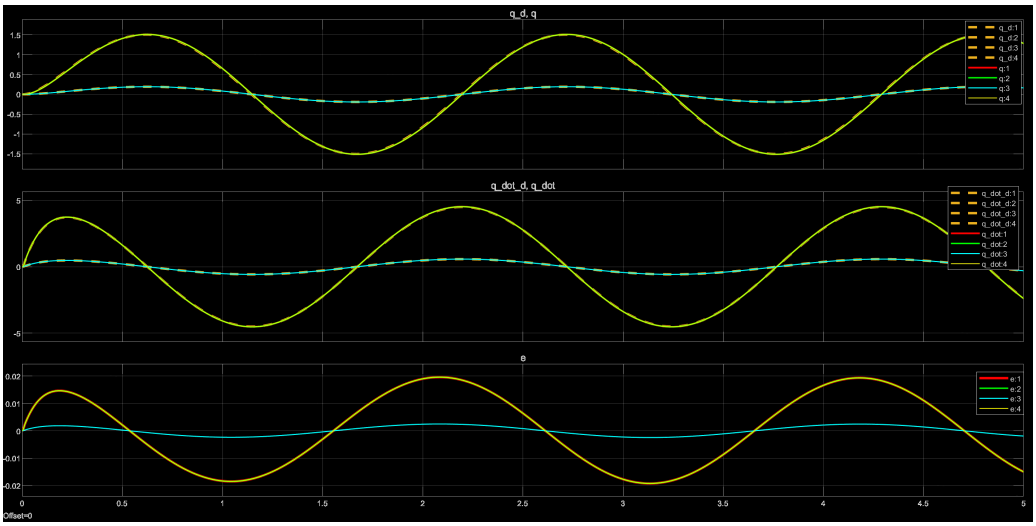

Figure 10: Feedback linearization controller: sinusoidal reference with $\omega = 3$rad/s starting from $q_1(0)$

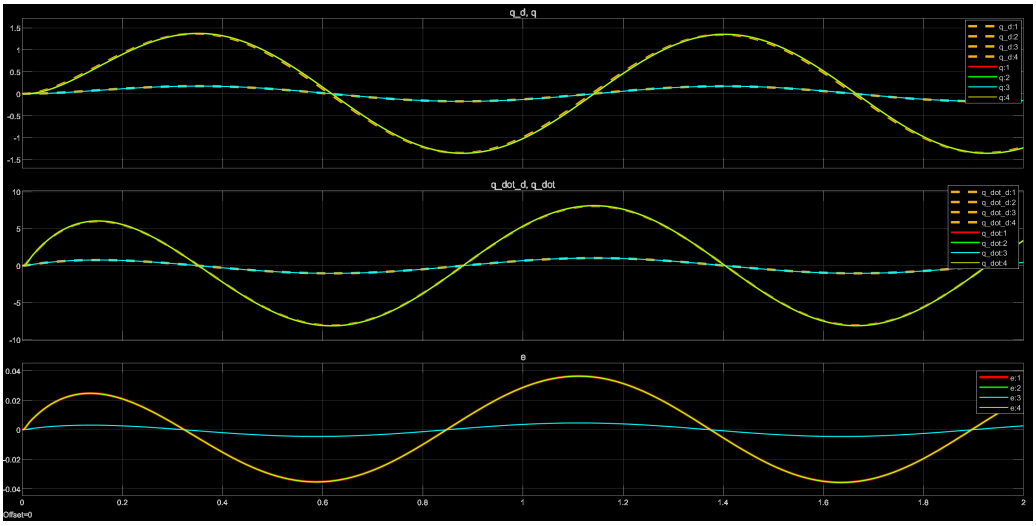

Figure 11: Feedback linearization controller: sinusoidal reference with $\omega = 6$rad/s starting from $q_1(0)$

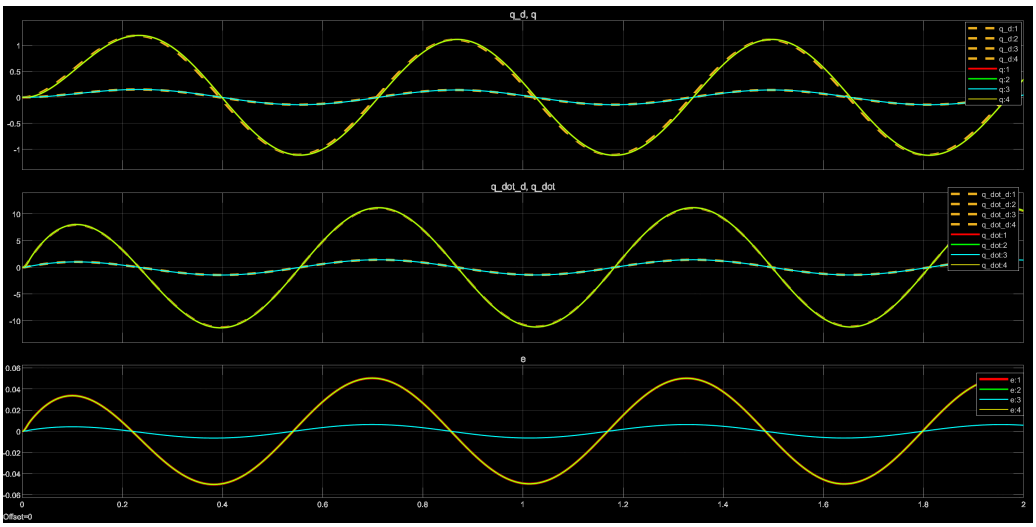

Figure 12: Feedback linearization controller: sinusoidal reference with $\omega = 10$rad/s starting from $q_1(0)$

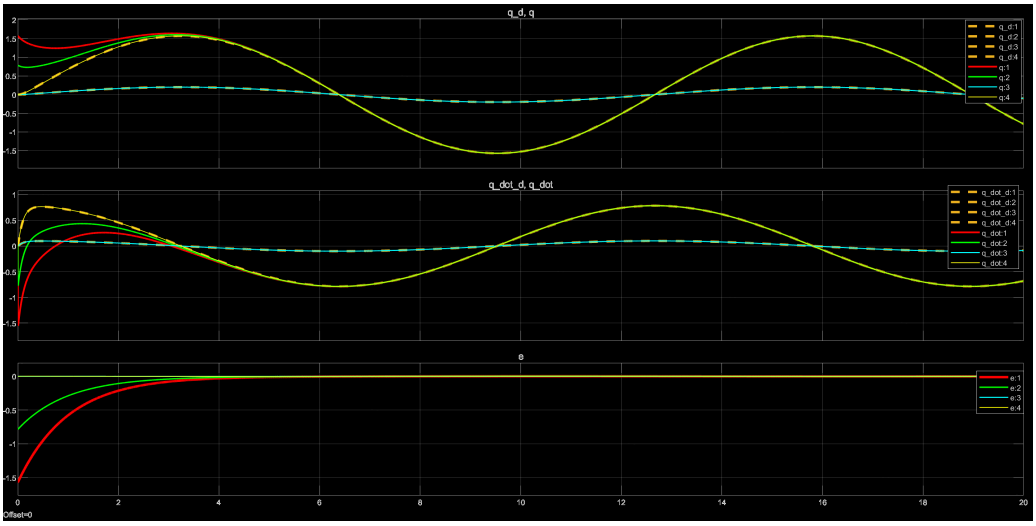

Figure 13: Feedback linearization controller: sinusoidal reference with $\omega = 0.5$rad/s starting from $q_2(0)$

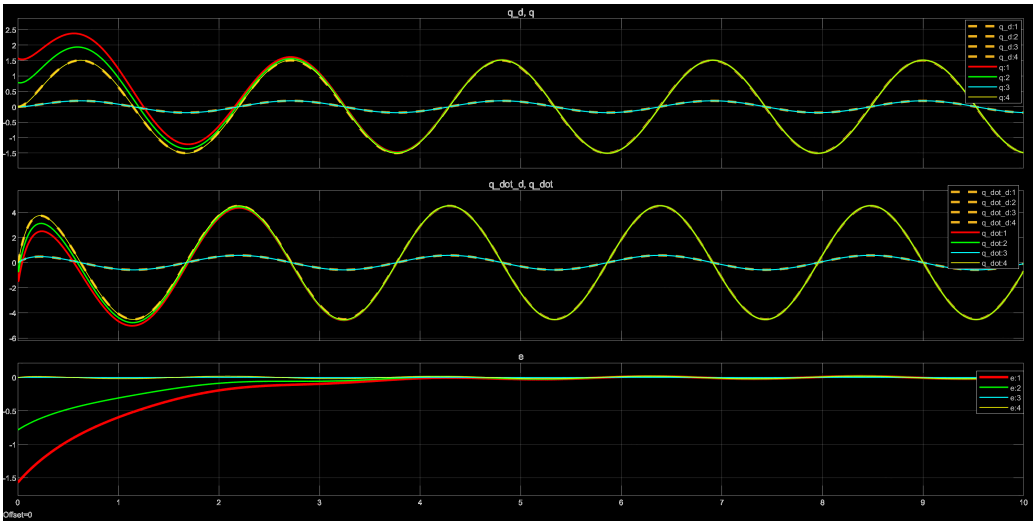

Figure 14: Feedback linearization controller: sinusoidal reference with $\omega = 3$rad/s starting from $q_2(0)$

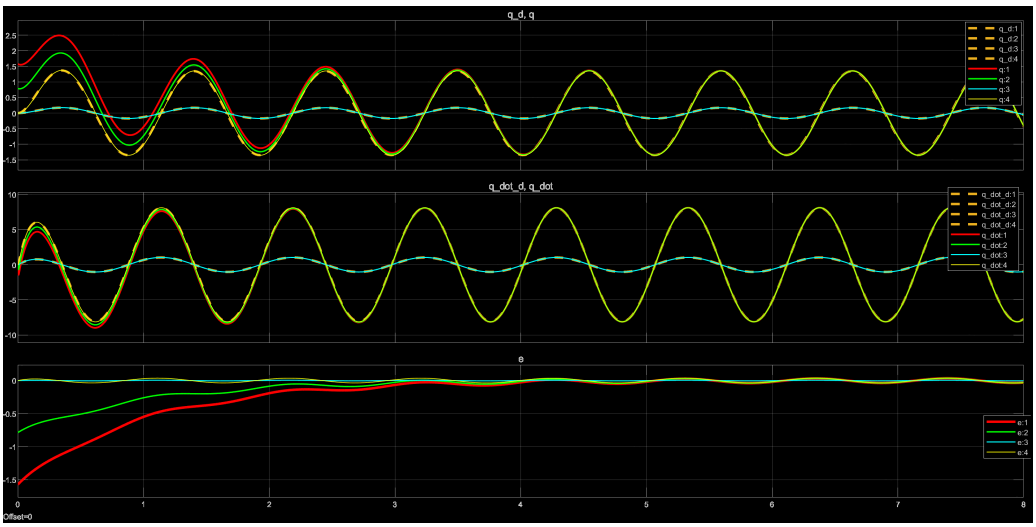

Figure 15: Feedback linearization controller: sinusoidal reference with $\omega = 6$rad/s starting from $q_2(0)$

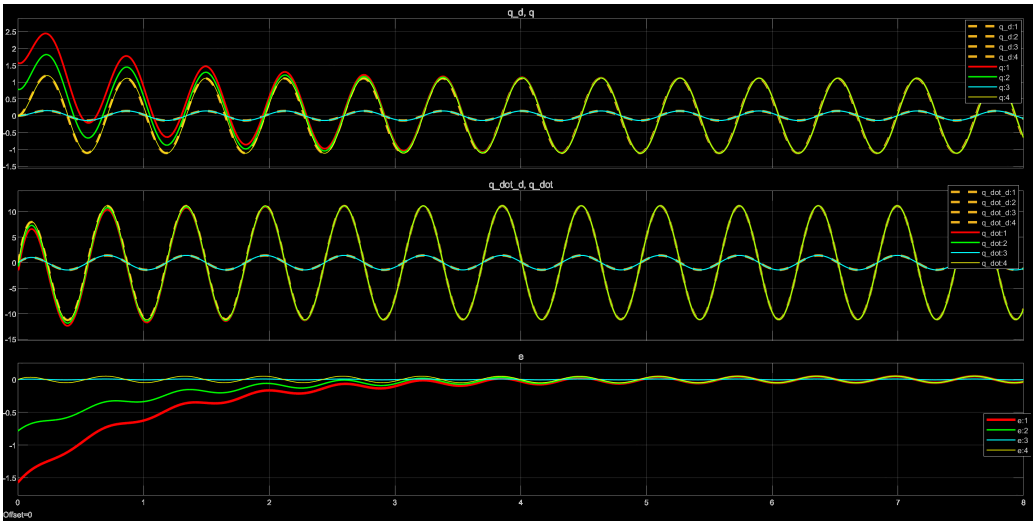

Figure 16: Feedback linearization controller: sinusoidal reference with $\omega = 10$rad/s starting from $q_2(0)$

2 Derivation of the UR10 kinematics

To complete this second task most of the procedure was the same as the one explained in the numerical experiences, so I will not repeat the common steps, the main differences were related to the absence of link 0 in the UR10, but apart from this everything was the same.
There were three task to test the kinematics and the dynamics equation obtained. In the following I will briefly explain how I obtained the required *.mat* files.

- **POS**: I took the first 3 elements of f_x calculated for the desired q and iterated through all q in Q_kin_test;
- **ANG**: I took the last three elements (corresponding to last link) of the rotation matrix calculated for each q given and then through the apposite function converted the values to yaw, pith and roll angles;
- **TAU**: after implementing the various elements of the necessary matrices, B, C (using f_C_vect) and g, I calculated τ as required for each element of $**\_dyn\_test$ files provided.

As said at the beginning all the results and the code needed to run everything is included in the *.zip* folder provided, inside the folder named: "2"; moreover the file called named **Results.mat** contains the 3 variables required (if *UR10.mlx* is run again the results file will be overwritten).