Homework 3 Robotics and Control 1

Christian Francesco Russo 2087922

1 Control of the SCARA robot

1.1 PD with gravity compensation: Target position

For the SCARA robot, implement, simulate, and test a PD controller with gravity compensation for the target position task.

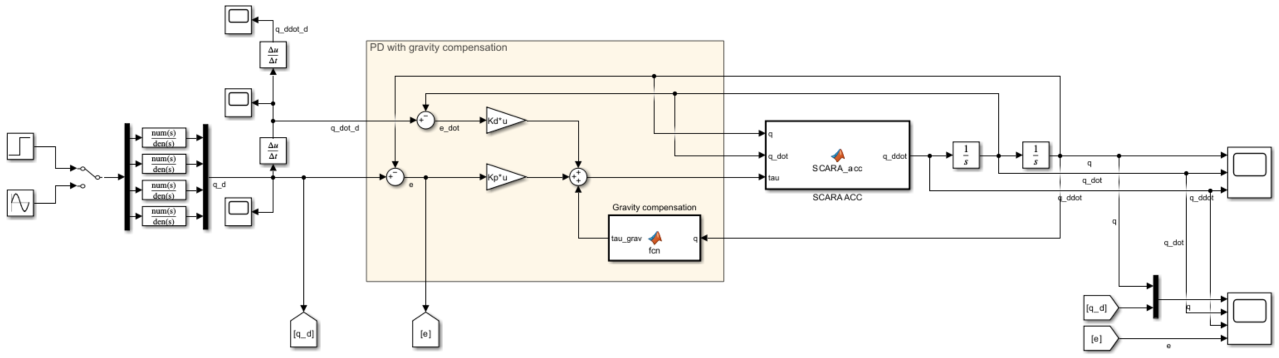
PD gravity compensation, namely:

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

where $\boldsymbol{e} = \boldsymbol{q}_d - \boldsymbol{q}$ and $\boldsymbol{g}(\boldsymbol{q})$ are respectively the tracking error and the vector of gravitational forces. Set the gains of the PD controller to $K_p = 1000\mathbb{I}$ and $K_d = 1000\mathbb{I}$.

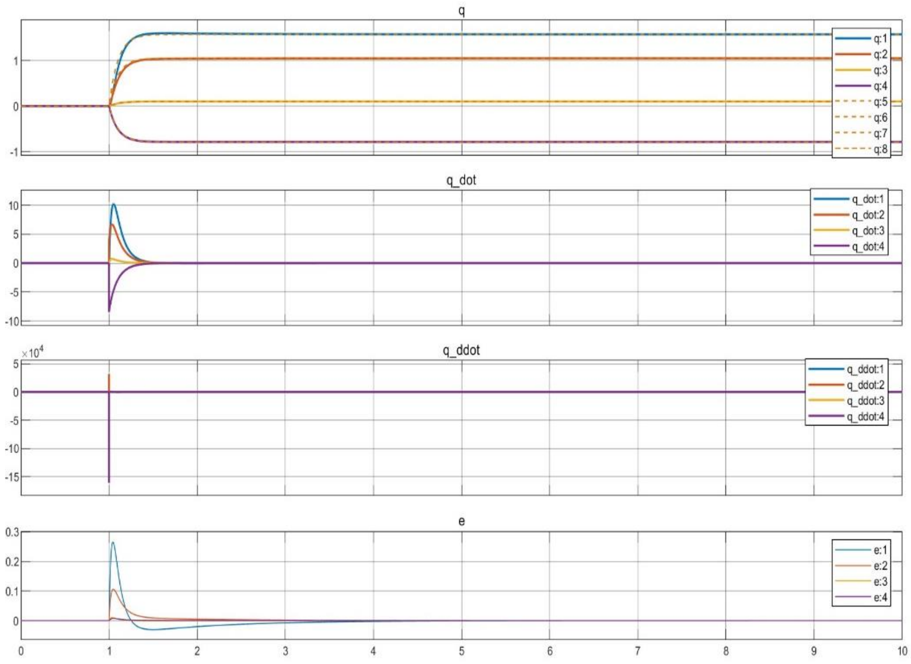
Code: folder `Exercise_1.1`.

Simulink control schema:



Results:

Initial condition $\boldsymbol{q} = [0, 0, 0, 0] \rightarrow$ target position $\boldsymbol{q} = [\pi/2\,,\ \pi/3\,,\ 0.1,\ -\pi/4]$:



The PD gravity compensation control successfully allows the robot to reach the desired position and to remain steady in the desired configuration (i.e., $\dot{\boldsymbol{q}} = \mathbf{0}$ after reaching the desired configuration). Besides, the error is quite low at the beginning, and quickly converges to 0.

1.2 PD with gravity compensation: Trajectory tracking

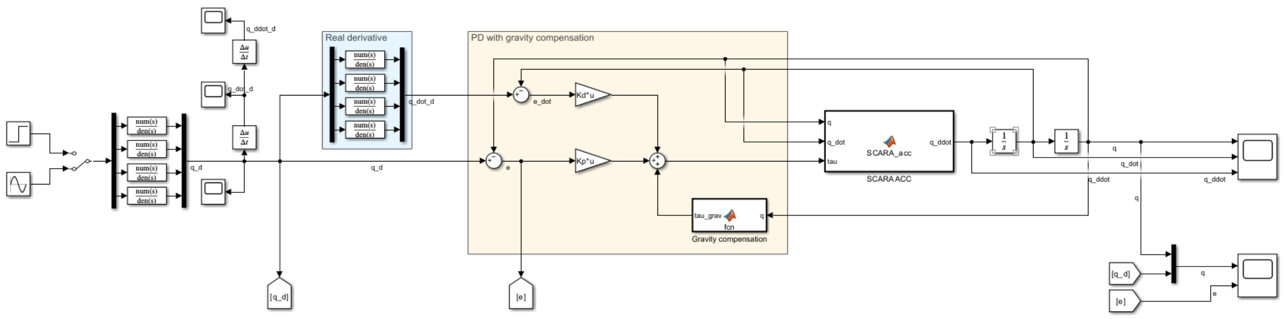
For the SCARA robot, implement, simulate, and test a PD controller with gravity compensation for the trajectory tracking task.
PD gravity compensation, namely:

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

where $\boldsymbol{e} = \boldsymbol{q}_d - \boldsymbol{q}$ and $\boldsymbol{g}(\boldsymbol{q})$ are respectively the tracking error and the vector of gravitational forces. Set the gains of the PD controller to $K_p = 1000\mathbb{I}$ and $K_d = 1000\mathbb{I}$.

Code: folder `Exercise_1.2`.

Simulink control schema:



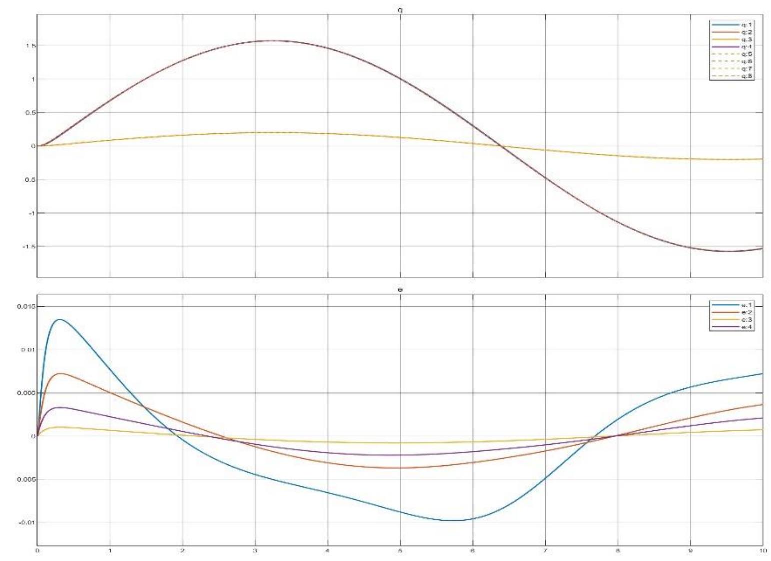
Note: this schema is similar to the previous one, with the difference that the discrete derivative was replaced with a real derivative having transfer function:

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

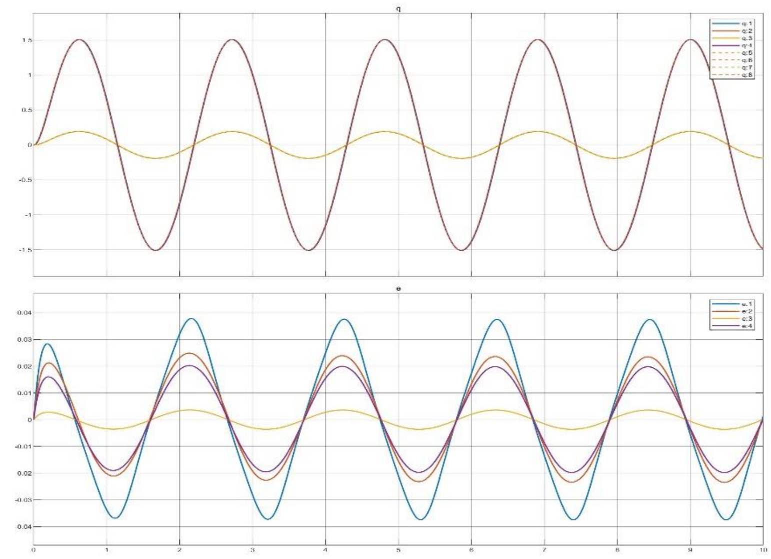
where $\delta = 1/\sqrt{2}$ and $\omega_c = 2\pi 50 \text{ rad}/s$. This change allows the simulation to run faster, even if still not in real-time, and does not require the generation of a sampled trajectory using casADi.

Results:
Comparison of the results considering an input reference sinusoid with $\omega = 0.5, 3, 6, 10$, with relative set gains for K_p and for K_d:

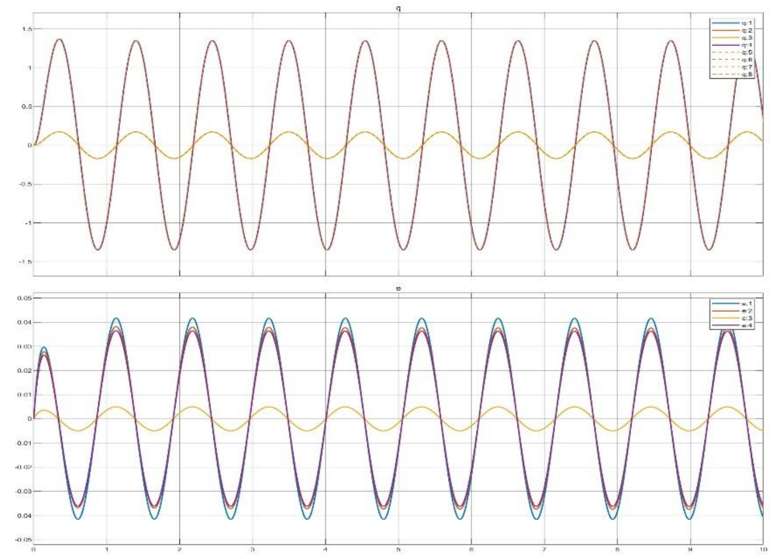
Test 1: $\omega = 0.5\ [\text{rad}/s],\ \ K_p = 1000\mathbb{I},\ \ K_d = 2000\mathbb{I}$



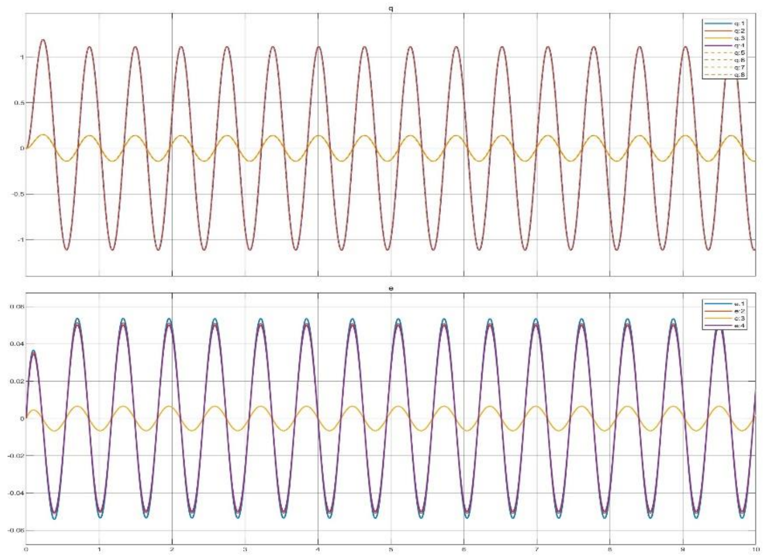
Test 2: $\omega = 3$ [rad/s], $K_p = 4000\mathbb{I}$, $K_d = 8000\mathbb{I}$



Test 3: $\omega = 6$ [rad/s], $K_p = 16000\mathbb{I}$, $K_d = 48000\mathbb{I}$



Test 4: $\omega = 10\ [\mathrm{rad}/s],\ \ K_p = 64000\mathbb{I},\ \ K_d = 96000\mathbb{I}$



From these tests we can derive the following results:

- Higher frequencies require higher gains, for both the proportional and derivative controllers, to keep the errors e_i of each link below an acceptable threshold.
- The error e_i of each link never converges, instead, it keeps oscillating indefinitely within a fixed range.
- At higher frequencies, it becomes increasingly difficult to bound the error, necessitating impractically high gains.
- The computation time is significantly slowed down by the magnitude of the gain K_d, in particular, higher K_d values result in slower simulations.

1.3 Feedback linearization

For the SCARA robot, implement, simulate, and test a feedback linearization controller for the trajectory tracking task.

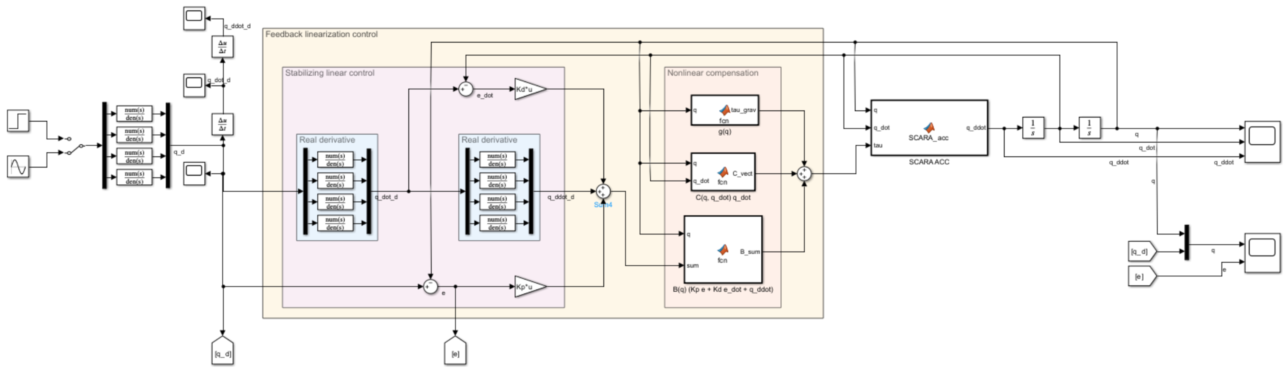
Feedback linearization controller, namely:

$$\boldsymbol{\tau} = B(\boldsymbol{q})\left(K_p\boldsymbol{e} + K_d\dot{\boldsymbol{e}} + \ddot{\boldsymbol{q}}_d\right) + \boldsymbol{g}(\boldsymbol{q}) + C(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$$

where $C(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$ corresponds to the vector of forces due to fictitious contributions.

Code: folder `Exercise_1.3`.

Simulink control schema:

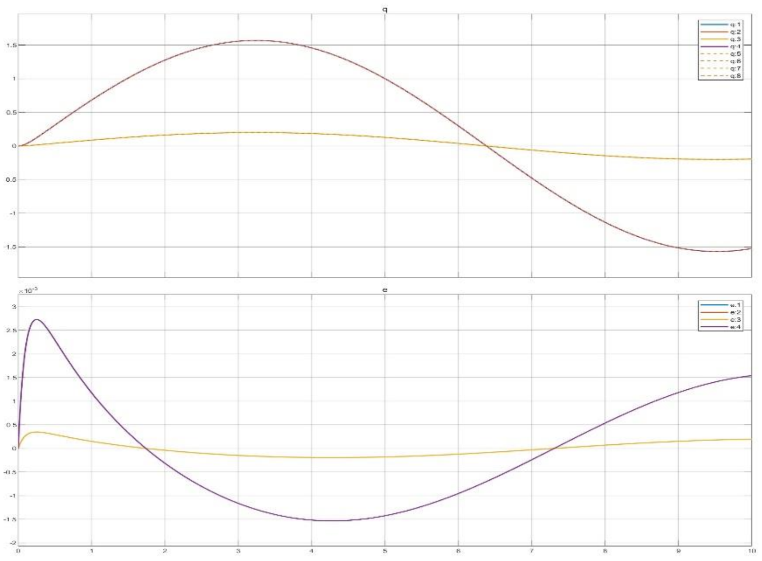


Note: the matrices $B(\boldsymbol{q})$, $\boldsymbol{g}(\boldsymbol{q})$, $C(\boldsymbol{q}, \dot{\boldsymbol{q}})$ were provided.
In this control schema, analogously as done before, we used the real derivative to get $\dot{q}_d$, and then another real derivative was added in cascade to it in order to obtain also $\ddot{q}_d$.

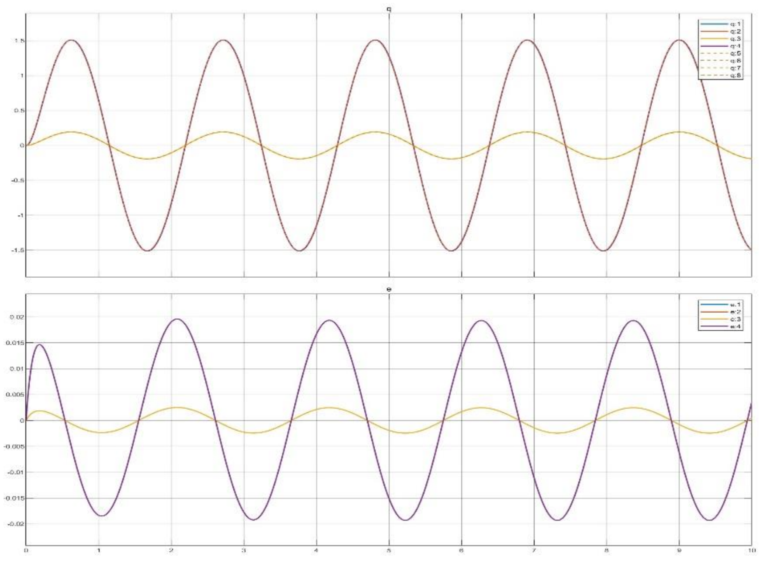
Results:
Comparison of the results considering an input reference sinusoid with $\omega = 0.5, 3, 6, 10$.

- Case initial value $\boldsymbol{q}(0) = [0, 0, 0, 0]$:

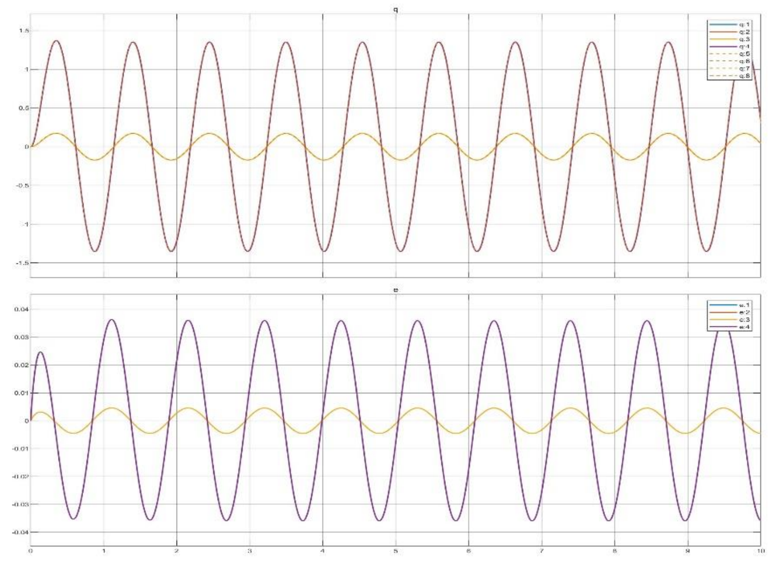
Test 1: $\omega = 0.5$ [rad/s]



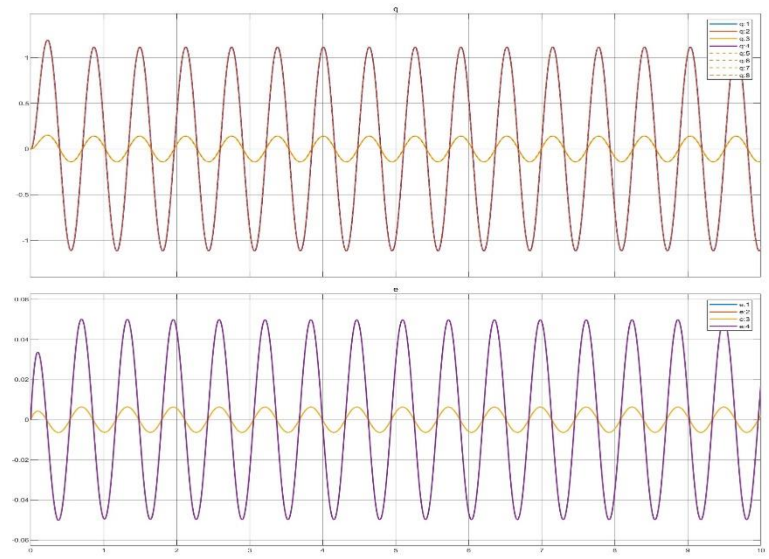
Test 2: $\omega = 3$ [rad/s]



Test 3: $\omega = 6$ [rad/s]

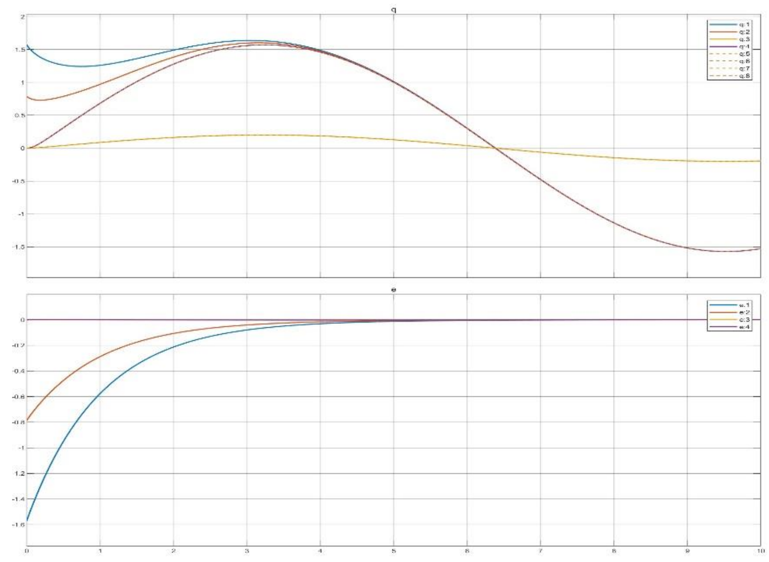


Test 4: $\omega = 10$ [rad/s]

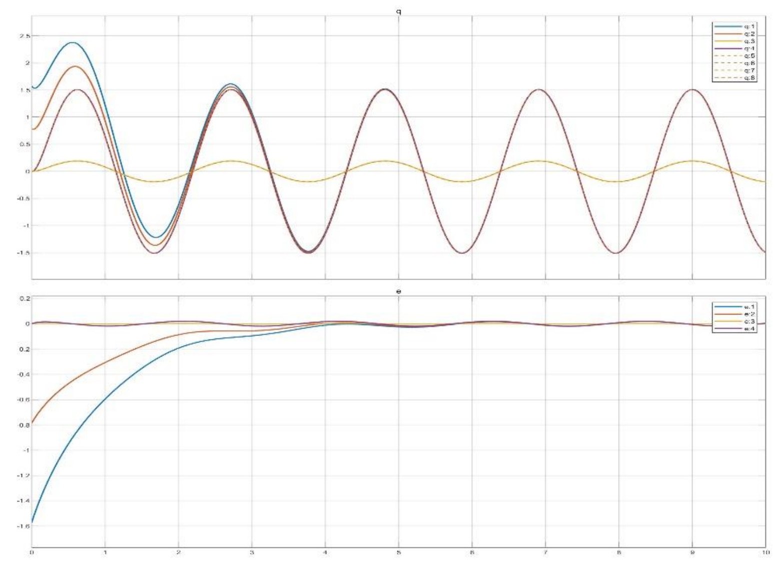


- Case initial value $\boldsymbol{q}(0) = [\pi/2\,, \pi/4\,, 0, 0]$:

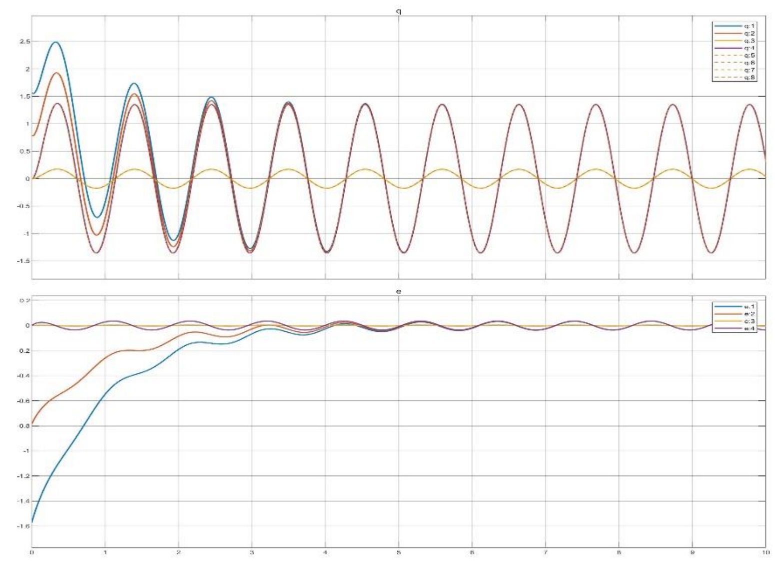
Test 1: $\omega = 0.5$ [rad/s]



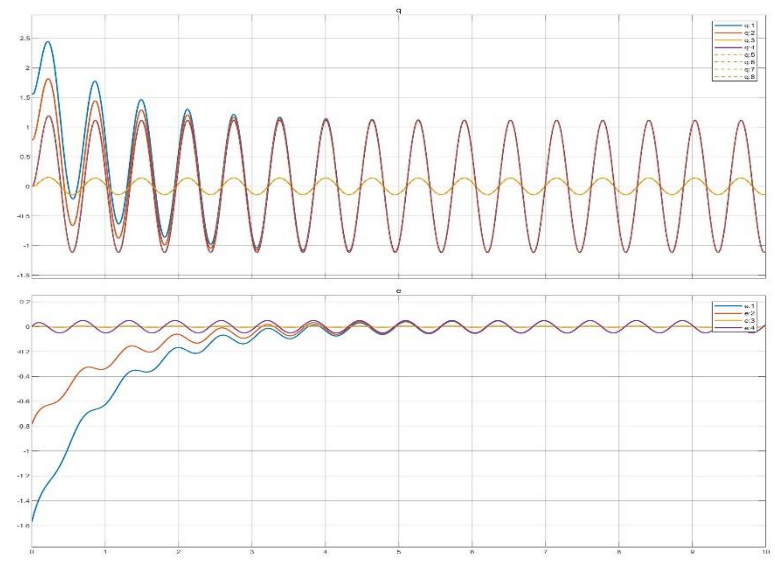
Test 2: $\omega = 3\ [\mathrm{rad}/s]$



Test 3: $\omega = 6\ [\mathrm{rad}/s]$



Test 4: $\omega = 10\ [\mathrm{rad}/s]$



Note: for all these tests we have used the default gains $K_p = 1000\mathbb{I}$ and $K_d = 1000\mathbb{I}$.

From these tests we can derive the following results:

- Even with lower gain values, the feedback linearization control performs better than the standard PD control with gravity compensation.
- Much better results could be achieved with higher gains.
- The fact that this controller works well even with lower gains results in much faster computation and good trajectory tracking.
- For the initial value $\boldsymbol{q}(0) = [0, 0, 0, 0]$, where the robot starts from the desired trajectory, the error keeps oscillating, as for the previous controller, but it remains limited.
- For the initial value $\boldsymbol{q}(0) = [\pi/2\,, \pi/4\,, 0, 0]$, where the robot does not start from the desired trajectory, obviously the error is high initially, but it converges quite quickly towards the desired trajectory, then again it keeps oscillating within a small range.
- Similarly to the previous controller, the error grows with the frequency ω.

2 Derivation of the UR10 kinematics and dynamics using casADi

For the UR10 derive the kinematics and the dynamics, and write a Matlab file which generates the functions computing the following quantities:

- The position of the last DH reference frame w.r.t. the base frame;
- The orientation of the last DH reference frame w.r.t. the base frame, expressed in yaw, pitch and roll angles;
- The expression of $B(\boldsymbol{q})$, $\boldsymbol{g}(\boldsymbol{q})$, and $C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$;

In particular provide a .mat file containing the following variables: POS, ANG, TAU.

Code: folder `Exercise_2`, file `UR10.mlx`.

Variables: POS, ANG, TAU stored in the .mat file `results.mat`.