Laboratory N.3

Michele Tartari, Stefano Castagnetta

October 31, 2017

Abstract

This paper represents the report for MoCom Laboratory N.3 held on the 23^{rd} October 2017 and during which it was addressed the problem of motion planning and robotic control. This paper can be divided in two main sections, in the first we developed and validated the trajectory generator for the Scara robot. Differently, for the second part we implemented PID and Torque control law.

Contents

1. Trajectory Generation

1.1. inter_arti_d5.m

We implemented a MATLAB function that alongside a main file (init_d5.m) is capable to evaluate and plot a motion in the joint space using an interpolation function of degree 5. This function is a motion profile in which the initial and final values are null for joint position, velocity and acceleration. The general expression can be defined as:

$$q(t) = a_0 + a_1 \cdot t + a_2 \cdot t^2 + a_3 \cdot t^3 + a_4 \cdot t^4 + a_5 \cdot t^5$$

We applied to such case the studies conducted by Bindford in 1977. To do that we first calculated the maximum of minimum time $t_{f,j}$ required to perform the motion for each joint. We can assume the end-effector will be moving on a straight line between initial position q^i and final position and the distance between the two of them can be written for each joint as:

$$D_j = q_j^f - q_j^i$$

Given the velocity saturation $k_{v,j}$ and the acceleration saturation $k_{a,j}$ for each joint, we have:

$$\begin{aligned} t_{f,j} &= MAX\left[\frac{3|D_j|}{2k_{v,j}}, \frac{3|D_j|}{\sqrt[2]{3}_{a,j}}\right] \\ t_f &= MAX\left[t_{f,j}\right] \end{aligned}$$

We then imposed synchronization between joint motion by setting all $t_{f,j} = t_f$. The position of each joint will be defined as:

$$\begin{aligned} r_j(t) &= 10\left({}^{t}/_{t_f}\right)^3 - 15\left({}^{t}/_{t_f}\right)^4 + 6\left({}^{t}/_{t_f}\right)^5 \\ q(t) &= q_i + r_j(t) \cdot D_j \\ v_j(t) &= \dot{r}_j(t) \cdot D_j \\ a_j(t) &= \ddot{r}_j(t) \cdot D_j \end{aligned}$$

1.2. Validation

To validate our code we tested it on the model of a SCARA robot that has maximum articular speed and acceleration $k_v = \begin{bmatrix} 50^\circ & 40^\circ \end{bmatrix}^T deg/s$ and $k_a = \begin{bmatrix} 100^\circ & 60^\circ \end{bmatrix}^T deg/s^2$, and performs a motion between the points of the joint space $q^i = \begin{bmatrix} 100^\circ & 100^\circ \end{bmatrix}^T$ and $q^f = \begin{bmatrix} 6^\circ & 60^\circ \end{bmatrix}^T$. As visible in figure 1 the script does generate the desired trajectory.

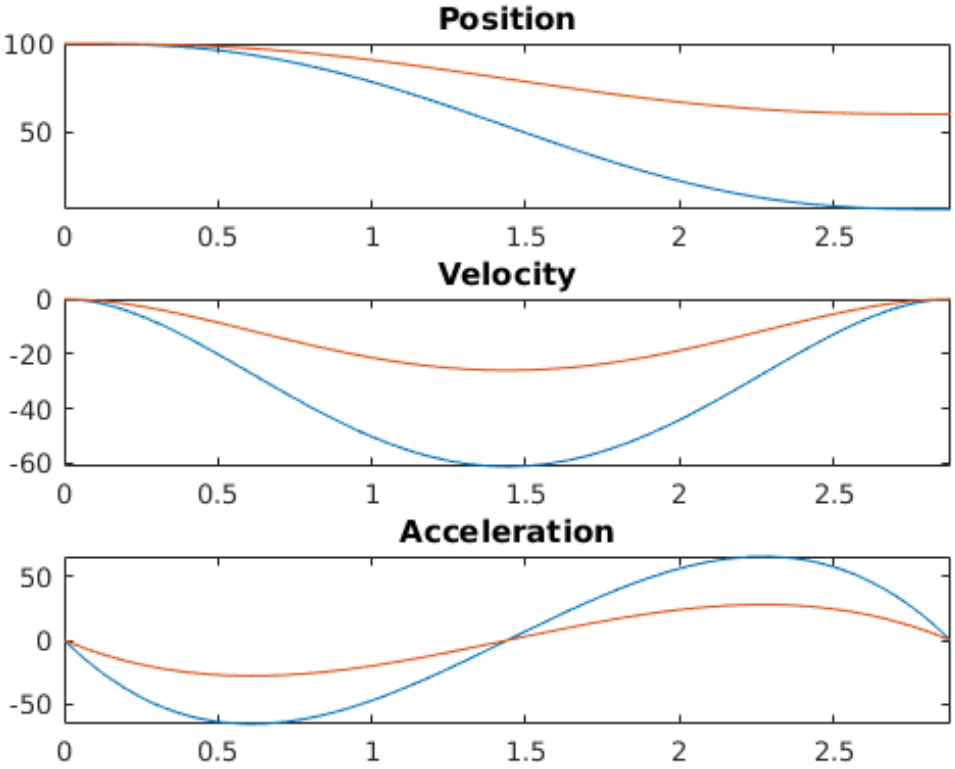

Figure 1: Motion between q^i and q^f.

1.3. Simulink Validation

A second testing took place using the Simulink tool, where using the file () we compared the output of the joint acceleration direct model with that entry to inverse dynamic model to validate the DDM and IDM models given.

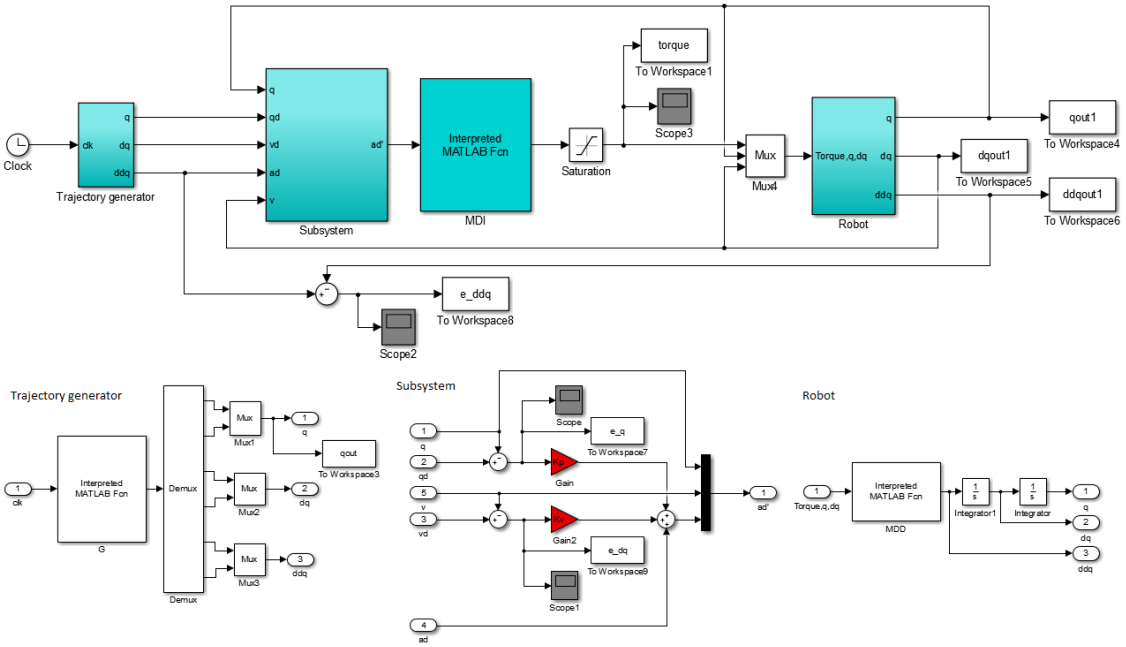

Figure 2: Simulink Model

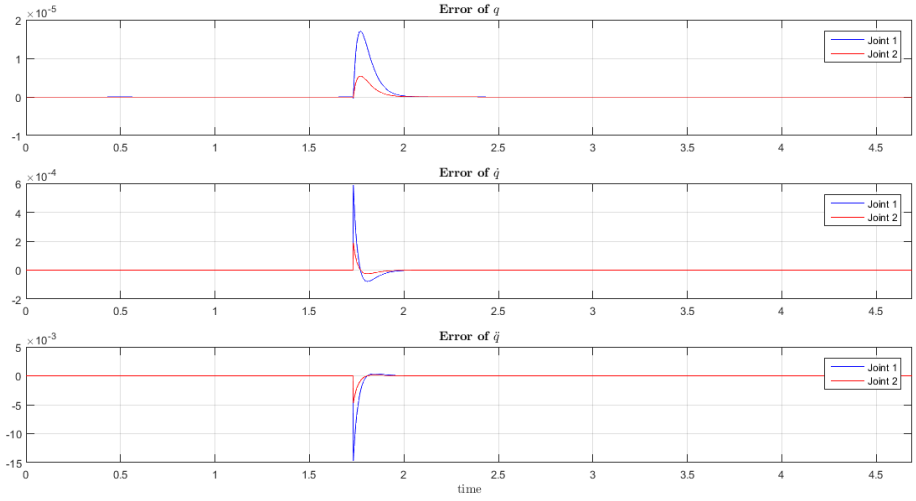

Figure 3: Validation through simulink Model

Since the error is very small, as seen in figure 3, our models are validated.

1.4. Multiple points interpolation

Finally we calculated the trajectory for multiple points,in this case we had three (q^1 , q^2 , q^3), using polynomial interpolation for two different cases:

1. The robot stops at the intermediate point.
2. The robot does not stop at the intermediate point.

we had to ensure the continuity in acceleration in the joint space for the 2 dof robot between all positions.

For the first case we used the function created before (polynomial of degree 5 as we have 6 constraints) applied first for points q1 and q2 and after for points q2 and q3. We considered a motion between these three points: $q^1 = \begin{bmatrix} 130^\circ & 130^\circ \end{bmatrix}^T$, $q^2 = \begin{bmatrix} 50^\circ & 90^\circ \end{bmatrix}^T$ and $q^3 = \begin{bmatrix} 6^\circ & 60^\circ \end{bmatrix}^T$.

In this case the minimum travel time can be obtained with the formula used before.
For this configuration we obtained the result shown in figure 4:

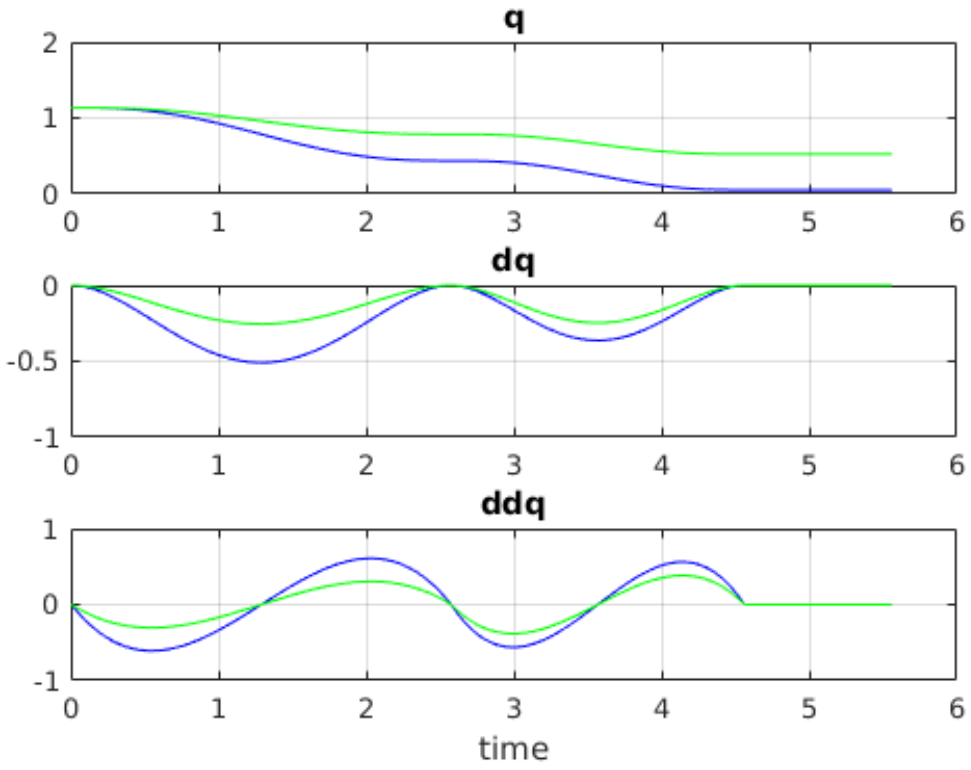

Figure 4: Motion with stop at intermediate point

In the case of non zero velocity at the intermediate point we have to add one more constraint, thus the polynomial becomes of degree 6. The generic equation is:

$$q(t) = a \cdot t^6 + b \cdot t^5 + c \cdot t^4 + d \cdot t^3 + e \cdot t^2 + f \cdot t + g$$

The additional constraint is:

$$q(t_{f1}) = q_2$$

By observing position, velocity and acceleration equations for time zero we quickly obtain that:

$$\begin{gathered} g = q_1 \\ f = 0 \\ e = 0 \end{gathered}$$

By using the solve function of matlab we solved the following system of equations:

$$\begin{aligned} eq_1 &: \quad 30 \cdot A \cdot t_f^4 + 20 \cdot B \cdot t_f^3 + 12 \cdot C \cdot t_f^2 = 0 \\ eq_2 &: \quad 6 \cdot A \cdot t_f^5 + 5 \cdot B \cdot t_f^4 + 4 \cdot C \cdot t_f^3 + 3 \cdot D \cdot t_f^2 = 0 \\ eq_3 &: \quad A \cdot t_f^6 + B \cdot t_f^5 + C \cdot t_f^4 + D \cdot t_f^3 + q_1 = q_3 \\ eq_4 &: \quad A \cdot t_{f,1}^6 + B \cdot t_{f,1}^5 + C \cdot t_{f,1}^4 + D \cdot t_{f,1}^3 + q_1 = q_2 \end{aligned}$$

obtaining the symbolic values for the coefficients in order to implement them in the motion planning function:

$$
\begin{aligned}
a &= -\frac{2\cdot\left(q_1\cdot t_f^5+9\cdot q_1\cdot t_{f,1}^5-q_2\cdot t_f^5-9\cdot q_3\cdot t_{f,1}^5-15\cdot q_1\cdot t_f\cdot t_{f,1}^4+15\cdot q_3\cdot t_f\cdot t_{f,1}^4+5\cdot q_1\cdot t_f^2\cdot t_{f,1}^3-5\cdot q_3\cdot t_f^2\cdot t_{f,1}^3\right)}{t_f^5\cdot t_{f,1}^3\cdot\left(t_f^3-3\cdot t_f\cdot t_{f,1}^2+2\cdot t_{f,1}^3\right)} \\
b &= +\frac{3\cdot\left(q_1\cdot t_f^6+6\cdot q_1\cdot t_{f,1}^6-q_2\cdot t_f^6-6\cdot q_3\cdot t_{f,1}^6-15\cdot q_1\cdot t_f^2\cdot t_{f,1}^4+8\cdot q_1\cdot t_f^3\cdot t_{f,1}^3+15\cdot q_3\cdot t_f^2\cdot t_{f,1}^4-8\cdot q_3\cdot t_f^3\cdot t_{f,1}^3\right)}{t_f^5\cdot t_{f,1}^3\cdot\left(t_f^3-3\cdot t_f\cdot t_{f,1}^2+2\cdot t_{f,1}^3\right)} \\
c &= -\frac{15\cdot(q_1-q_3)}{t_f^4} \\
d &= -\frac{q_1\cdot t_f^6-10\cdot q_1\cdot t_{f,1}^6-q_2\cdot t_f^6+10\cdot q_3\cdot t_{f,1}^6+24\cdot q_1\cdot t_f\cdot t_{f,1}^5-24\cdot q_3\cdot t_f\cdot t_{f,1}^5-15\cdot q_1\cdot t_f^2\cdot t_{f,1}^4+15\cdot q_3\cdot t_f^2\cdot t_{f,1}^4}{t_f^3\cdot t_{f,1}^3\cdot\left(t_f^3-3\cdot t_f\cdot t_{f,1}^2+2\cdot t_{f,1}^3\right)}
\end{aligned}
$$

At this point the we used the tf obtained in the zero velocity case to obtain a feasible trajectory and them we tried to minimize it until kv or ka were reached. The result obtained are the following:

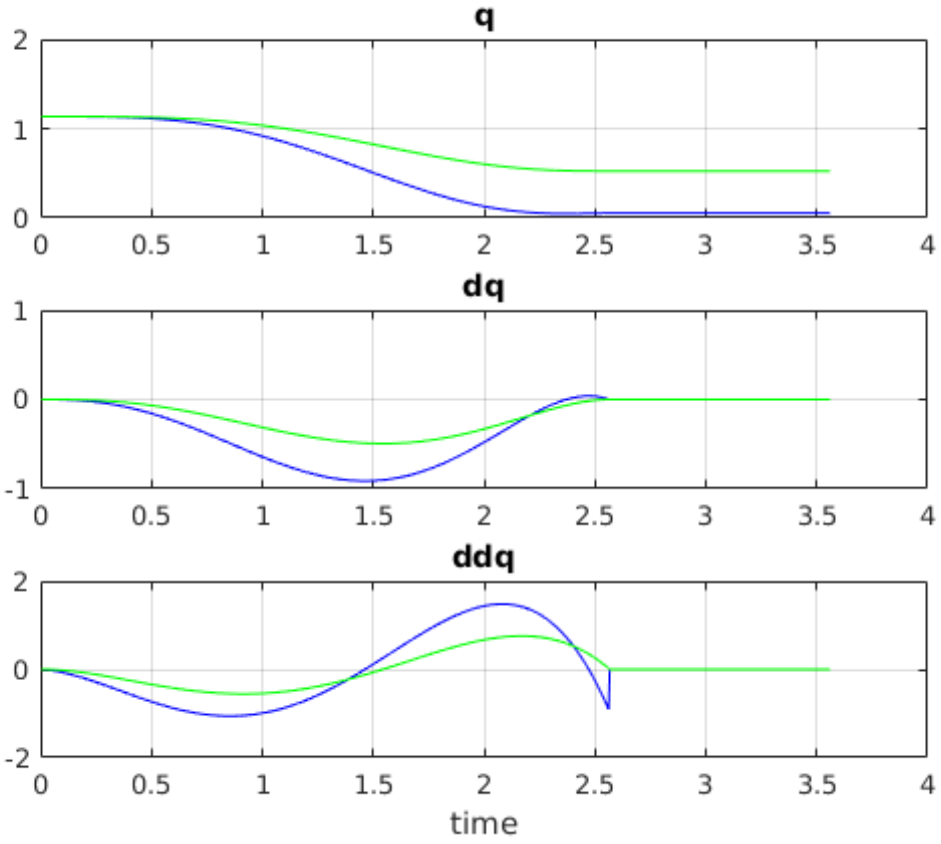

Figure 5: Motion without stop at intermediate point

As can be seen in figure 5 the minimum travel time is decreased although we get a strange behavior for the acceleration in q_3 and in this case kv and ka are never reached, so the algorithm can be improved and tested on a wider range of scenarios.

2. Control

2.1. PID Control

We implemented a dynamic control in joint space through PID control law. For this model we assumed that the joint variables and velocities are obtained from the dynamic model without taking into account respectively the effect of discretization of position sensors and the numerical differentiation or filtering. The system is considered as a continuous time system without using a zero holder for the input torques. Taking as true the previous statements we can summarize the control law as:

$$\underline{\Gamma} = K_p \cdot \left(\underline{q^d} - \underline{q}\right) - K_v \cdot \left(\underline{\dot{q}^d} - \underline{\dot{q}}\right) + K_r \cdot \int_{t_0}^{t} \left(\underline{q^d} - \underline{q}\right) \cdot d\tau$$

where q^d is the desired position of each joint, that is calculated by the trajectory planner and has been described in the previous sections of the paper. Differently K_p, K_v and K_r can be calculated as:

$$\begin{aligned} \omega &= BW/2 \\ F_v &= \begin{bmatrix} F_{V,1} & F_{V,2} \end{bmatrix} \\ K_p &= 6a \cdot \omega^2 \\ K_v &= 3a \cdot \omega - F_v \\ K_r &= a \cdot \omega^3 \end{aligned}$$

where a can be obtained from the dynamic model, while the bandwidth BW, $F_{V,1}$ and $F_{V,2}$ were given.

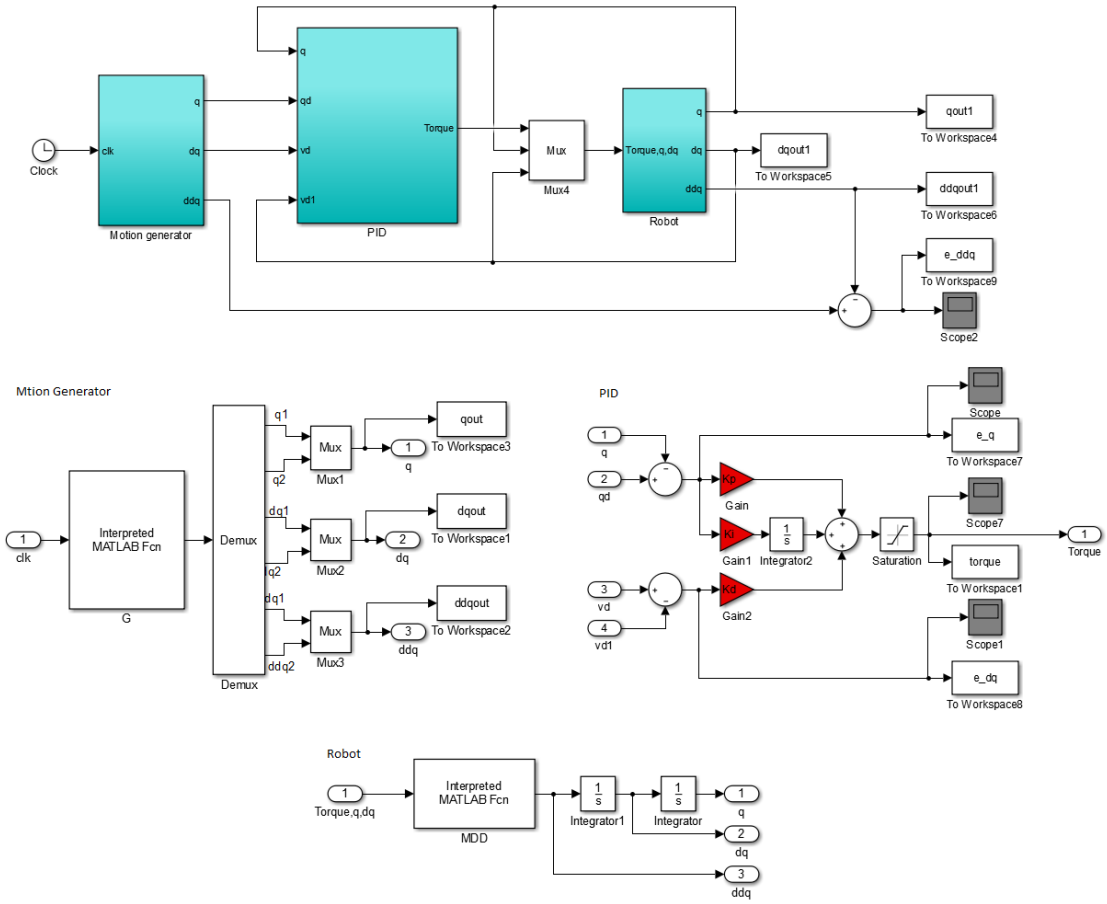

Figure 6: PID control law Simulink Model

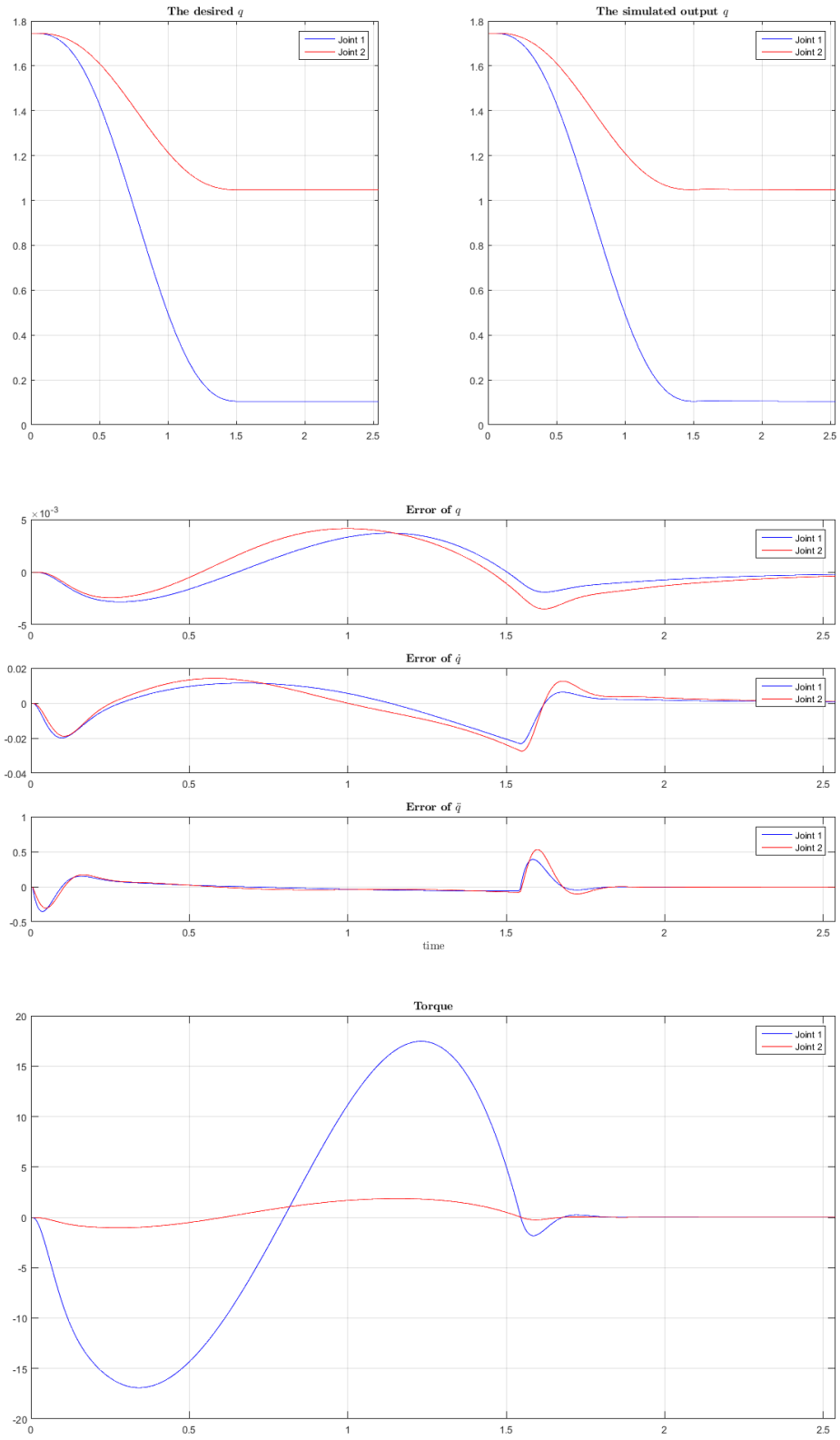

Figure 7: Performance of the PID simulink Model

2.2. Torque Control

We then implemented a dynamic control in joint space through torque control law. For this model we assumed that the joint variables and velocities are obtained from the dynamic model without taking into account respectively the effect of discretization of position sensors and the numerical differentiation or filtering. The system is considered as continuous time system without using a zero holder for the input torques. Taking as true the previous statement we can summarize the control law as:

$$\underline{\ddot{q}} = \underline{\ddot{q}^d} + K_v \cdot \left(\underline{\dot{q}^d} - \underline{\dot{q}}\right) + K_p \cdot \left(\underline{q^d} - \underline{q}\right)$$

where q^d is the desired position of each joint, that is calculated by the trajectory planner and has been described in the previous sections of the paper. Differently K_p, and K_v can be calculated as:

$$\begin{aligned} \omega &= BW/2 \\ F_v &= \begin{bmatrix} F_{V,1} & F_{V,2} \end{bmatrix} \\ K_p &= \omega^2 \\ K_v &= 2 \cdot \xi \cdot \omega \end{aligned}$$

where ξ can be be assumed to be equal to 1, while the bandwidth BW, $F_{V,1}$ and $F_{V,2}$ were given. Through this calculations we obtained the compensated acceleration that will be easily converted to the desired torque though the direct dynamical model. As visible from figure 9 the error on the joint position is very small so it satisfies our needs. Moreover the error on velocity and acceleration is also small so this control provides high performance for high speed tasks.

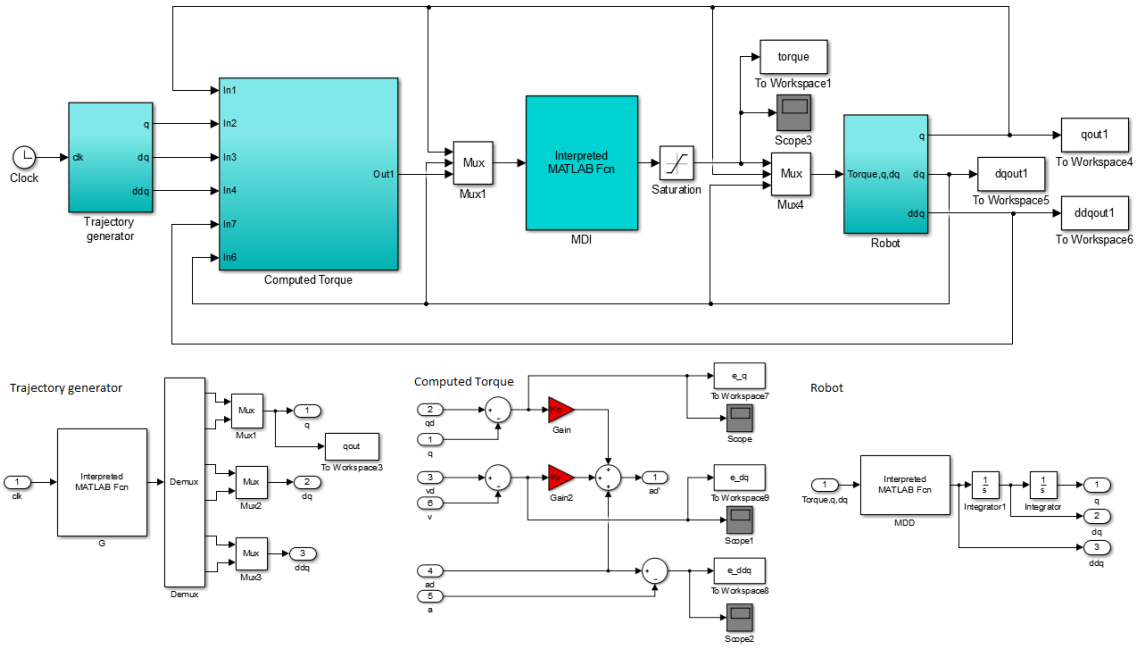

Figure 8: Torque control law Simulink Model

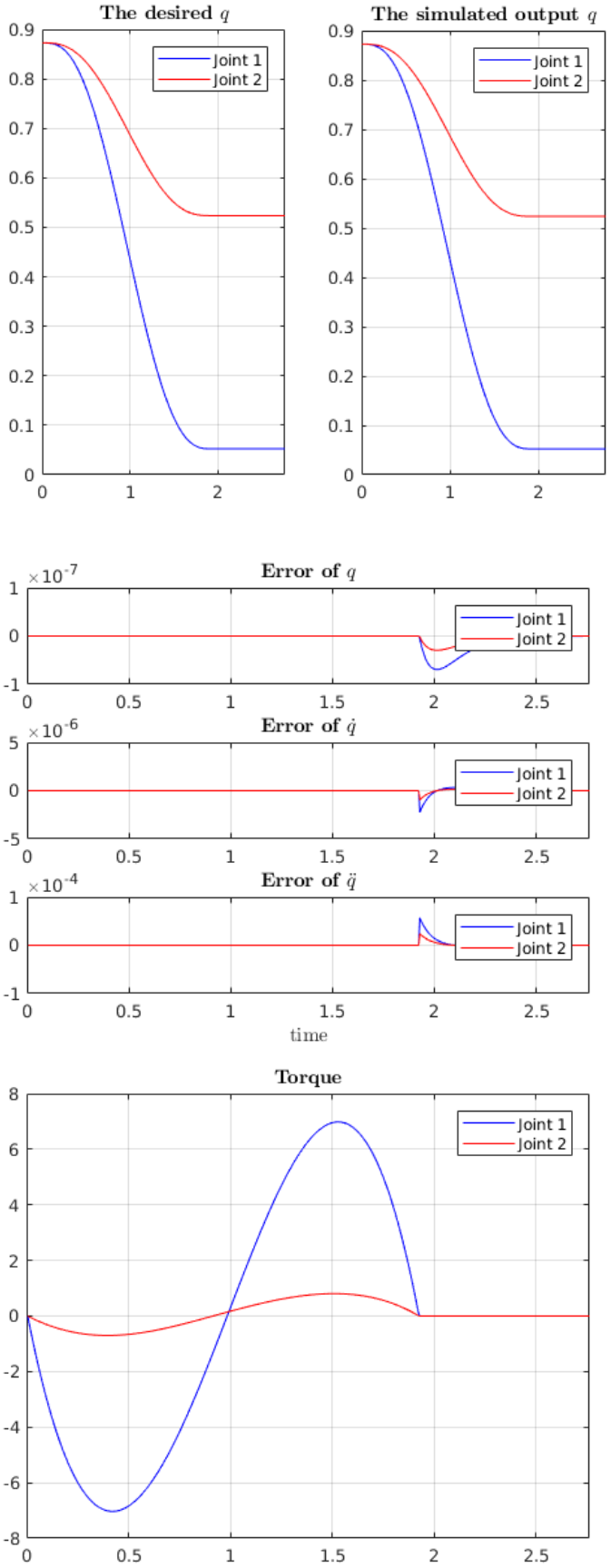

Figure 9: Performance of the torque control law simulink Model

Conclusions

For this paper we implemented the trajectory generator and several control laws for the SCARA robot. In section 1.1 to 1.3 we first implemented the trajectory generator and later we validated that model using both MATLAB and Simulink.
For section 1.4 while it was easy to implement the motion planning in the zero-velocity case, a longer computation was requested for the second case. This could not be suitable for real time application, thus the algorithm has to be improved, either through a faster minimization of t_f or by finding its analytical expression and implement it in the code.
Finally in section2.1 and 2.2 we implemented the PID and the computed torque control laws and we plotted the desired trajectory, the simulated output, the position tracking error ($e = q^d - q$) and the joint torques for each one of them. As visible from fig. 7 and 9 for both the control laws the absolute joint position tracking error is always much lower than the required 0.01 so they both satisfy our needs. If we compare the two control laws we can notice that the PID is easier to implement and requires less computing time since it has no need for real time computation, the torque control guarantees higher performance since it is like having a double integrator for each joint with a derivative and proportional behavior, thus we eliminate the steady state error (thanks to the integrator) and the cumulation of errors since we don't have a second oder system. Moreover the model is optimal for every configuration. In fact we compare the quality of the outputs, we can notice that all PID errors are much higher than the ones of the computed torque (roughly 10^4 times higher) and all the PID plots also show overshoots that are not present when using the other control law.