Università di Genova   Rivi Alice 5135011

Robot Dynamics & Control – A.A. 2021/2022

Assignment 3: Dynamic Robot Control

The dynamics of a manipulator is described by a set of non-linear second-order differential equations, where the coupling terms represent the gravitational torques, which depend on the coordinates of the joints, the reaction torques, due to the acceleration of the joints, and the Coriolis and centrifugal torques.

The joint variables are related to generalised torques, so the control schemes have two levels.
The first level requires coordinate transformations to convert the description of a desired path from task-oriented space, usually Cartesian space, to joint space, also referred to as oriented space.
The second level calculates the generalised torques, then the actuator inputs, using the dynamic model of the robotic arm.

Dynamic control is very sensitive to model inaccuracies. These inaccuracies are mainly due to errors related to the lengths of links, their masses and inertias, the elasticities of gear trains, backlash and variable payloads.

The amount of calculations required at the first and second levels sets an upper limit for the sampling rate to be used. A finite sampling rate and inaccuracies in the model of the robotic manipulator led to inevitable errors in positioning and especially in target tracking.
To overcome these difficulties at the first level, a number of task-oriented control schemes have been developed, while robust control techniques for uncertain implants are used for the type of problems that arise at the second level.

The mathematical dynamic model can be written in the form:

$$M(q)q\_2dot + C(q,q\_dot)q\_dot + G(q) = \tau + \tau^{ext}$$

where M(q) is the generalized inertial matrix, C(q,q_dot) is the matrix that take into account the Coriolis and centrifugal effect and G(q) is the vector of the gravitational torque. On the other hand, τ is the vector of the control action and τ^{ext} described the torques due to the disturbances and noise.

This formula can be represented through a block diagram:

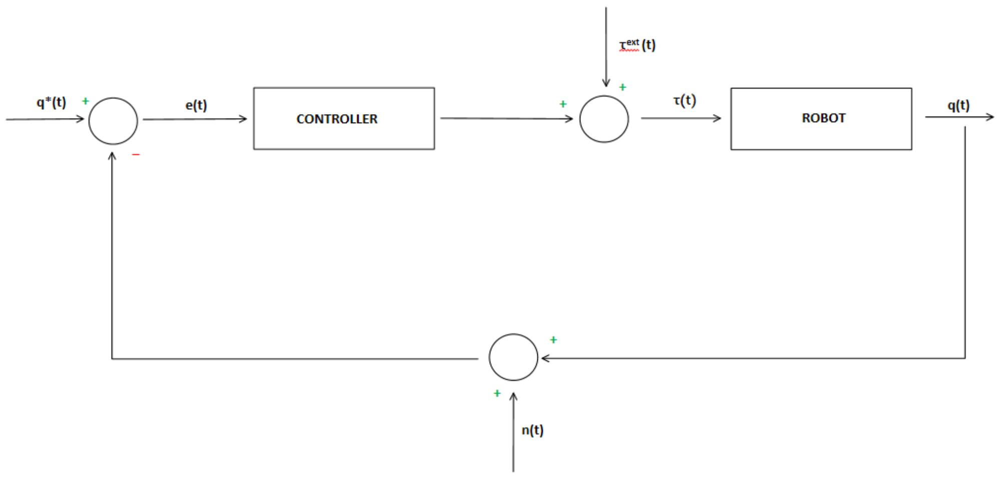

Figure 1 - Block diagram

where n(t) is the contribution taking into account white noise and e(t) = q*(t) – q(t) represents the errors.

In this assignment, we evaluated the control dynamics of a UR5 manipulator. The UR5 robotic arm is a collaborative robot, consisting of 6 rotoidal joints that give it 6 degrees of freedom. This type of manipulator is perfect for automating repetitive and hazardous collaborative tasks with low weight.
Through a simulation model of the UR5, built using Simulink, we can check how the robot's configuration varies by providing torque commands to the joints and we can read the position, speed and acceleration of the joints.

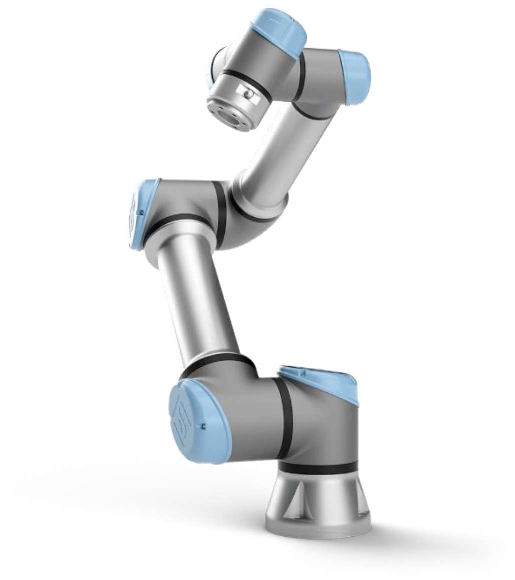
Figure 2 - UR5

In order to work on this model, certain packages and toolboxes released by Matlab and Simulink are required:
- Robotics System Toolbox™
- Simscape Multibody™
- Robot Library Data Support Package

Robotics System Toolbox™ provides tools and algorithms to design, simulate, test and deploy manipulator applications. This toolbox includes algorithms for manipulator collision control, path planning, trajectory generation, direct and inverse kinematics and dynamics using a rigid-body tree representation.
It also allows you to build test scenarios and use the provided reference examples to validate common industrial robotics applications. It also includes a library of commercially available industrial robot models that you can import, visualise, simulate and use with reference applications.

Simscape Multibody™ provides a multibody simulation environment for 3D mechanical systems such as robots. Through this package, multibody systems can be modelled using blocks representing bodies, joints, constraints, force elements and sensors. Simscape Multibody™ formulates and solves the equations of motion for the complete mechanical system. CAD geometries including masses, inertias, joints, constraints and 3D geometries can be imported into your model. An automatically generated 3D animation allows you to visualise the dynamics of the system.

Robot Library Data Support Package provides source mesh files to visualise and simulate Robot Library robots on platforms other than MATLAB.

Exercise 1 – Gravity Compensation

In this exercise, the force of gravity acting along the Z-axis must be compensated to prevent the robot from falling to the ground.

This is possible by adding the Gravity Torque block provided by the Robotics System Toolbox™.
This block takes the robot's configuration q as input and returns the joint torques τ that balance gravity.

The initial joint configuration that must be maintained is:

q0 = (0 -1.570796e+00 7.853982e-01 0 0 0) [rad/s]

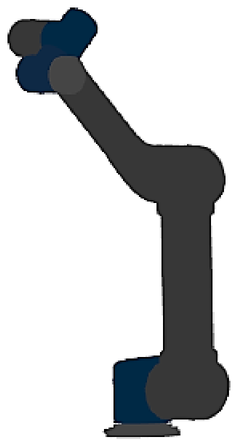

Figure 3 - Intial configuration q0

This is possible thanks to the gravitational compensation equal to the torque τ:

τ = (-9.6556e-16 -1.1213e+01 -1.1213e+01 -1.2336e-01 0 0) [Nm]

The block diagram of the system turns out to be:

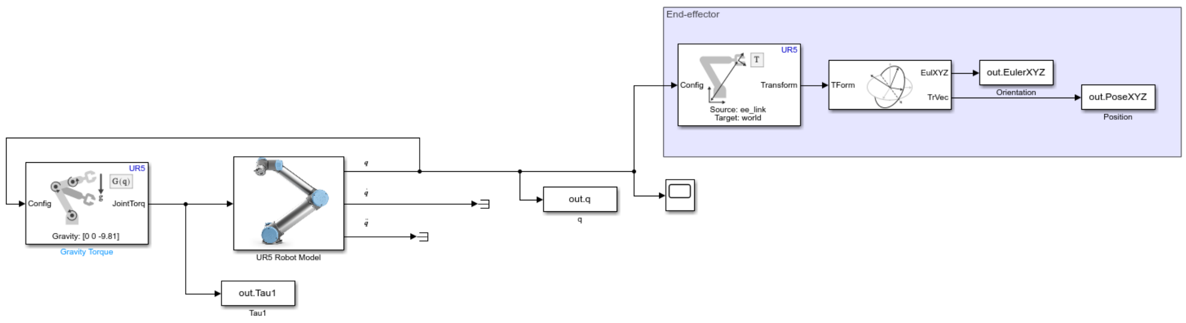

Figure 4 - Simulink block diagram

The part within the end-effector area will be used for calculations in subsequent exercises.
Thanks to the Get Transform block , we can calculate the configuration of the end-effector with respect to

the absolute reference system, while through the Coordinate Transformation Conversion block we can convert the coordinate transformation to obtain the orientation and the pose of the end-effector.

The graph showing the trend of the manipulator configuration is equal to:

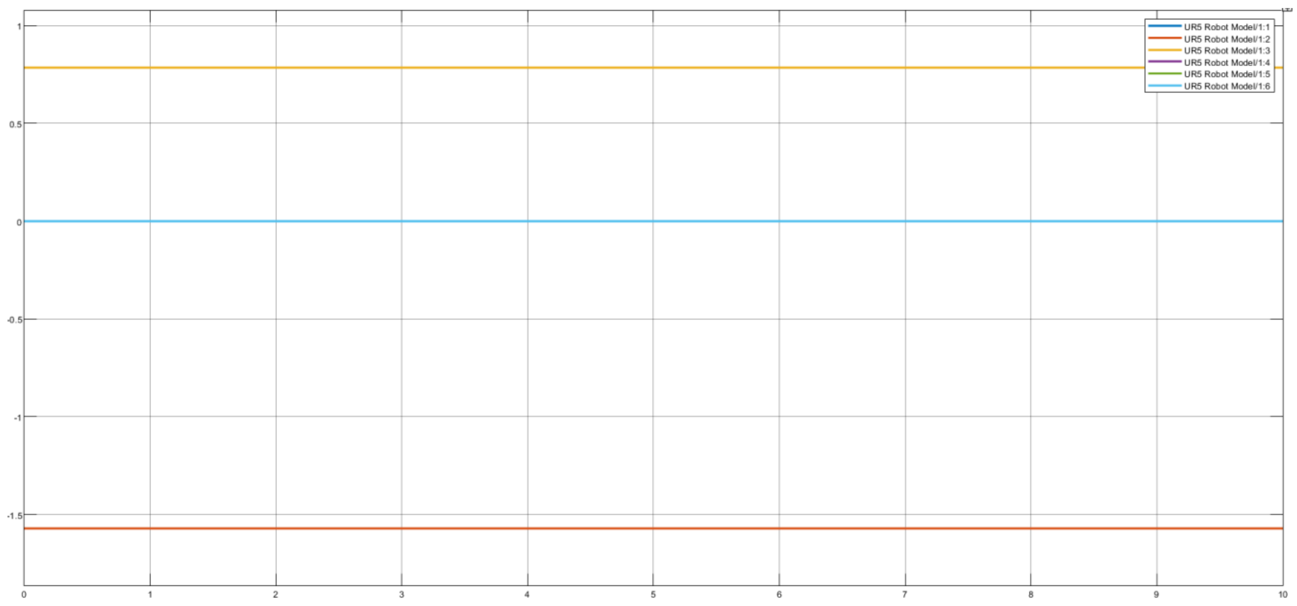

Figure 5 - Robot configuration trend

Exercise 2 – Linear Joint Control

In the second exercise, we have to provide torque commands in order to reach the desired joint configuration q*. q* is equal to the sum of q0 and Δq, where q0 is the initial joint configuration, while $\Delta q = [\pi/4, -\pi/6, \pi/4, \pi/3, -\pi/2, \pi]$.

In this exercise we must consider the gravity compensated PD control action in the joint space:

$$\boldsymbol{\tau = -K_v \delta q\_dot(t) - K_e \delta q(t) + G(q)}$$

with $\delta q = q^* - q$, $K_v = \mathrm{diag}(K_{v,i})$ and $K_e = \mathrm{diag}(K_{e,i})$ positive diagonal matrices defined.

The initial configuration is still the same q0, while the desired configuration $q^* = q0 + \Delta q$ is:

q* = (7.8539e-01 -2.0943 1.5707 1.0471 -1.5707 3.1415) [rad/s]

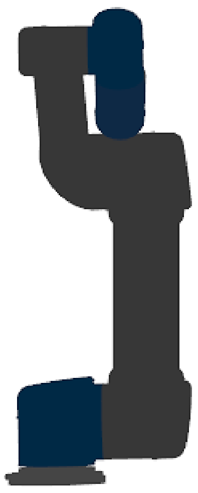

*Figure 6 - Desired joint configuration q**

The robot reaches this configuration thanks to the torque τ. The first value of τ is equal to:

τ = (7.8539 -1.6449e+01 -3.3595 1.0348e+01 -1.5707e+01 3.1415e+01) [Nm]

It then varies to balance the manipulator and allow it to achieve the desired configuration.

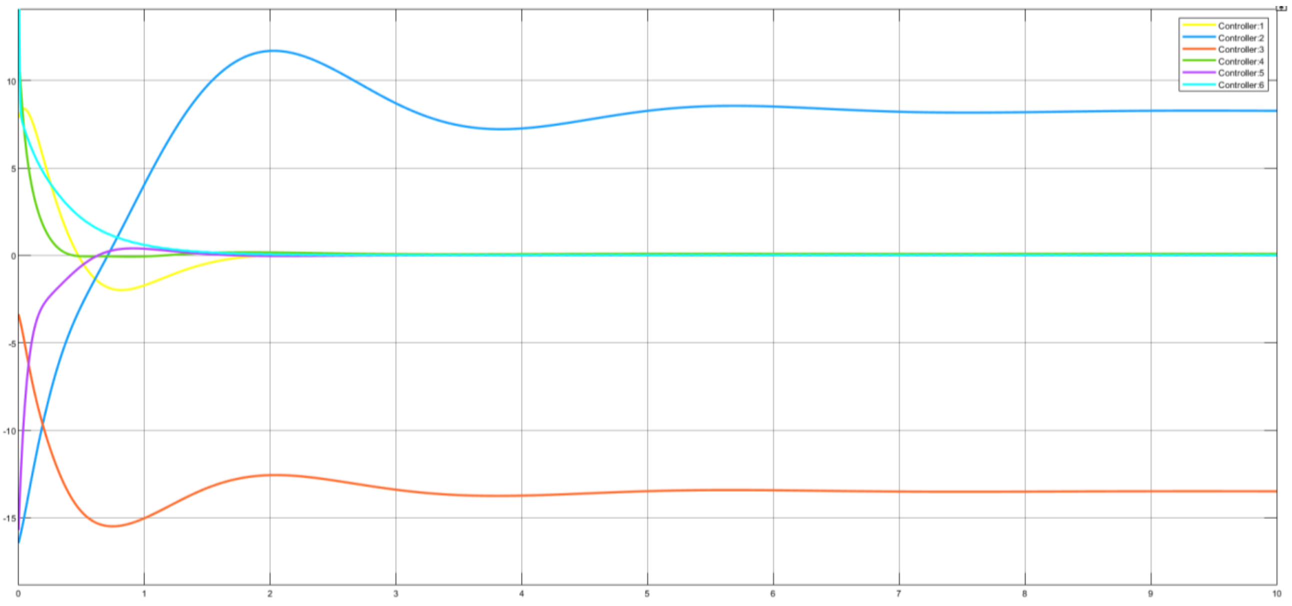

Figure 7 - Trend of torques acting on the robot

The block diagram of the system turns out to be:

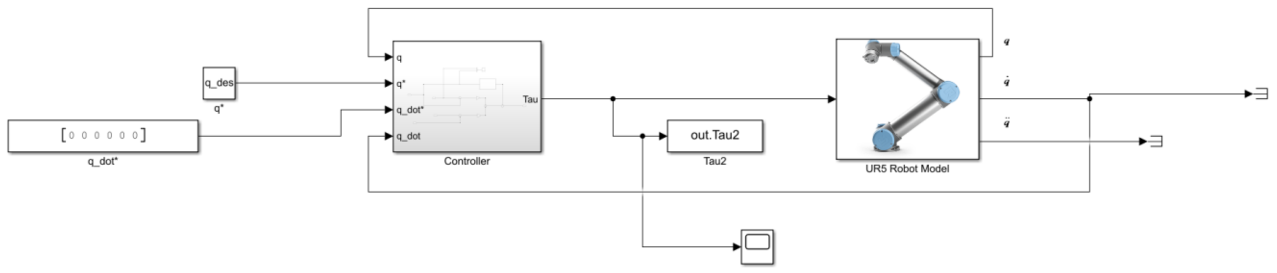

Figure 8 - Simulink block diagram

Inside the Controller block are all the components for calculating the torque τ to be applied to the manipulator:

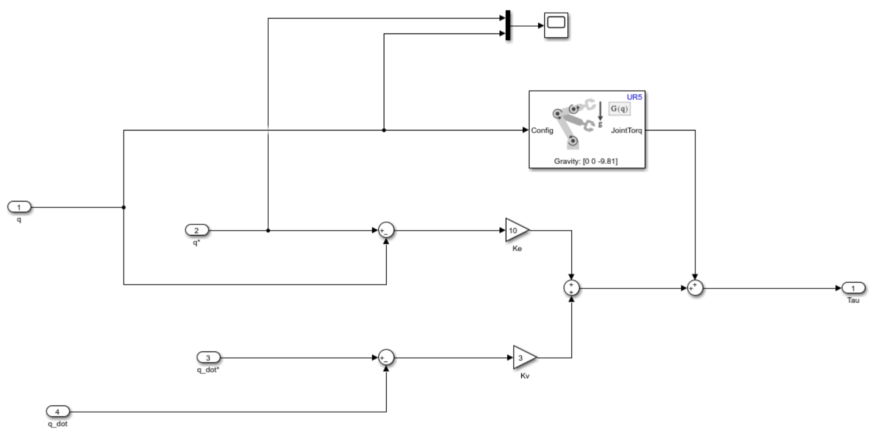

Figure 9 - Simulink block diagram

$K_v = 3$
$K_e = 10$

Through the calculation of errors, we can verify how the desired configuration q* differs from the real one q:

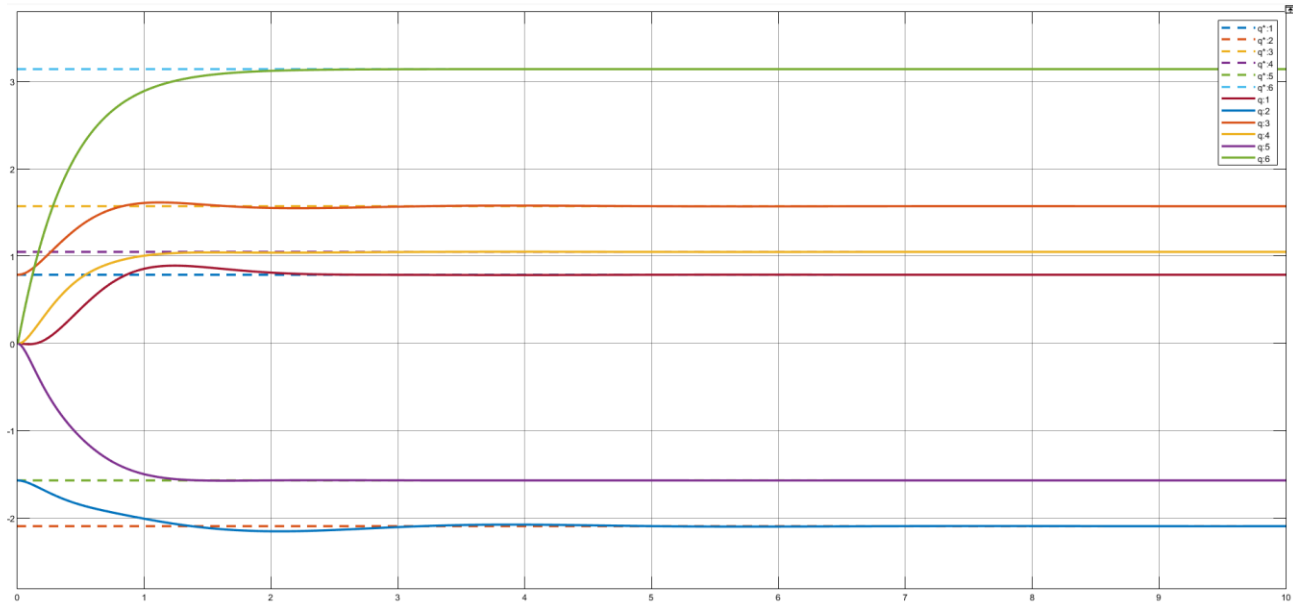

Figure 10 - Comparison of trend of the desired configuration q and the actual configuration q of the robot*

If gravity compensation is not considered, the term G(q) becomes null, so τ results:

$$\tau = -K_v \delta q\_dot(t) - K_e \delta q(t)$$

The first value of τ is equal to:

τ = (7.8539 -5.2359 7.8539 1.0471e+01 -1.5707e+01 3.1415e+01) [Nm]

It then varies in the following way:

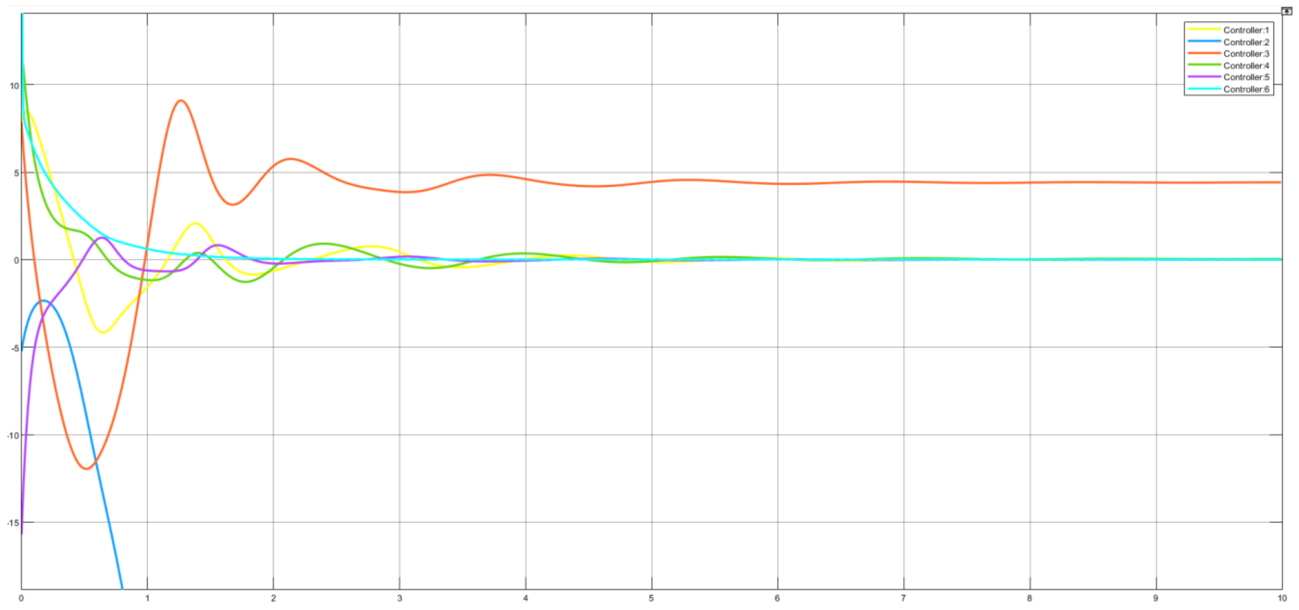

Figure 11 - Trend of torques acting on the robot without gravity compensation

The robot does not achieve the desired configuration, but assumes the following one:

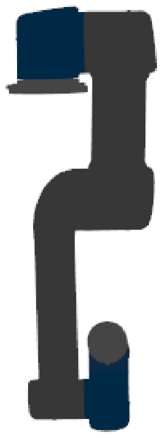

Figure 12 - Desired joint configuration q without gravity compensation*

Exercise 3 – Linear Cartesian Control

In the third exercise, we must provide torque commands in order to reach the desired cartesian configuration x*. Also, in this case x* is equal to the sum of x0 and Δx, x0 in the initial cartesian pose of the end effector and $\Delta x = [0.2, -0.08, -0.15, \pi/4, -\pi/4, \pi/2]$.

The reasoning for executing this exercise is the same as the previous one. The main difference is that, before we were working in joint space and we wanted the robot to achieve a certain joint configuration, now we are in Cartesian space and our goal is for the end-effector to reach the desired cartesian configuration.

$$\tau = -K_v \delta x\_dot(t) - K_e \delta x(t) + G(q)$$

with $\delta x = x^* - x$, $K_v = \mathrm{diag}(K_{v,i})$ and $K_e = \mathrm{diag}(K_{e,i})$ positive diagonal matrices defined.

The initial Cartesian configuration x0 is obtained by taking the initial pose and initial orientation of the end-effector from the first exercise:

x0 = (3.4429e-01 1.9145e-01 7.2459e-01 [m]

-3.1415 7.8539e-01 -1.5707) [rad/s]

In this way, the desired Cartesian configuration x* results:

x* = (5.4429e-01 1.1145e-01 5.7459e-01 [m]

-2.3561 9.7998e-12 -4.8978e-12) [rad/s]

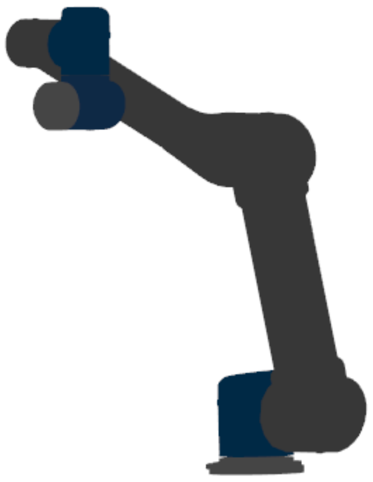

*Figure 13 - Desired Cartesian configuration x**

The robot reaches this position thanks to the torque τ. The last value of τ is equal to:

τ = (3.1549e-03 -2.5333e+01 -1.4164e+01 -1.3001e-03 -2.5775e-04 1.4816e-03) [Nm]

In fact, the torque follows the following trend:

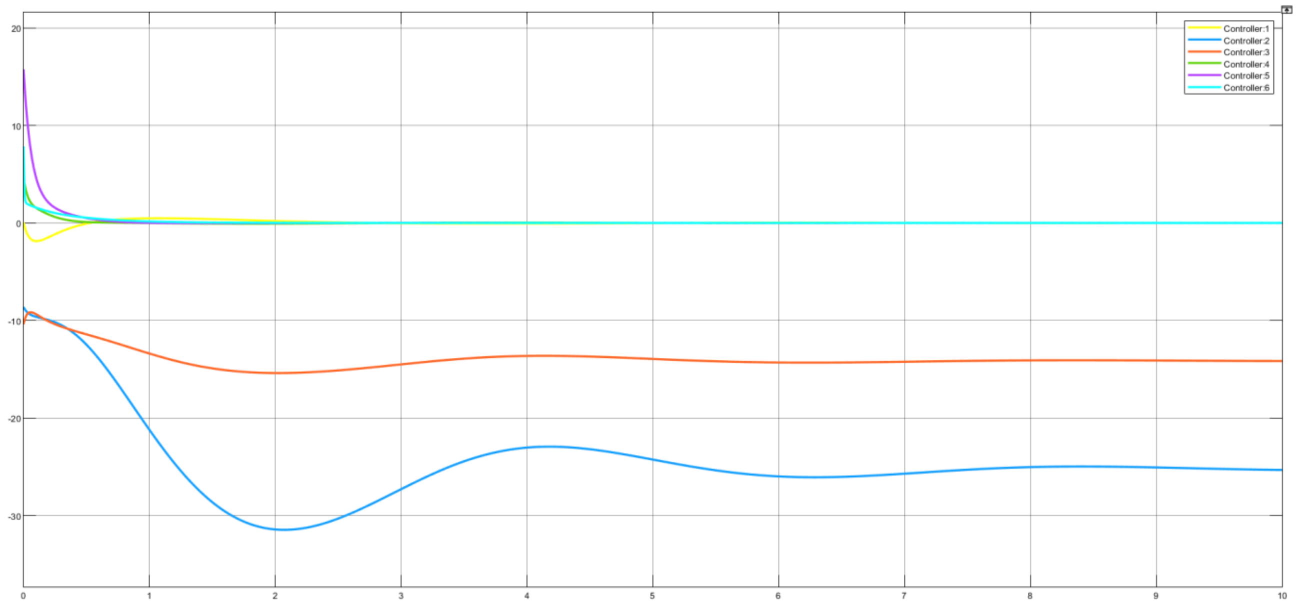

Figure 14 - Trend of torques acting on the robot

The block diagram of the system turns out to be:

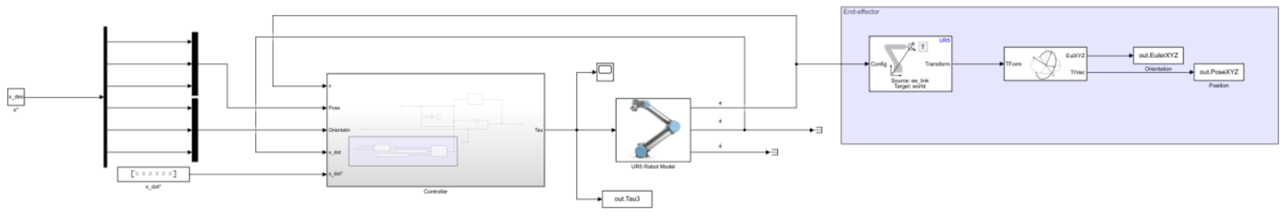

Figure 15 - Simulink block diagram

Inside the Controller block are all the components for calculating the torque τ to be applied to the manipulator:

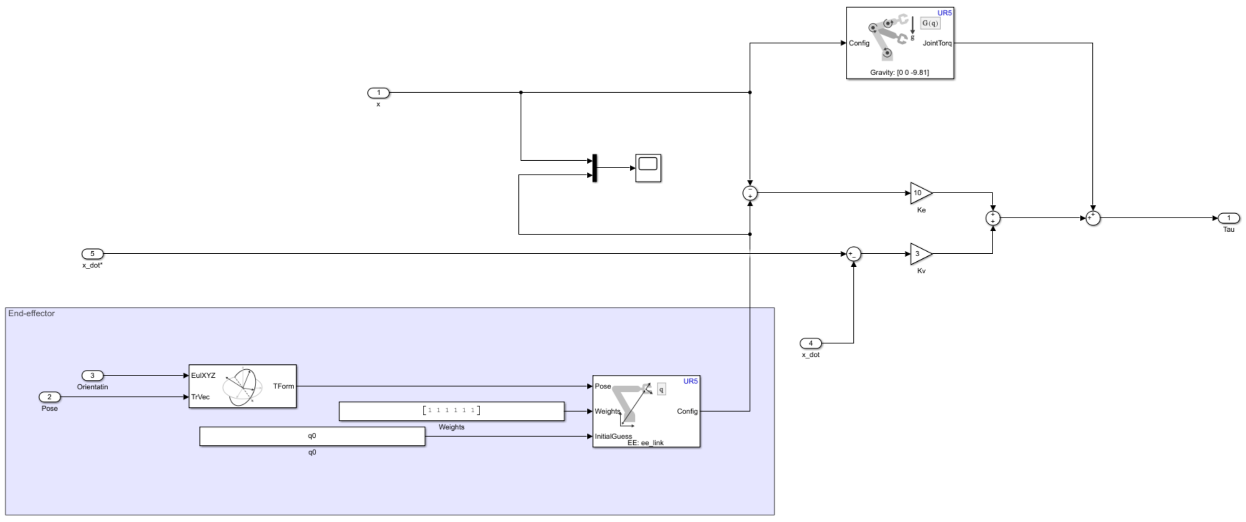

Figure 16 - Simulink block diagram

$K_v = 3$
$K_e = 10$

The areas relative to the end-effector make it possible to first calculate its inverse kinematic (inside the Controller block) and then its direct kinematic. Also in this case thanks to the Get Transform block , we can calculate the configuration of the end-effector with respect to the absolute reference system, while through the Coordinate Transformation Conversion block we can first convert the orientation and pose of the end-effector to obtain its position, then convert the coordinate transformation to obtain its orientation and pose.

Through the calculation of errors, we can verify how the desired configuration differs from the real one:

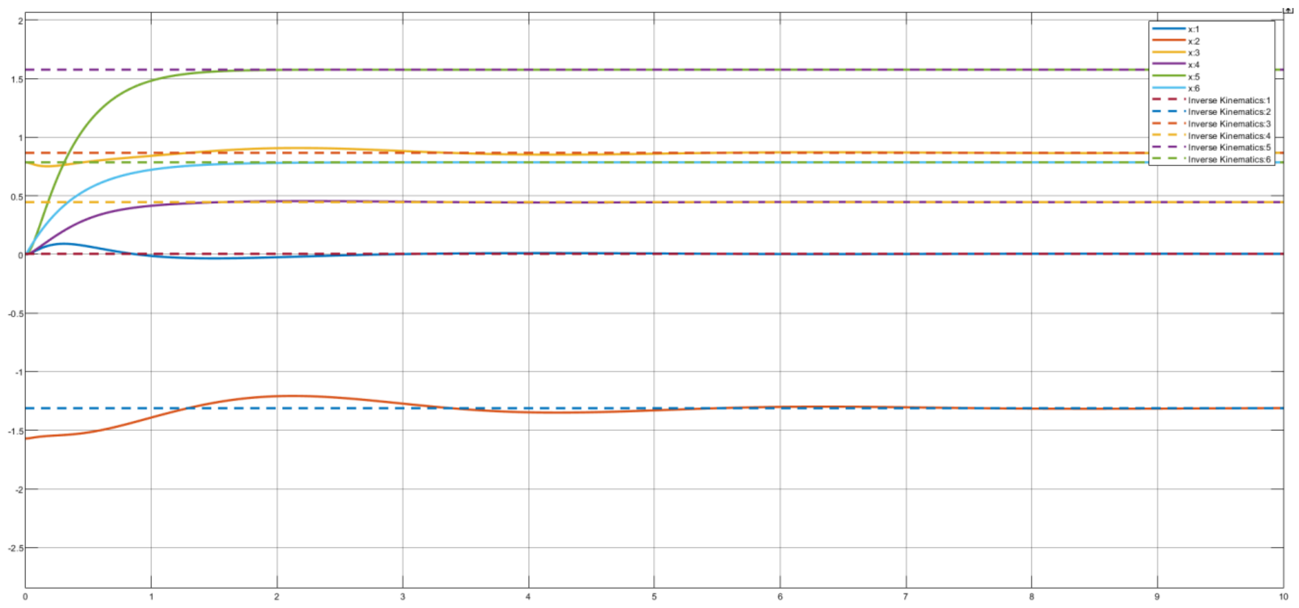

Figure 17 - Comparison of trend of the desired configuration q and the actual configuration q of the robot*

Exercise 4 – Computed Torque Control

In the fourth exercise, we must again consider the initial joint configuration q0 and calculate the trajectory in joint space to obtain q*.

q0 = (0 -1.5707 7.8539e-01 0 0 0) [rad/s]

q* = (7.8539e-01 -2.0943 1.5707 1.0471 -1.5707 3.1415) [rad/s]

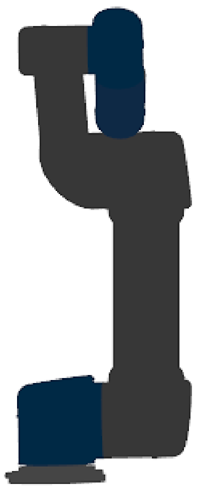

*Figure 18 - Desired joint configuration q**

Through feedback linearisation we can calculate the control torque τ to achieve the desired trajectory:

$$\tau = M(q)[q\_2dot^* - K_v\delta q - K_e\delta q] + C(q,q\_dot)q\_dot + G(q)$$

Thus, the last value of the torque τ to be applied to the manipulator results:

$$\tau = (-1.4663e\text{-}05\ 8.2480\ -1.3495e{+}01\ 8.7235e\text{-}02\ -1.3615e\text{-}05\ 1.0439e\text{-}03)\ [Nm]$$

In fact, the torque follows the following trend:

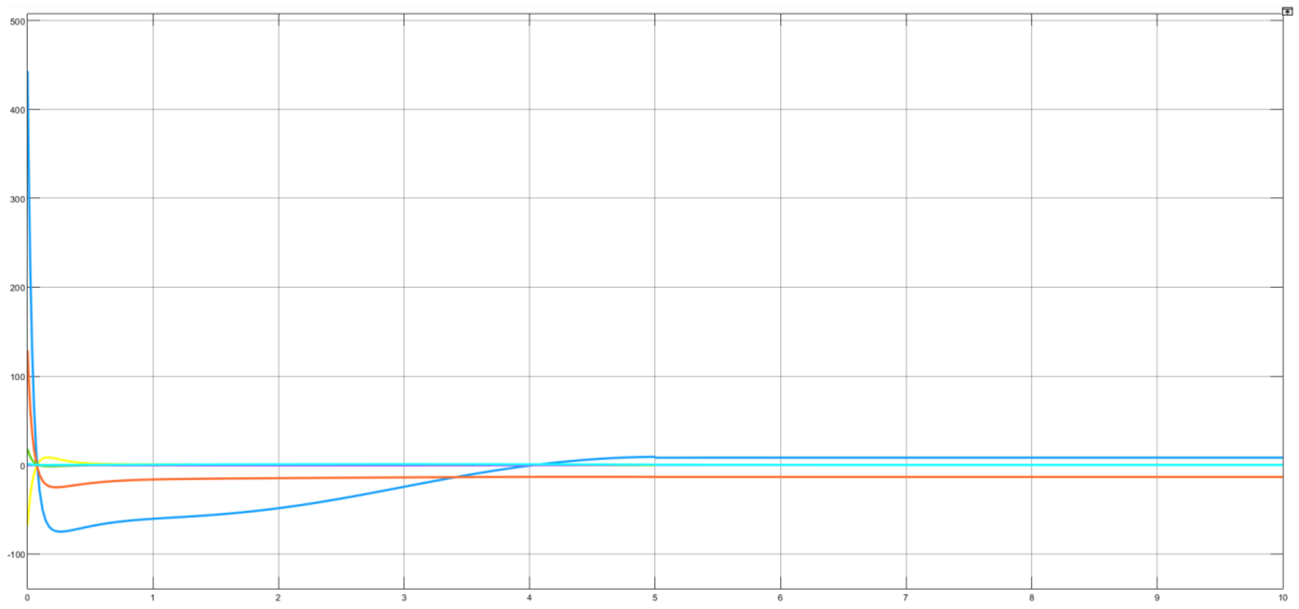

Figure 19 - Trend of torques acting on the robot

The block diagram of the system turns out to be:

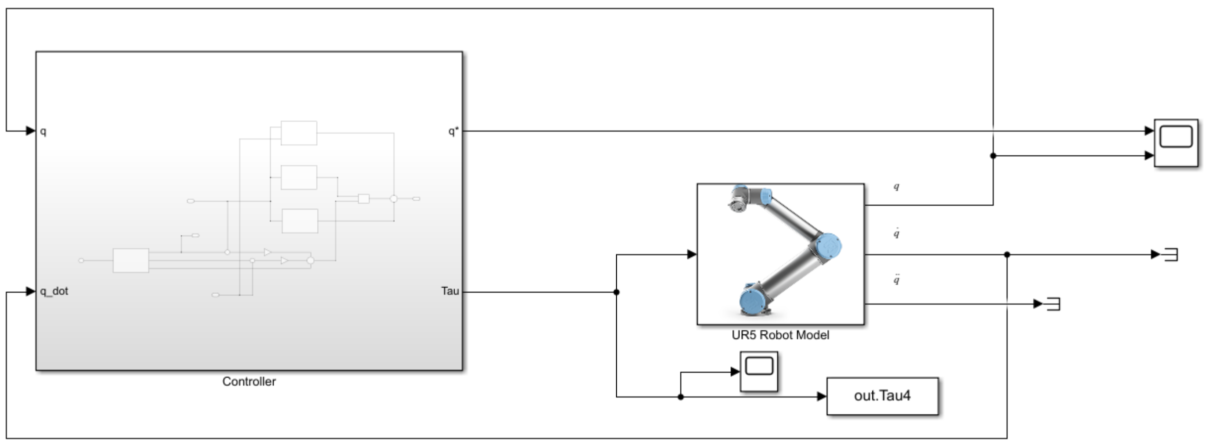

Figure 20 - Simulink block diagram

Inside the Controller block are all the components for calculating the torque τ to be applied to the manipulator:

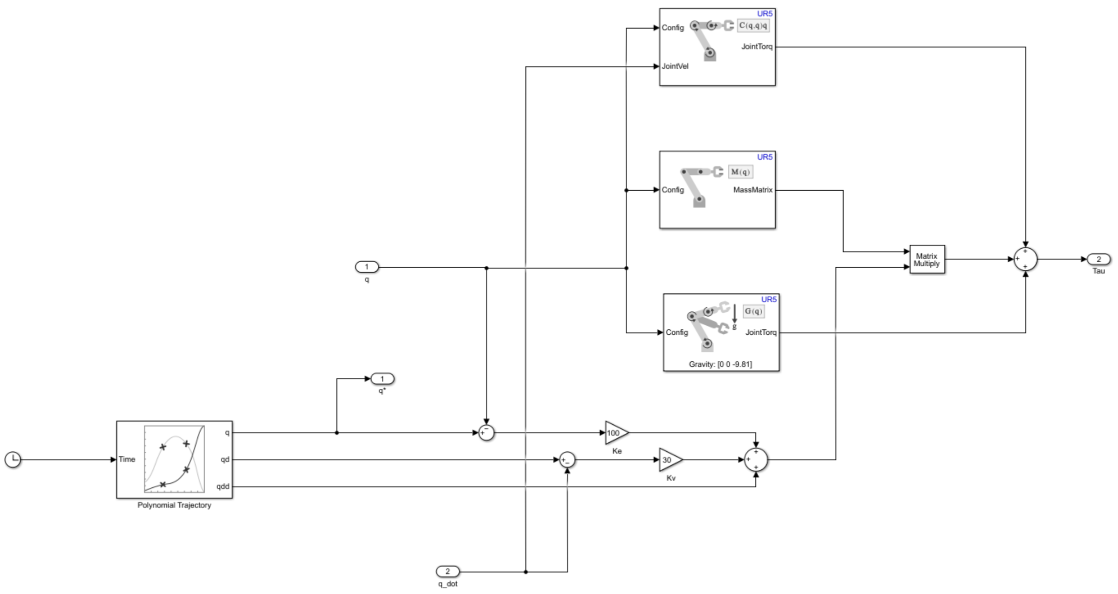

Figure 21 - Simulink block diagram

K_v = 30
K_e = 100
The values of the coefficients K_v and K_e are only changed to improve the simulation in terms of time.

In addition to the blocks in the second exercise, we must consider the Velocity Product Torque block, that allows to calculate the torque that cancel the velocity-induced forces for the given joint positions and velocities, and the Joint Space Mass Matrix, that determines the joint-space mass matrix for the given joint positions.
Instead, the Polynomial Trajectory block generates trajectories to travel through waypoints at the given time using cubic polynomials.

Through the calculation of errors dynamics, we can verify how the desired configuration differs from the real one:

δq_2dot + K_vδq_dot + K_eδq = 0

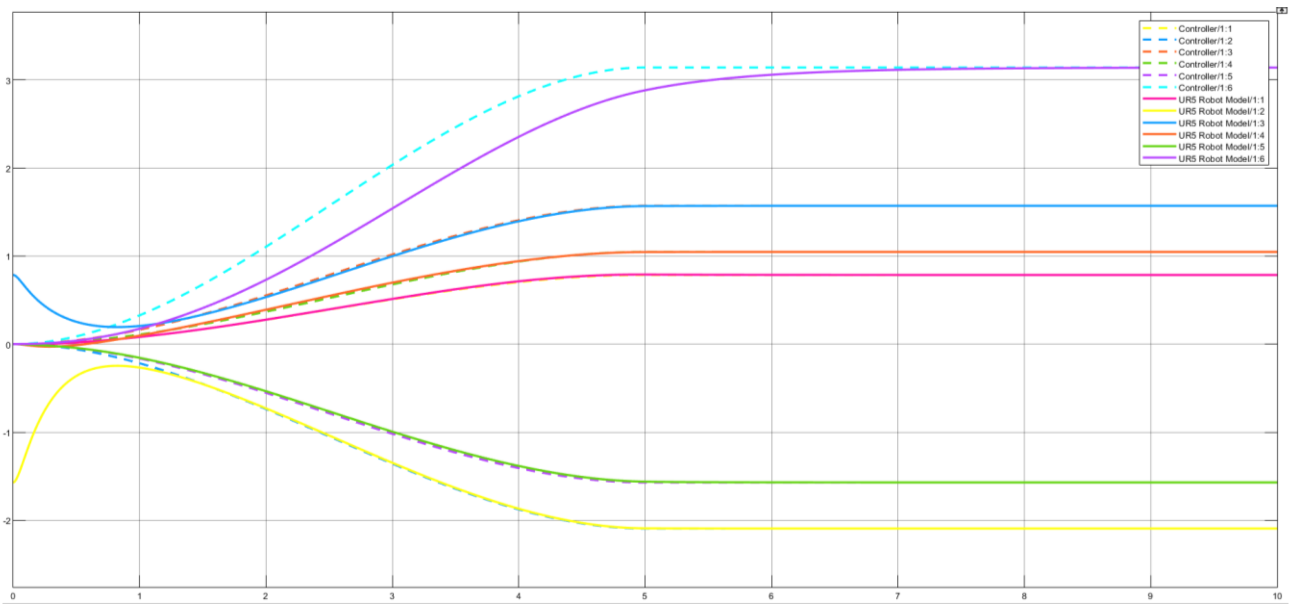

Figure 22 - Comparison of trend of the desired configuration q and the actual configuration q of the robot*

If we eliminate the damping coefficient that simulates friction in the joints, we notice that the manipulator's trajectory changes, becoming:

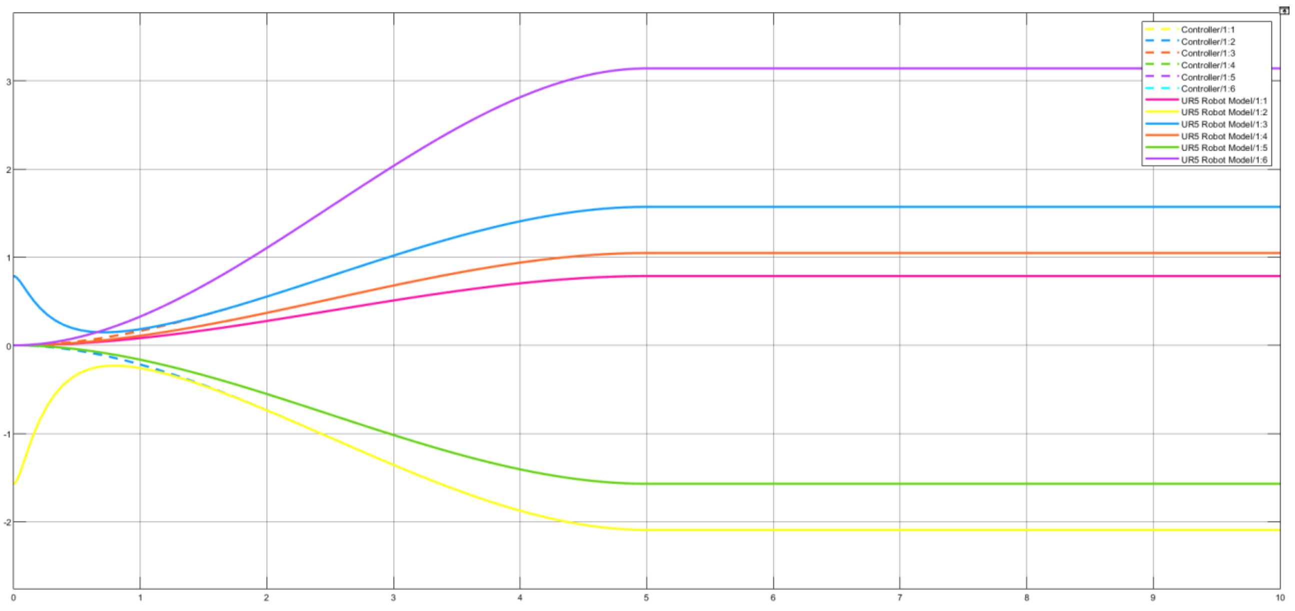

Figure 23 - Comparison of trend of the desired configuration q and the actual configuration q of the robot*

This is due to the new torque, which is equal to:

$$\tau = (8.2539e\text{-}04 \; 8.2463 \; \text{-}1.3495e\text{+}01 \; 8.7296e\text{-}02 \; \text{-}2.5513e\text{-}04 \; 2.7107e\text{-}05) \; [Nm]$$

MATLAB script

To obtain the results, just run the MATLAB script. The programme will initially load the robot data, then execute each exercise by starting the Simulink file and simulation.
For each exercise, it will print out data regarding the configurations and values of the torques to be applied to the manipulator.
To view the graphs, simply open the various scopes in the Simulink files.