Lugano Giacomo mat.5400573

UNIVERSITÀ DEGLI STUDI DI GENOVA

Robotics Engineering

AY. 2021/2022

Robot dynamics and control third assignment's report:

Dynamic robot control

0 Introduction:

This report's purpose is to explain the basics theory concepts implemented on MATLAB for the control of the collaborative manipulator ur5, we will implement different control law……

1 Theory concepts:

We will consider our robot in a moving condition, to describe his behavior we can consider the Newton-Euler applied at the i-body of the robot:

$$\begin{cases} I_{Ci}\left(\dot{\omega}_{i/0}\right) + \omega_{i/0} \times I_{Ci}\left(\omega_{i/0}\right) = M_{Ci} \\ m_i \dot{v}_{Ci/0} = F_{Ci} \end{cases}$$

We can consider the left-hand part of the equations constituted by inertial forces and moments so we can move all the term on the left side of the equation in order to obtain a dynamic equilibrium condition:

$$\begin{cases} I_{Ci}\left(\dot{\omega}_{i/0}\right) + \omega_{i/0} \times I_{Ci}\left(\omega_{i/0}\right) - M_{Ci} = 0 \\ m_i \dot{v}_{Ci/0} - F_{Ci} = 0 \end{cases}$$

Then we can evaluate the virtual work of each force and moment:

$$\begin{cases} \delta\theta_i[I_{Ci}\left(\dot{\omega}_{i/0}\right) + \omega_{i/0} \times I_{Ci}\left(\omega_{i/0}\right) - M_{Ci}] = 0 \\ \delta C_i[m_i \dot{v}_{Ci/0} - F_{Ci}] = 0 \end{cases}$$

To obtain the total virtual work we can think to sum the two equations, in a vector way it will be:

$$\begin{bmatrix} \delta\theta_i \\ \delta C_i \end{bmatrix}^T \cdot \begin{bmatrix} I_{Ci}\left(\dot{\omega}_{i/0}\right) + \omega_{i/0} \times I_{Ci}\left(\omega_{i/0}\right) - M_{Ci} \\ m_i \dot{v}_{Ci/0} - F_{Ci} \end{bmatrix} = 0$$

Where every term in the equation is projected on a common reference frame, for example the absolute one <0>, now we can perform a brief analysis of the terms in the matrix to reorganize them:

$$\begin{matrix} I_{Ci}\left(\dot{\omega}_{i/0}\right) \\ m_i \dot{v}_{Ci/0} \end{matrix} = \begin{bmatrix} I_{Ci} & 0 \\ 0 & m_i I \end{bmatrix} \begin{bmatrix} \dot{\omega}_{i/0} \\ \dot{v}_{Ci/0} \end{bmatrix}$$

$$\begin{bmatrix} M_{Ci} \\ F_{Ci} \end{bmatrix}$$

$$\begin{bmatrix} \omega_{i/0} \times I_{Ci}\left(\omega_{i/0}\right) \\ 0 \end{bmatrix}$$

In this way we can rewrite the equation in the following form:

$$\begin{bmatrix} \delta\theta_i \\ \delta C_i \end{bmatrix}^T \cdot \left\{ \begin{bmatrix} I_{Ci} & 0 \\ 0 & m_i I \end{bmatrix} \begin{bmatrix} \dot{\omega}_{i/0} \\ \dot{v}_{Ci/0} \end{bmatrix} + \begin{bmatrix} \omega_{i/0} \times I_{Ci}\left(\omega_{i/0}\right) \\ 0 \end{bmatrix} - \begin{bmatrix} M_{Ci} \\ F_{Ci} \end{bmatrix} \right\} = 0$$

Now we can remember that:

$$\begin{bmatrix} \delta\theta_i \\ \delta C_i \end{bmatrix} = J_{i/0} \delta q \qquad \begin{bmatrix} \omega_{i/0} \\ v_{Ci/0} \end{bmatrix} = J_{i/0} \dot{q}$$

Moreover, we can evaluate the time derivative of the velocity's matrix:

$$\begin{bmatrix} \dot{\omega}_{i/0} \\ \dot{v}_{Ci/0} \end{bmatrix} = J_{i/0} \ddot{q} + \dot{J}_{i/0} \dot{q}$$

Where:

- $\dot{J}_{i/0}$ is a function of q and $\dot{q}$ and each column is linear in terms of $\dot{q}$.

- $J_{i/0}$ is a function of q.
- $\begin{bmatrix} \omega_{i/0} \times I_{Ci}\left(\omega_{i/0}\right) \\ 0 \end{bmatrix}$ is a linear function of $\dot{q}$.

So we can obtain:

$$\delta q^T \left\{ J_{i/0}{}^T \begin{bmatrix} I_{Ci} & 0 \\ 0 & m_i I \end{bmatrix} \left(J_{i/0}\ddot{q} + \dot{J_{i/0}}\dot{q} \right) + J_{i/0}{}^T \begin{bmatrix} \omega_{i/0} \times I_{Ci}\left(\omega_{i/0}\right) \\ 0 \end{bmatrix} - J_{i/0}{}^T \begin{bmatrix} M_{Ci} \\ F_{Ci} \end{bmatrix} \right\} = 0$$

And we can define:

- $A_i = J_{i/0}{}^T \begin{bmatrix} I_{Ci} & 0 \\ 0 & m_i I \end{bmatrix} J_{i/0}$ which is a square symmetric semipositive matrix
- $B_i\dot{q} = J_{i/0}{}^T \begin{bmatrix} I_{Ci} & 0 \\ 0 & m_i I \end{bmatrix} \dot{J_{i/0}}\dot{q} + J_{i/0}{}^T \begin{bmatrix} \omega_{i/0} \times I_{Ci}\left(\omega_{i/0}\right) \\ 0 \end{bmatrix}$ which is a function of q and $\dot{q}$.

Then:

$$\delta q^T \{A_i\ddot{q} + B_i\dot{q}\} - \delta q^T \cdot J_{i/0}{}^T \left\{ \begin{pmatrix} M_i^{(a)} \\ F_i^{(a)} \end{pmatrix} + \begin{pmatrix} M_i^{(r)} \\ F_i^{(r)} \end{pmatrix} \right\} = 0$$

Where we've divided the forces and moments between the actuation ones and the reaction ones. We can then sum this equation for each body:

$$\delta q^T \left\{ \sum_i A_i\ddot{q} + \sum_i B_i\dot{q} \right\} - \delta q^T \cdot \sum_i J_{i/0}{}^T \begin{bmatrix} M_i^{(a)} \\ F_i^{(a)} \end{bmatrix} - \delta q^T \cdot \sum_i J_{i/0}{}^T \begin{bmatrix} M_i^{(r)} \\ F_i^{(r)} \end{bmatrix} = 0$$

Where the last term in the equation $\delta q^T \cdot \sum_i J_{i/0}{}^T \begin{bmatrix} M_i^{(r)} \\ F_i^{(r)} \end{bmatrix}$ is the total virtual work of the reaction forces and moments, which for definition is equal to zero, we can now define the inertial matrices:

$$A_{(q)} = \sum_i A_i \qquad B_{(q,\dot{q})} = \sum_i B_i\dot{q}$$

And obtain:

$$\delta q^T \left\{ A_{(q)}\ddot{q}_{(t)} + B_{(q,\dot{q})}\dot{q}_{(t)} - \tau + C_{(q)} - \sum_i J_{i/0} \begin{bmatrix} M_{Ci}^{ext} \\ F_{Ci}^{ext} \end{bmatrix} \right\} = 0$$

Where τ is the actuation control torque vector, $C_{(q)}$ is the gravity compensation vector and $\sum_i J_{i/0} \begin{bmatrix} M_{Ci}^{ext} \\ F_{Ci}^{ext} \end{bmatrix} = \tau^{ext}$ is the sum of the external forces and moments projected on the joint space.
Since δq^T is an arbitrary vector, to satisfy the equation the terms inside brace must be equal to zero, so we end up with an n set of second order differential equations in terms of $\dot{q}$ and $\ddot{q}$:

$$A_{(q)}\ddot{q}_{(t)} + B_{(q,\dot{q})}\dot{q}_{(t)} - \tau + C_{(q)} - \tau^{ext} = 0$$
$$A_{(q)}\ddot{q}_{(t)} + B_{(q,\dot{q})}\dot{q}_{(t)} + C_{(q)} = \tau + \tau^{ext}$$

Where:

- $A_{(q)}$ is the inertia matrix and it's a symmetric strictly positive definite matrix, it's invertible and bounded.
- $B_{(q,\dot{q})}\dot{q}_{(t)}$ this term considers the Coriolis and centripetal effects, it's a continues function in terms of $\dot{q}_{(t)}$ and it's bounded by a constant for every q.
- $C_{(q)}$ is the projection of the gravity force of each link on the joint space and it's bounded.

This law defines the behavior of the robot, so we need to check that it has at least one solution and possibly that the solution is unique:
Since $A_{(q)}$ is always invertible we can rewrite the equation like:

$$\ddot{q}_{(t)} = A_{(q)}{}^{-1}\left[-B_{(q,\dot{q})}\dot{q}_{(t)} - C_{(q)} + \tau + \tau^{ext}\right]$$

So we can define the state vector:

$$\underline{X} = \begin{bmatrix} q_{(t)} \\ \dot{q}_{(t)} \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Where:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = f_{(\underline{X})} + g_{(\underline{X})}[\tau + \tau^{ext}] \end{cases}$$

In which the term $\tau + \tau^{ext}$ can be seen as an input for the equation, and we can call it $\underline{u}$, so our model became:

$$\underline{\dot{X}}_{(t)} = f_{(\underline{X})} + g_{(\underline{X})}\underline{u}$$

Which is a set of first order nonlinear differential equation in terms of $\underline{X}_{(t)}$, where $A_{(q)}$, $B_{(q,\dot{q})}$ and $C_{(q)}$ are continues functions, moreover I expect that also $\underline{u}$ is a continues function, so $f_{(\underline{X})}$ is continues, and $g_{(\underline{X})}$ is a matrix.

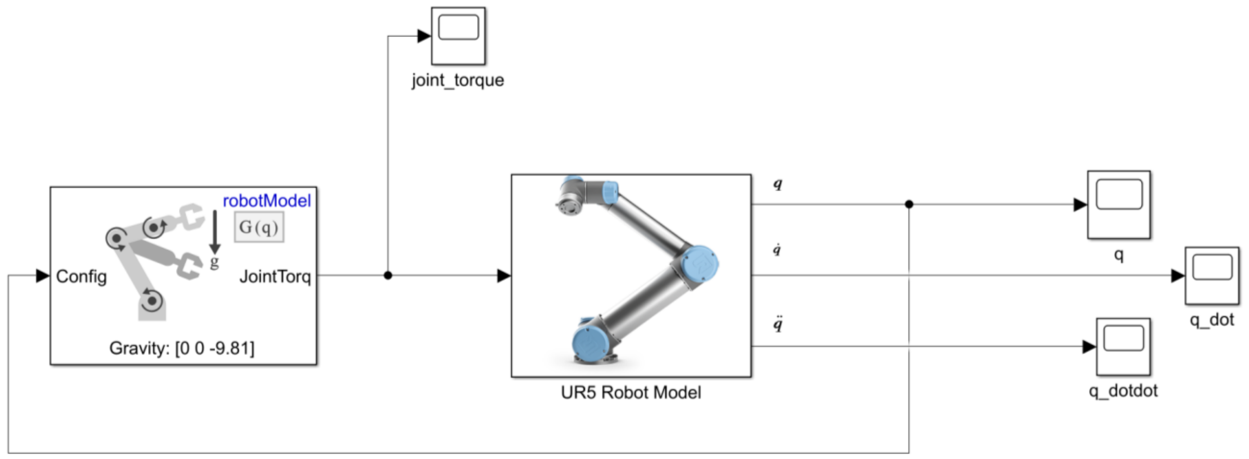
Under these assumptions and given an initial condition the equations have only one solution which is a motion consistent with our model.

$$s.t. \quad \underline{X}_{(t_0)} = \underline{X}_0 \quad \rightarrow \quad \exists! \, \underline{X}_{(t)}$$

2 Exercises:

Exercise 1: Provide torque commands to the robot such that it doesn't fall due to gravity but holds the initial configuration.

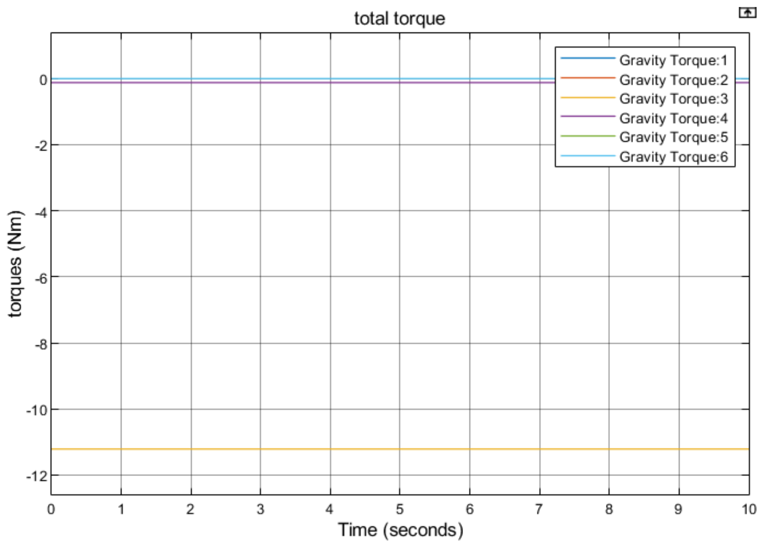
First of all let we analyse the block diagram of the exercise:



Block scheme of the program that control the robot.

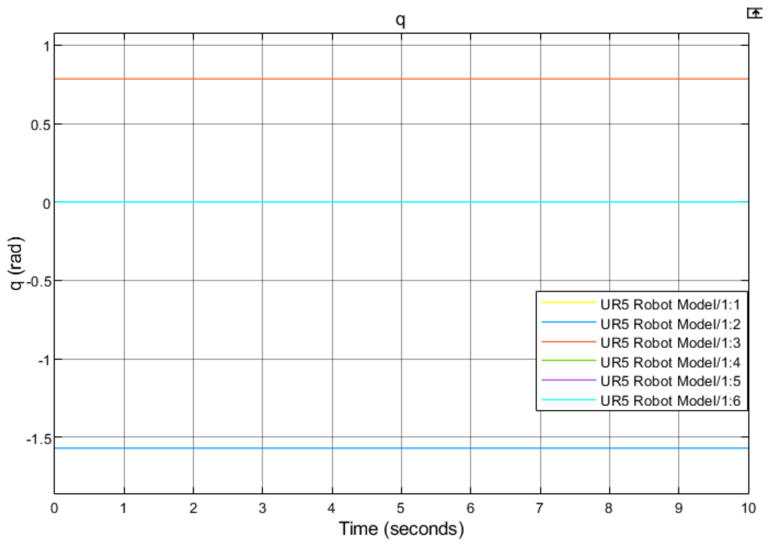
Since the request is to compute the torque necessary to compensate the gravity effect of the masses of the manipulator, we've used the robotics toolbox block named gravity torque, in particular this block compute the torque produced from the masses of the manipulator in the actual configuration, which is our $C_{(q)}$ term in the previous control equation.

Let's have a look at the graph:



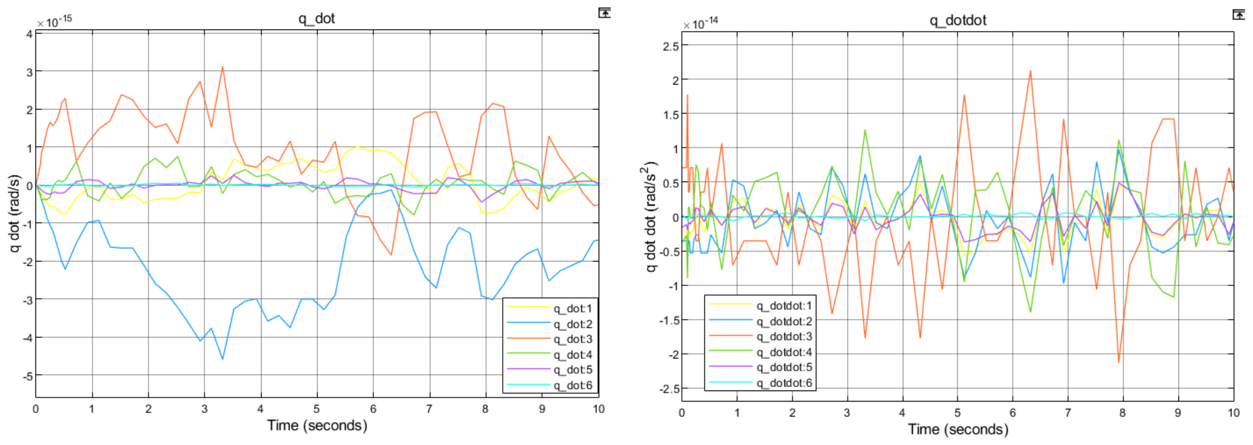
Total torque given as input at the robot.

As we can see from the torque graph and since we maintain the initial position the torque command is constant during all the simulation, more over in the graph the gravity torques 2 and 3 coincide as the gravity torques 1, 5 and 6.



Joint configuration of the robot.

As we can expect also the configuration doesn't change during the simulation, as the configuration velocities and accellerations which oscillate near zero under the control of the gravity torque block.



Joint's configuration velocity (left) and acceleration (right).

Exercise 2.1: Given a desired joint configuration q* = q0 + Δq where q0 is the initial joint configuration and Δq=[pi/4, -pi/6, pi/4, pi/3, -pi/2, pi], provide torque commands in order to reach q*.

In this case starting from the robot equation:

$$\tau = A_{(q)}\ddot{q}_{(t)} + B_{(q,\dot{q})}\dot{q}_{(t)} + C_{(q)}$$

Where we have assumed that the external torques τ^{ext} are equal to zero, we can define the problem as: we've a robot on which we want that the configuration velocity $\dot{q}_{(t)}$ goes to zero in a desired position q*.

In order to do this, we can define a potential equation like:

$$V = \frac{1}{2}\dot{q}^T A_{(q)} q + \frac{1}{2}\delta q^T K_e \delta q \qquad where\ \delta q = q_{(t)} - q*$$

In which the first term represents the kinematic energy of the robot and the second one an elastic energy that will dissipate the kinematic energy obtaining an equilibrium point (for how I've defined δq) in the q* configuration, where K_e is a square, symmetric, strictly positive definite matrix.

Then we can compute the time derivative of our potential energy function to extrapolate a control law:

$$\dot{V} = \dot{q}^T A_{(q)}\ddot{q} + \frac{1}{2}\dot{q}^T \dot{A}_{(q)}\dot{q} + \delta q^T K_e \dot{\delta q}$$

Where:

$$\frac{\partial}{\partial t}\delta q^T K_e \delta q = \delta q^T K_e \dot{\delta q} \qquad \dot{\delta q} = \dot{q}_{(t)} - \dot{q}* = \dot{q}_{(t)} \quad A_{(q)} = \tau - B_{(q,\dot{q})}\dot{q}_{(t)} - C_{(q)}$$

So:

$$\dot{V} = \dot{q}^T\left[\tau - C_{(q)}\right] + \dot{q}^T\left[\frac{1}{2}\dot{A}_{(q)}\dot{q} - B_{(q,\dot{q})}\right]\dot{q} + \dot{q}^T K_e \delta q$$

$$\left[\frac{1}{2}\dot{A}_{(q)}\dot{q} - B_{(q,\dot{q})}\right] = screw\ symmetric \rightarrow \dot{q}^T\left[\frac{1}{2}\dot{A}_{(q)}\dot{q} - B_{(q,\dot{q})}\right]\dot{q} = 0$$

$$\dot{V} = \dot{q}^T\left[\tau + K_e\delta q - C_{(q)}\right]$$

Now $\dot{q}^T K_e \delta q$ and $\dot{q}^T C_{(q)}$ could have any sign so we want to get rid of them by choosing τ as:

$$\tau = -K_e\delta q + C_{(q)} - K_v\dot{q}$$

Where $-K_v\dot{q}$ is used to make $\dot{V}$ strictly negative definite:

$$\dot{V} = -\dot{q}^T K_v \dot{q}$$

Where K_v is strictly positive.

Now $\dot{q} = 0$ is the only case in which $\dot{V} = 0$, and in this situation:

$$\tau = A_{(q)}\ddot{q}_{(t)} + B_{(q,\dot{q})}\dot{q}_{(t)} + C_{(q)} = -K_e\delta q + C_{(q)} - K_v\dot{q}$$

$$A_{(q)}\ddot{q}_{(t)} = -K_e\delta q$$

We can rearrange the equation to obtain:

$$\ddot{q}_{(t)} = -A_{(q)}{}^{-1}K_e\delta q = 0 \ \leftrightarrow\ \delta q = 0$$

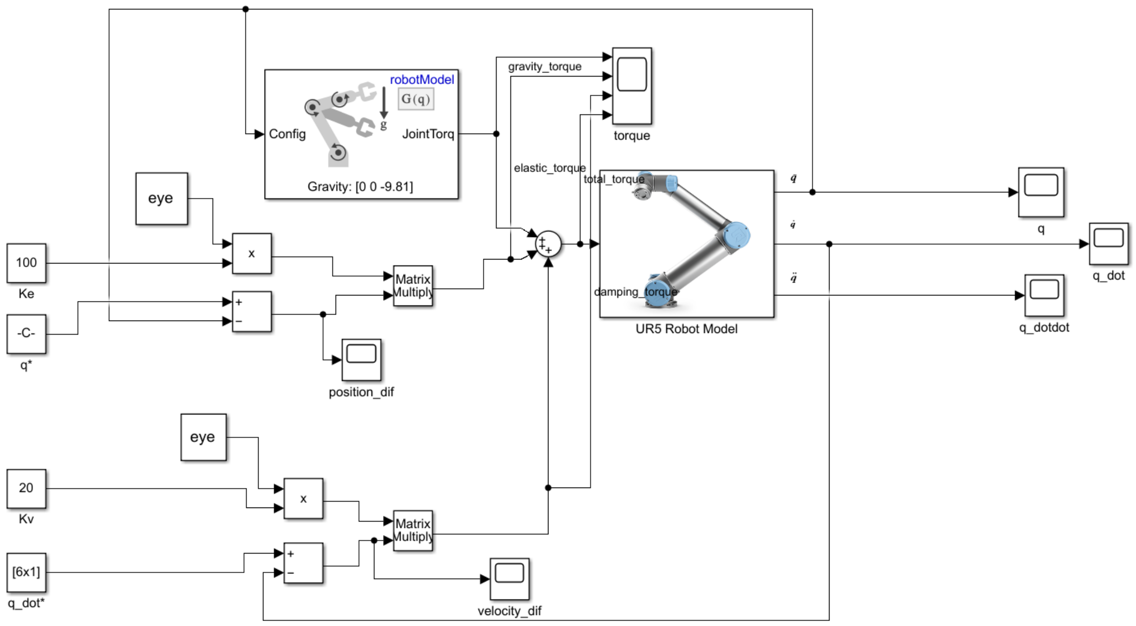
So, we have proved that the system can never stop in a position different from $\delta q = 0$, which is our desired position, and we've found an asymptotically stable equilibrium and the larger is K_e, the larger will be $\left|\ddot{q}_{(t)}\right|$ so we will reach the target position quicker.

In conclusion our control law to implement in our software will be:

$$\tau = -K_e\delta q + C_{(q)} - K_v\dot{q}$$

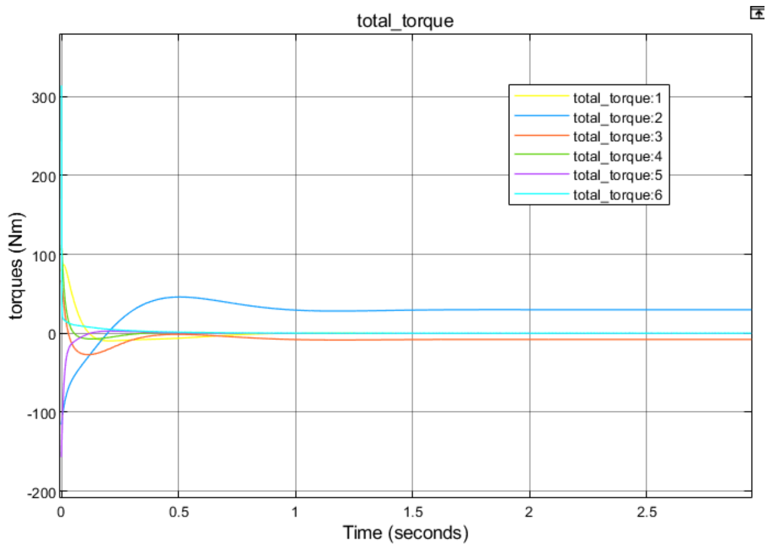
Where usually we choose K_e and K_v to be diagonal.

This control law has been implemented in Simulink as it follows:



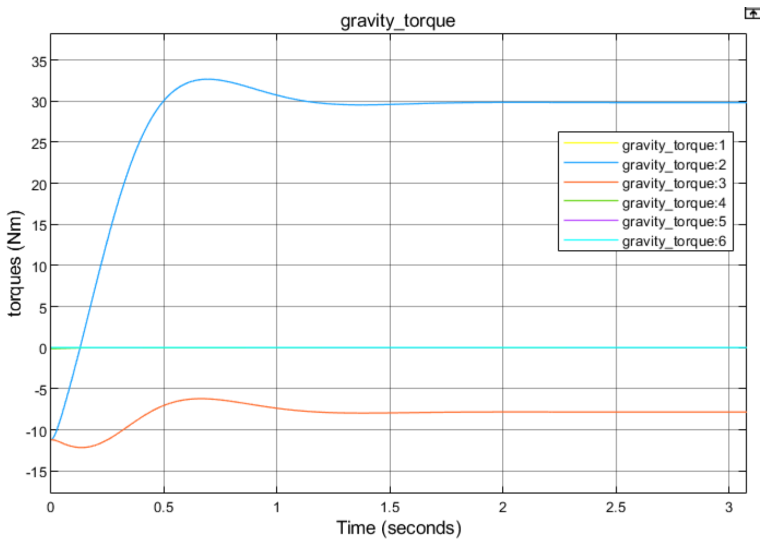
Block scheme of the program that control the robot.

We can see how the control errors for both the position δq and velocities $\delta\dot{q}$ have been implemented as a difference between the actual value of the configuration/configuration velocity and the desired one, then the results have been multiplied by the respective matrix K_e or K_v and in the end summed together with also the gravity torque to obtain the law defined above.
In the simulation the desired position q* has been set as q*= [pi/4; -(5*pi)/6; pi/2; pi/3; -pi/2; pi], while since we want to stop the robot in the above defined position q*_dot=[0; 0; 0; 0; 0; 0].
The elastic and damping matrices have been set as a diagonal matrix equal to respectively $K_e = 100\ I$ and $K_v = 20\ I$ (where I the identity matrix of dimension 6x6).
We can now investigate the results of this control law thanks to the graphs:



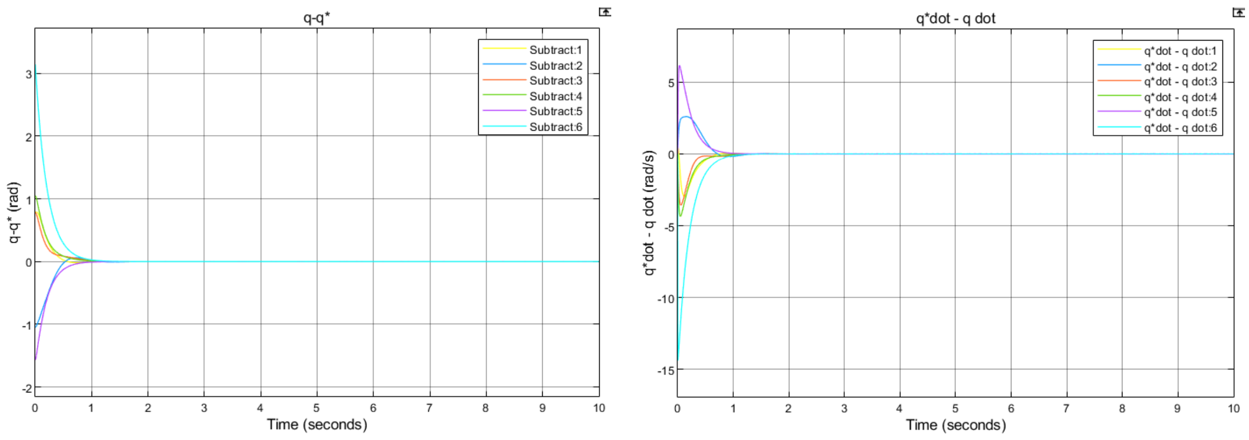
Total torque given as input at the robot.

As we can see after an initial phase were the torques change to reach the desired position they approach asymptotically a stable value, which is equal to the gravity torque for the final configuration of the robot, as we can see from his graph:



Gravity torque computed by the program.

Moreover we can check how the control errors for both the configuration δq and the configuration velocities $\delta \dot{q}$ change during the simulation:



Control configuration velocity error (right) and control configuration error (left).

As we can see both the errors goes to zero asymptotically defining a stable equilibrium condition imposed by our control law.

Exercise 2.2: Given a desired joint configuration q* = q0 + Δq where q0 is the initial joint configuration and Δq=[pi/4, -pi/6, pi/4, pi/3, -pi/2, pi], provide torque commands in order to reach q*, without compensate the gravity effect.

In this case starting from the theory described for the case above we can consider that the gravity is a conservative force with an associated potential function $U_{(q)} + U_0 > 0$, where U_0 is used to set the zero of the function, such that:

$$\left(\frac{\partial U}{\partial q}\right)^T = C_{(q)}$$

So in we can add this term in the potential equation defined above to obtain:

$$V = \frac{1}{2}\dot{q}^T A_{(q)} q + \frac{1}{2}\delta q^T K_e \delta q + U_{(q)} + U_0$$

And we can repeat the same procedure of the previous exercise:

$$\dot{V} = \dot{q}^T\left[\tau + K_e\delta q - C_{(q)}\right] + \left(\frac{\partial U}{\partial q}\right)^T \dot{q}$$

Where:

$$\left(\frac{\partial U}{\partial q}\right)^T \dot{q} = \dot{q}^T \left(\frac{\partial U}{\partial q}\right)^T = \dot{q}^T C_{(q)}$$

$$\dot{V} = \dot{q}^T[\tau + K_e\delta q]$$

And we can define a control law as:

$$\tau = -K_e\delta q - K_v\dot{q}$$

Obtaining a gravity compensated PD (proportional derivatives) control action.
In this way when $\dot{q} = 0$ we obtain:

$$A_{(q)}\ddot{q}_{(t)} + C_{(q)} = -K_e\delta q$$

And here we can't ensure that δq goes to zero and the equilibrium is simple stable, in fact every configuration like:

$$C_{(q)} = -K_e\delta q$$

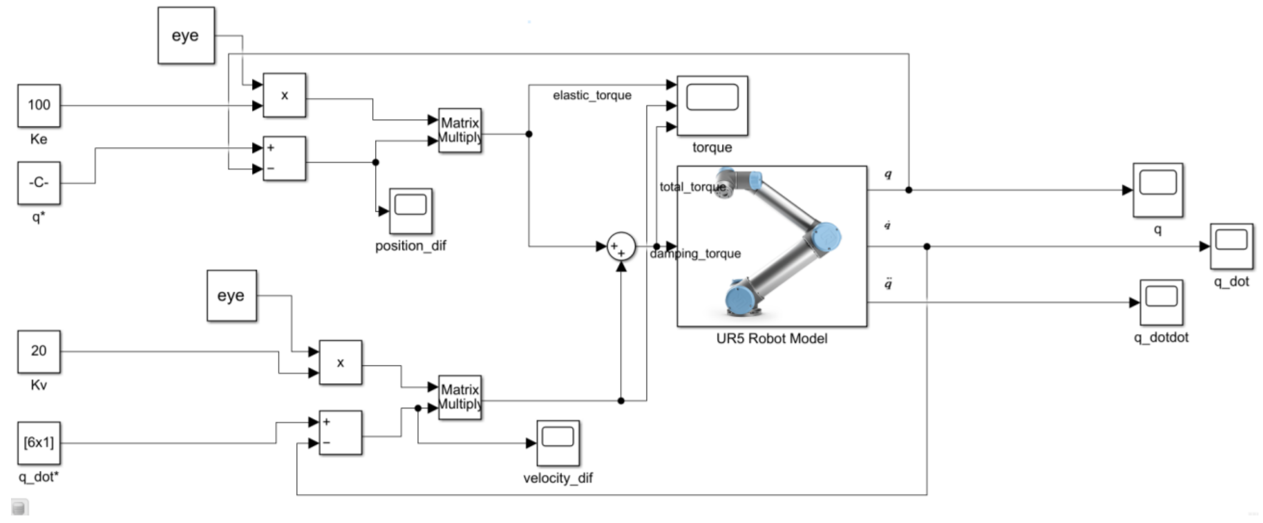
are an equilibrium configuration, but we may consider that K_e is invertible and so we can obtain these configurations as:

$$\delta q = -K_e^{-1}C_{(q)}$$

Which is an error range around our desired configuration q^*, and is bounded:

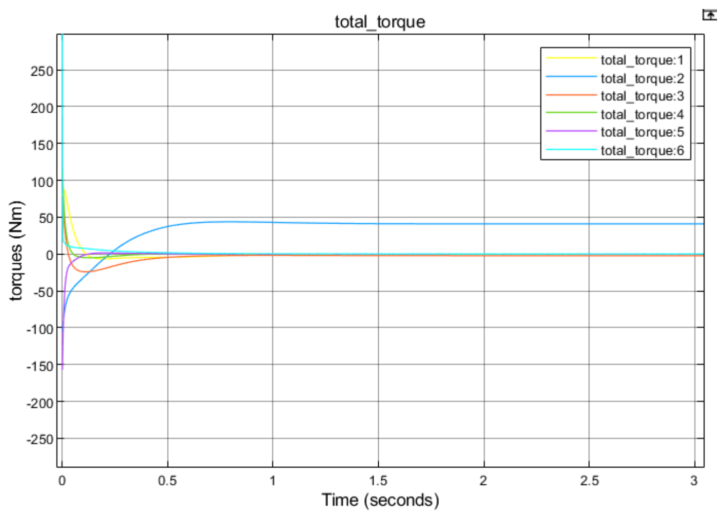
$$\|\delta q\| \leq \|K_e^{-1}\|C_0$$

Where C_0 is the maximum value of $C_{(q)}$, his upper bound, and the larger K_e will be, the smaller the error $\|\delta q\|$ will be but also the noises and other dynamic effects will be larger.
Back to our simulation the block scheme has become:



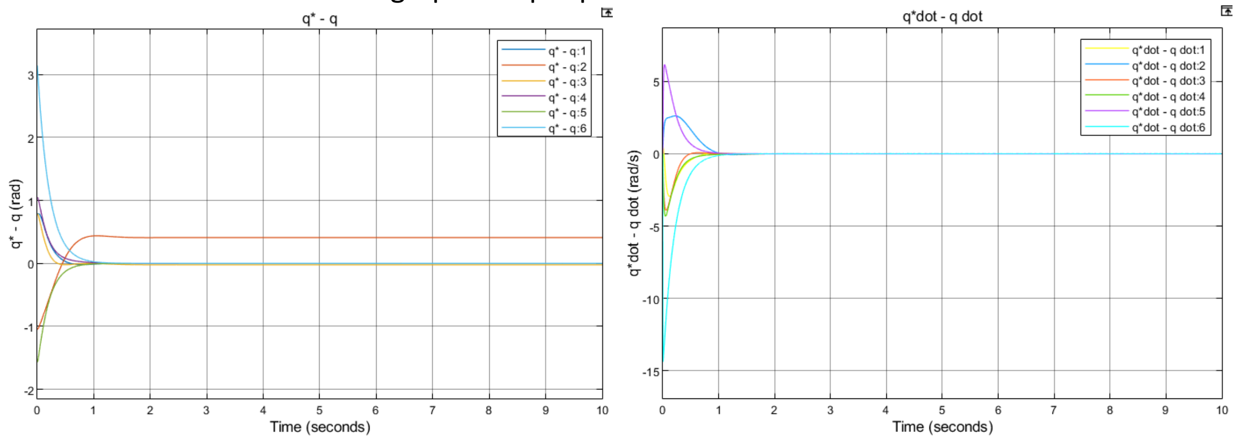
Block scheme of the program that control the robot.

Which is equal to the scheme above but without the gravity torque block, and with the same values for q*, q*_dot, K_e and K_v.
We can now check the graph to see the differences from the previous exercise:



Total torque given as input at the robot.

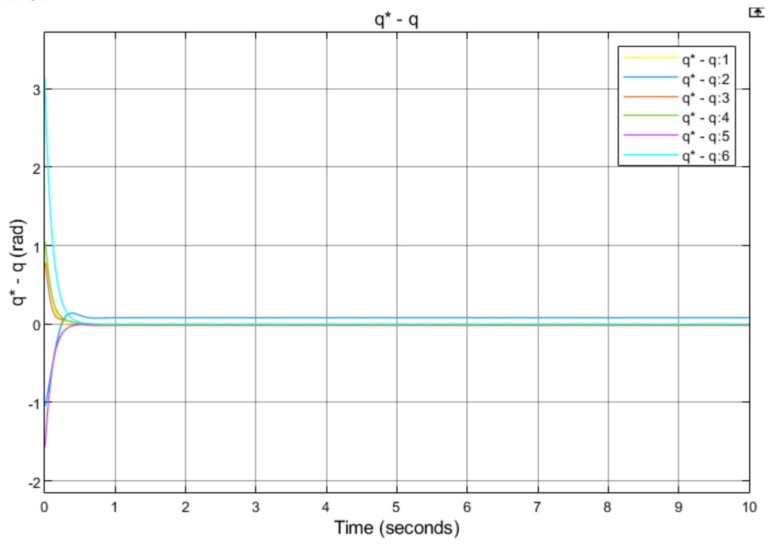
As we can see the total control torque τ is a little bit different from before, but the main difference can be seen in the graphs of q*- q:



Control configuration velocity error (right) and control configuration error (left).

In fact, while the configuration velocity converges to zero for all the joints, the difference between the desired configuration and the actual one converges to a value different from zero, which define our error.

As said above we can try to increase the value of K_e to decrease the error, in fact if we set $K_e = 400$ the graph become:

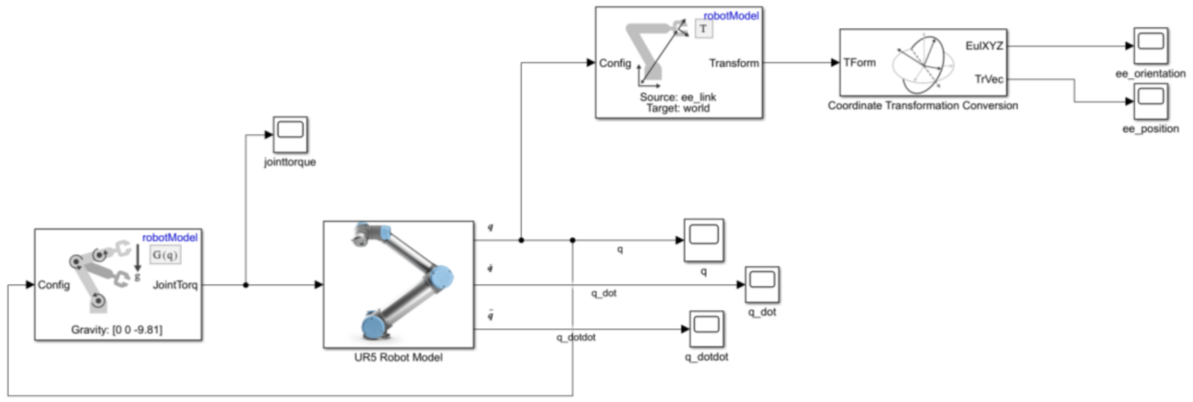


Control configuration error.

Where we can easily spot the difference in the final configuration error.

Exercise 3: Given a desired cartesian configuration x*=x0+Δx, where x0 is the initial cartesian pose of the end effector ('ee_link') and Δx=[0.2, -0.08, -0.15, pi/4, -pi/4, pi/2], provide torque commands in order to reach x*.

First of all, in order to obtain the initial end effector position and orientation we've used the first exercise block scheme with an additional part:



Block scheme of the program used to obtain the initial position and orientation of the end effector.

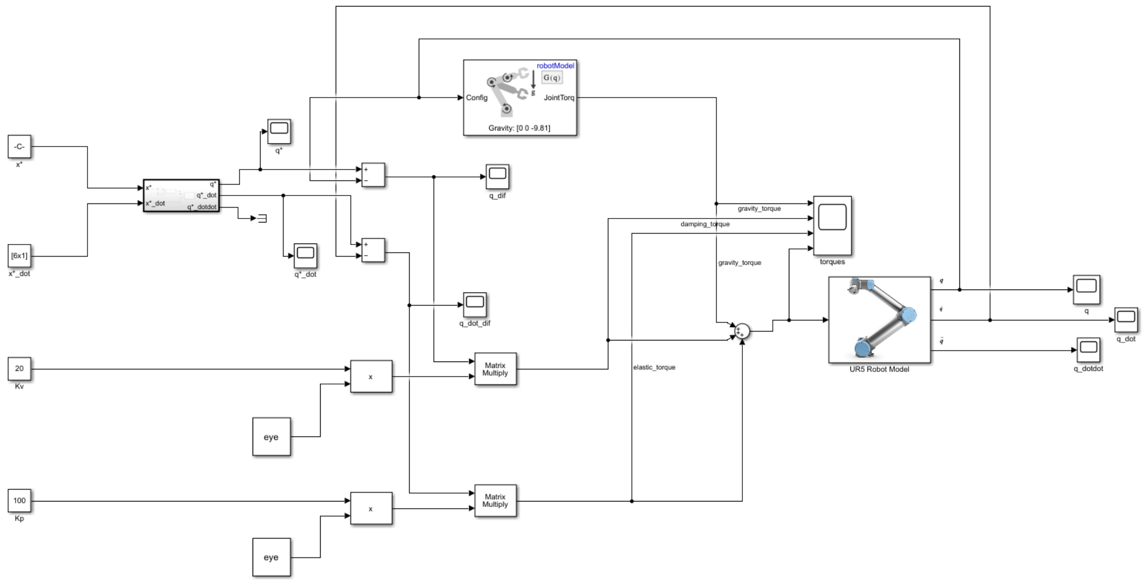
Where the first additional block (get transform) computes the transformation matrix from the world reference frame and the end effector frame, while the second (coordinates transformation conversion) translate the transformation matrix in the position and orientation, expressed by the Euler angles, of the end effector.

Thanks to this procedure we've obtained the initial position and orientation vector, and from it the desired final position and orientation vector:

$$X_0 = [0.3442\ \ 0.1914\ \ 0.7245 - \pi\ \ {}^{\pi}/_{4}\ \ {}^{-\pi}/_{2}]$$

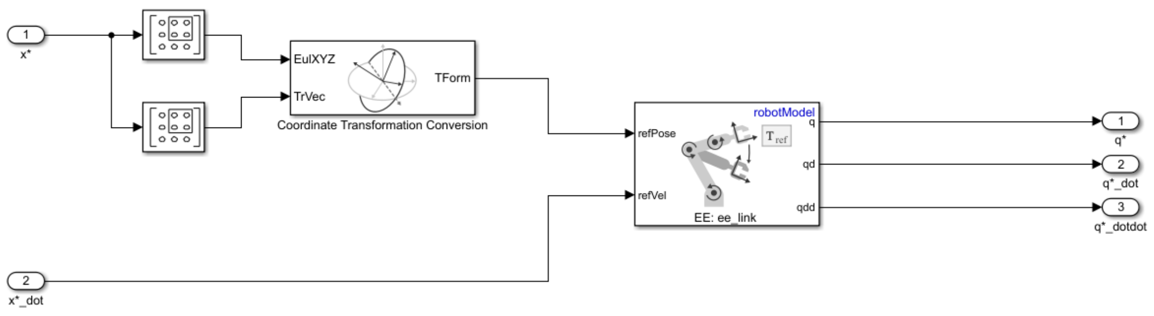
$$X^* = X_0 + \Delta X = [0.5442\ \ 0.1114\ \ 0.5745\ \ {}^{-3\pi}/_{4}\ \ 0\ \ 0]$$

Then we can use the same control law used in the Exercise 2.1 implemented as before with an addition in order to put as input the position and orientation of the end effector instead of the configuration vector:



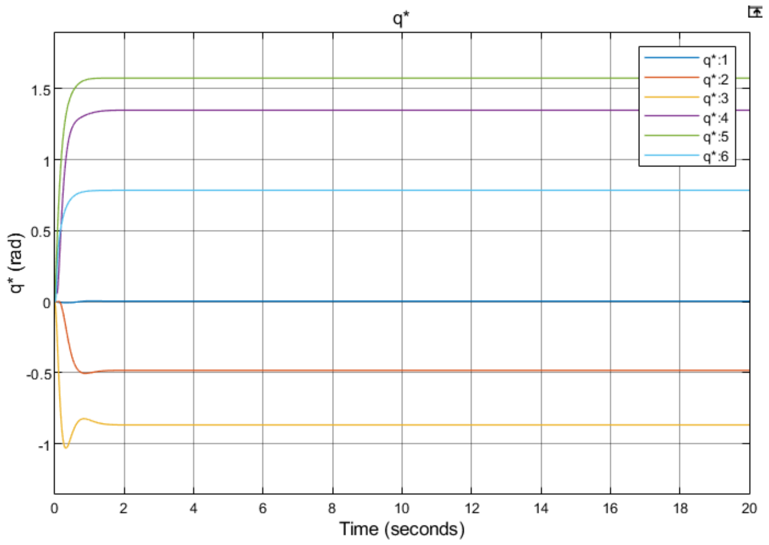
Block scheme of the program that control the robot.

where the sub system contains:



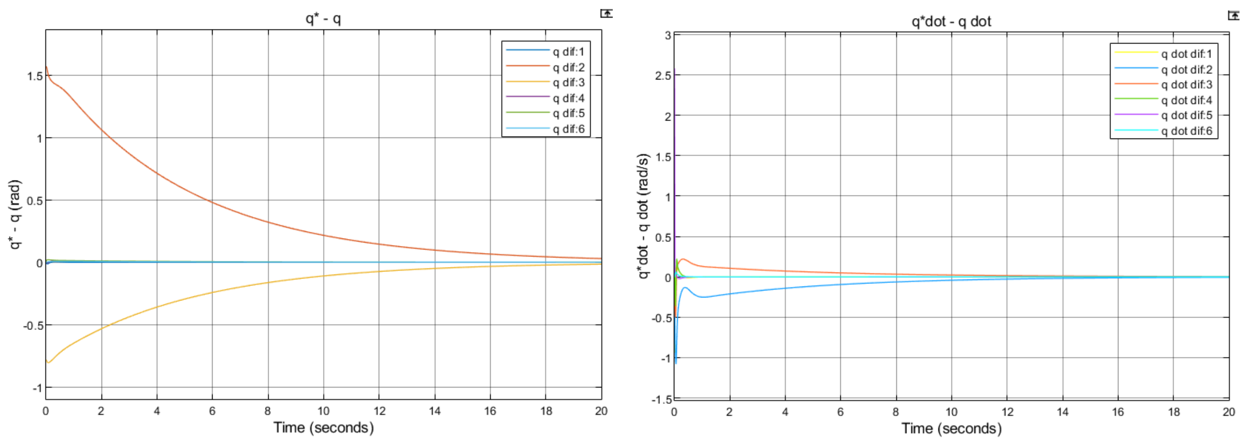
Block scheme of the subsystem.

The first two blocks (submatrix block) that follow the first input (x*) are used to divide the vector between the position part and the orientation part, that goes into the coordinates transformation conversion block which gives as output the transformation matrix between the end effector frame and the world reference frame which is used in input in the task space motion model block which convert it in the configuration vector q* of the robot.
As we can see the subsystem gives as out put the configuration vector q* starting from the zero configuration.



Desired configuration vector computed from the subsystem.

We can also observe that to reach the desired final configuration of the robot, with the same choice of K_e and K_v done before, the simulation requires nearly twenty seconds, for this reason it can be a good idea increase the values of K_e and K_v.



Control configuration velocity error (right) and control configuration error (left).

By the way we can see how the implemented law converge asymptotically the robot to a stable equilibrium in the desired configuration.

Exercise 4.1: In industrial robot control system even a simple regulation from a setpoint to another (like in Ex.2 and Ex.3) is done via joint trajectory tracking. Consider again q* from Ex.2 and generate joint-space trajectories in order to reach it from q0. Use feedback linearization to implement a computed torque control in order to track the desired joint trajectories.

In this case we can start from the dynamic model of the robot:

$$\tau = A_{(q)}\ddot{q}_{(t)} + B_{(q,\dot{q})}\dot{q}_{(t)} + C_{(q)}$$

Now a good control signal should compensate the gravity torque $C_{(q)}$ as the inertial torque $B_{(q,\dot{q})}$ and both the terms can be computed as well as $A_{(q)}$ which can be multiplied by an exogenous control signal $v_{(t)}$, so we can control the robot by a control law:

$$\tau = B_{(q,\dot{q})}\dot{q}_{(t)} + C_{(q)} + A_{(q)}v_{(t)}$$

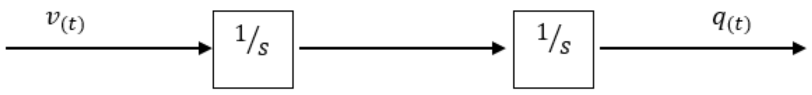
Then we can substitute the control signal in the dynamic model of the robot to obtain:

$$A_{(q)}\ddot{q}_{(t)} = A_{(q)}v_{(t)}$$

So in close loop the effect of our control signal is to transform the dynamic model in the above set of differential equations, and since $A_{(q)}$ is invertible we can write:

$$\ddot{q}_{(t)} = v_{(t)}$$

Which is a set of linear second order differential equations.
Now in our implementation $A_{(q)}$, $B_{(q,\dot{q})}$ and $C_{(q)}$ can be computed by the Newton-Euler equations (or directly by some dedicated functions), $\dot{q}_{(t)}$ and $q_{(t)}$ are a feedback signal from our robot while $v_{(t)}$ will be specified by us.
Now we may think that due to the effect of this nonlinear compensation law τ the robot will look like:



Linearized robot model.

So we've canceled all the nonlinear part, and we can think to implement an exogenous signal $v_{(t)}$ as :

$$v_{(t)} = \ddot{q} = -K_v\delta\dot{q} - K_p\delta q + \ddot{q}^*$$

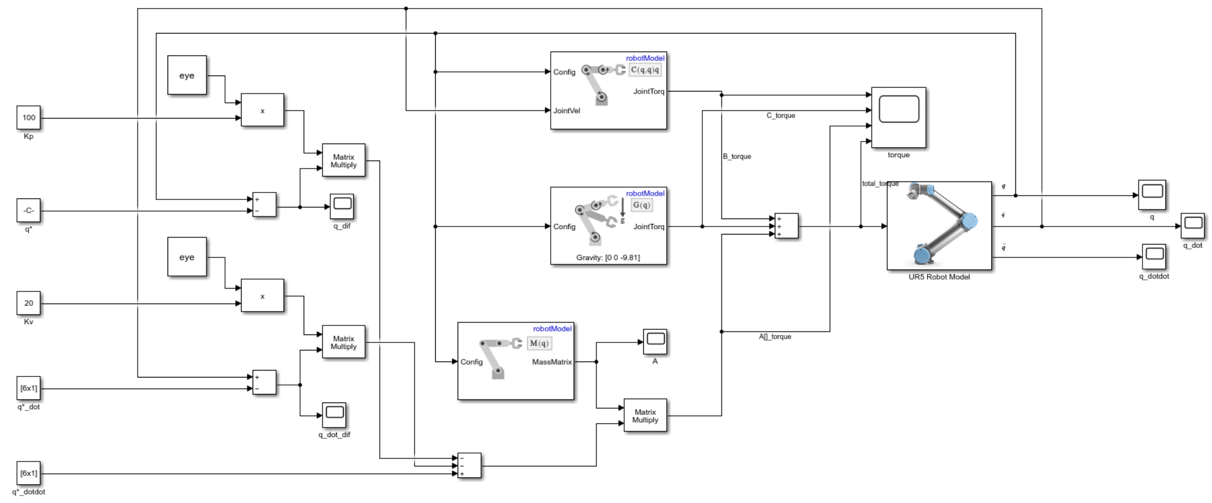
Where we've defined the control error in joint space $\delta\dot{q} = (\dot{q} - \dot{q}^*)$, $\delta q = (q - q^*)$ and the tracking acceleration $\ddot{q}^*$ which can be implemented in a control error like $\delta\ddot{q} = (\ddot{q} - \ddot{q}^*)$ and we can rewrite the differential equation like:

$$\ddot{q} = -K_v\delta\dot{q} - K_p\delta q + \ddot{q}^* \quad \rightarrow \quad (\ddot{q} - \ddot{q}^*) = -K_v\delta\dot{q} - K_p\delta q = \delta\ddot{q}$$

Which is a differential equation in terms of the control errors and it's the close loop dynamics model which describe the effect of the control action on the robot.
Moreover we can rewrite the equation like:

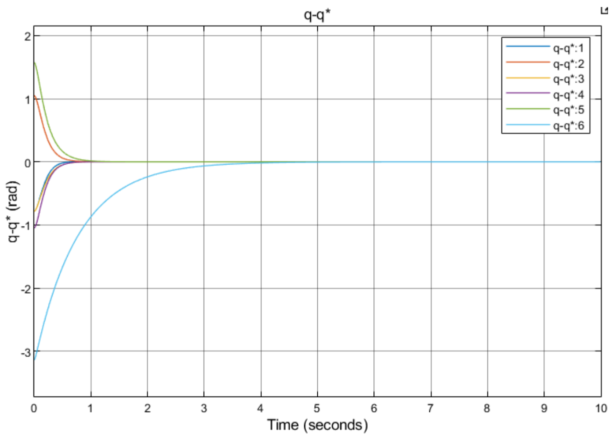
$$\delta\ddot{q} + K_v\delta\dot{q} + K_p\delta q = 0$$

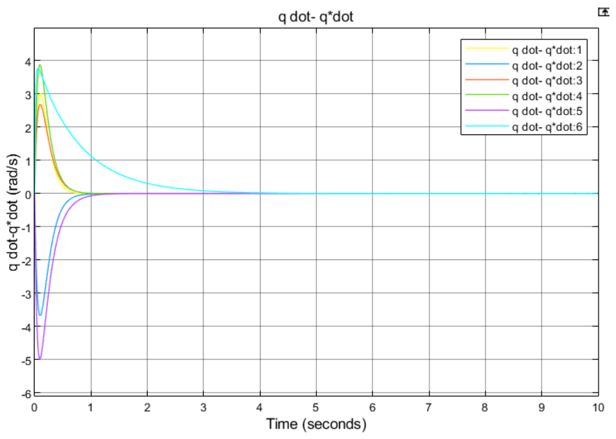
Which is a set of independent second order differential equation in homogeneous form, so δq is going to zero providing that the Engen values of this equations are all strictly in the left complex plane, and if we assume that the matrices K_p and K_v are diagonal and strictly positive, we can ensure that the Engen values of the equations are strictly negative in the real part, so we can say that:
$\forall K_p > 0, K_v > 0$ the origin ($\delta q = 0, \delta\dot{q} = 0$) of the state space for the system described by the equation is a global asymptotically stable equilibrium point.
Let we see the implementation of the above defined control law:



Block scheme of the program that control the robot.

As we can see in input we've the tracking configuration, velocity ad acceleration which are subtracted with the actual robot values of the homonymous quantities and then multiplied by the matrices K_p and K_v for the configuration and configuration velocities, while the tracking acceleration is just added.
Then the result is multiplied by the matrix $A_{(q)}$,obtained from the joint space mass matrix block, and summed with the matrix $B_{(q,\dot{q})}$, obtained from the velocity product torque block, and the matrix $C_{(q)}$, obtained from the gravity torque block.
As we can see from the graphs the desired configuration is achieved after some seconds, with the desired configuration velocity:

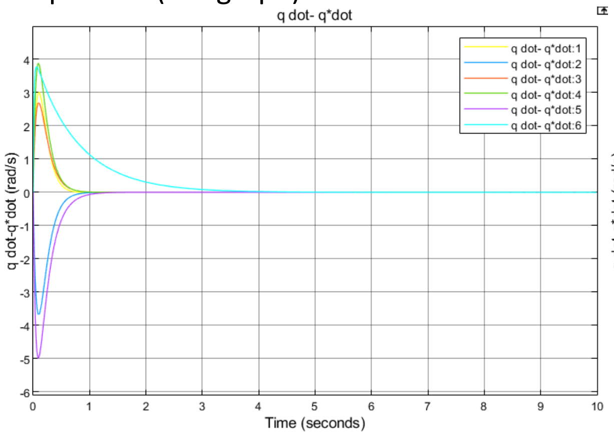


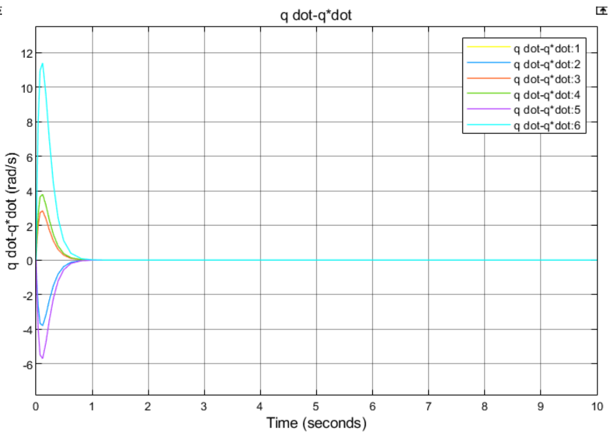


Control configuration velocity error (right) and control configuration error (left).

Exercise 4.2: In industrial robot control system even a simple regulation from a setpoint to another (like in Ex.2 and Ex.3) is done via joint trajectory tracking. Consider again q* from Ex.2 and generate joint-space trajectories in order to reach it from q0. Open the robot model and check the joint object: at the label 'internal mechanics' you will notice that a damping coefficient is provided to simulate joint friction; try to remove it from each joint and compare the performance of the computed torque controller.

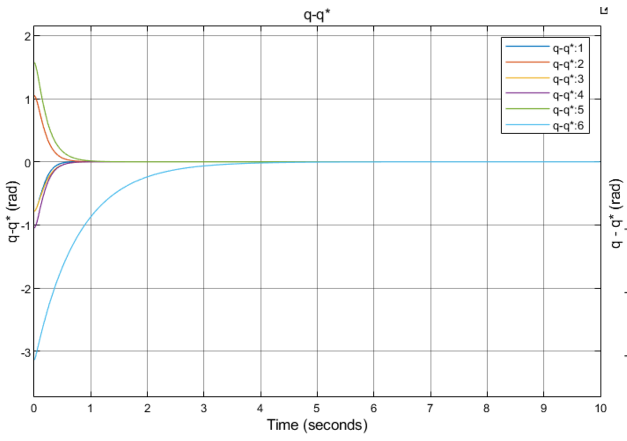
The controller scheme used in this simulation is the same of the previous one, let we investigate the graphs and their differences:
first of all the damping coefficient used in the joint simulate the joint's friction as a torque directly proportional to the joint speed.
So by turning it off we will have that for a given torque the joint speed $\dot{q}$ will be bigger, and that can be seen from the graph of $\dot{q} - \dot{q}^*$, indeed we can notice that since $\dot{q}^* = 0$ the graph rapresent $\dot{q}$ which is bigger in the case where the friction is neglected (right graph) respect the one where it's present (left graph):

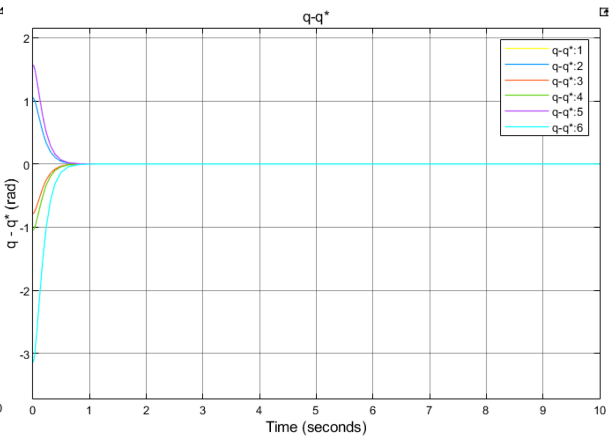




Control configuration velocity error with the joint's friction (left) and without it (right).

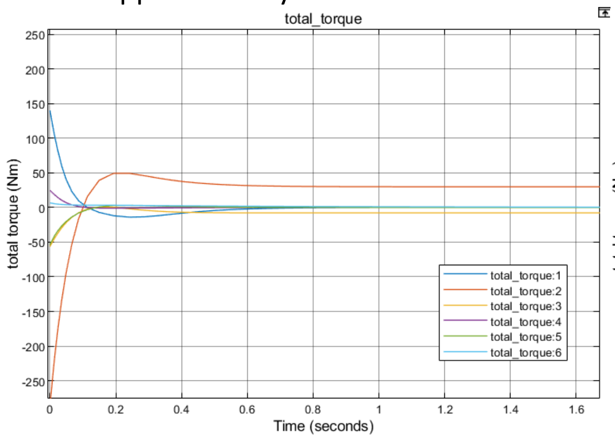
Starting from this consideration we can understand that a bigger joint speed for a given torque will leads to a quicker convergence of the joint configuration to the desired one as we can see from the $q - q^*$ graph:

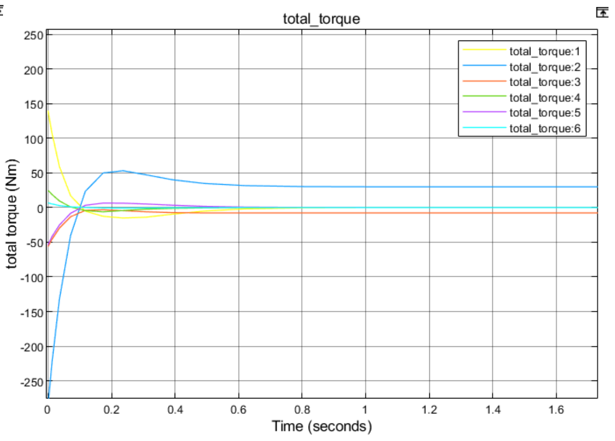




Control configuration error with the joint's friction (left) and without it (right).

these differences have a small effect on the global control torque provided at the robot that remain approximately the same:





Total torque applied on the robot with the joint's friction (left) and without it (right).