Real-time trajectory scaling for robot manipulators

Perugini Patrizio [a] Castelmare Mattia [b]
[a] La Sapienza, University of Rome
[b] La Sapienza, University of Rome

Robotics II project

Abstract

The aim of this project is to show the effectiveness of a feedback trajectory scaling approach which improves path performances and is able to recover the delay introduced by the speed modulation. We tested this method on two different trajectories performed by a 3R robot by successively scaling time, going from a time in which no scaling occurs to a more demanding case in which the control scheme activates.

1 Introduction

Online trajectory scaling methods deform the original timing law to satisfy the robot constraints. The input of the algorithm is the desired motion and the output is the scaled trajectory. Rather then changing the whole trajectory we keep the original path fixed and the control scheme modifies the velocity and acceleration profiles of the robot's joints. The final command for the actuators is the scaled velocity.

We will now introduce the control scheme that we used in the simulations. The control law is made up of two fundamental elements: delay recovering control loop that is an external control loop where the control variable is the timing law γ and a saturation fly-wheel block that is an inner integral control loop needed to improve path-following performances of the algorithm. The input of the scheme is γ_{ref} that is the reference timing law of the trajectory, therefore the control error is $e_\gamma = \gamma_{ref} - \gamma$. As shown in Figure 1 $\dot{\gamma}_{ref}$ acts as a feedfoward action: when the scaling is not activated it is the only input, otherwise the external loop increases the control action to recover the error. The other term that contributes to the control input is u_f. This term is obtained as follows: $\mathrm{u}_f = K_f \int_0^{t_0} (\dot{\gamma} - u)\, dt$.

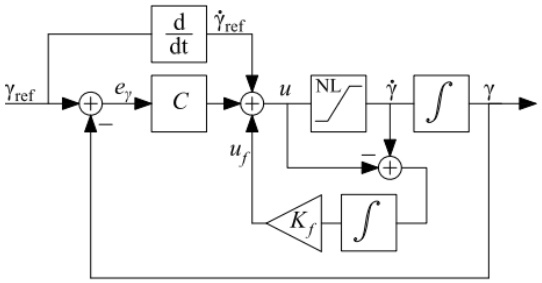

Figure 1: Control scheme

This inner control loop stabilizes $\dot{\gamma}_{ref}$ to a bounded value derived from the saturation of the robot's joint velocities along the desired trajectory. It's possible to appreciate that when the control input u is within the linear part of the non-linear saturation block the saturation fly-wheel block is zero while it behaves as a saturation block when the optimal control problem would exceed the bounds and activates the scaling method. In particular at each cycle we solve an online problem in which we check whether the bounds in velocity and acceleration at the following state are satisfied. If they are not then the speed is scaled by a factor k chosen as $K = max(1, K_{vel}, \sqrt{K_{acc}})$ where $K_{vel} = max(1, |\dot{q}_1|/\dot{q}_{max,1}, |\dot{q}_2|/\dot{q}_{max,2}, |\dot{q}_3|/\dot{q}_{max,3})$,
$K_{acc} = max(1, |\ddot{q}_1|/\ddot{q}_{max,1}, |\ddot{q}_2|/\ddot{q}_{max,2}, |\ddot{q}_3|/\ddot{q}_{max,3})$.
The addition of this control loop helps to keep the value of $\dot{\gamma}_{ref}$ close to $\dot{\gamma}$ since u_f would be a negative number if u is greater then the saturation value. In order to recover the delay accumulated because of the saturation during the slowdown phase, the commanded velocities would be higher then the reference one. The higher is the value of K_f the faster $\dot{\gamma}$ tracks the desired path parameterized by γ.
The entire formula is $u = \dot{\gamma}_{ref} + C(e_\gamma) + K_f \int_0^{t_0} (\dot{\gamma} - u)dt$.

2 Kinematics

In this first part we assume a kinematic model of the 3R robot commanded via joint velocity to carry out the simulations.

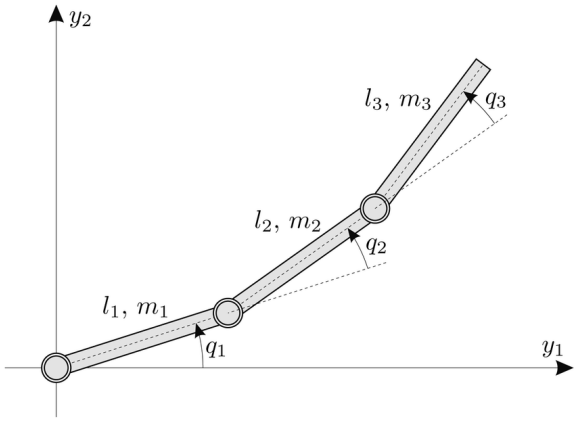

Figure 2: 3R robot

In this setting we consider an ideal low level controller which directly executes the commanded velocities.

Here follows the DH table that we used to compute the direct kinematics:

i	α	a	d	θ
1	0	L_1	0	q_1
2	0	L_2	0	q_2
3	0	L_3	0	q_3

We'll now go through the main calculations needed to find the velocities that have to be commanded. Starting from the DH-Table the direct kinematic of the robot can be easily found.

Let f(q) be the direct kinematics of the robot and J(q) the associated Jacobian, then $\dot{q}(t) = J^{-1}(q(t)) * (p'(\gamma(t)) * \dot{\gamma}(t))$. Where $p'(\gamma(t))$ is the derivative of the parametric representation of the path with respect to the timing law $\gamma(t)$ and $\dot{\gamma}(t)$ the derivative of γ with respect to time.

The values of q(t) are obtained from the inverse kinematic of a 3R robot starting from knowing $p(\gamma(t))$ (the position of the end-effector).
We have chosen 2 trajectories that involve only pointing and orientation in the plane in order to avoid kinematic redundancy of our robot.

2.1 Circular trajectory

The first trajectory that we used for testing our control law is a circular trajectory parameterized as follows:

$$p(\gamma) = \begin{pmatrix} X_0 + Rcos(2\pi\gamma) \\ Y_0 + Rsin(2\pi\gamma) \\ \pi/4 \end{pmatrix} \gamma(\tau) \in [0,1]$$

In our example we chose γ as a quintic polynomial. The general parameterization is $\gamma(\tau) = a\tau^5 + b\tau^4 + c\tau^3 + d\tau^2 + e\tau + f$ with τ = t / T (where T is the total time of the simulation). We imposed a rest to rest motion (initial and final velocities and accelerations are zero) and this led to this particular solution:
$\gamma(\tau) = \gamma_{in} + \Delta\gamma(6\tau^5 - 15\tau^4 + 10\tau^3)$ with $\Delta\gamma = \gamma_{fin} - \gamma_{in}$.
$\dot{\gamma}(t) = (30t^2)/T^3 - (60t^3)/T^4 + (30t^4)/T^5$.
$\ddot{\gamma}(t) = (60t)/T^3 - (180t^2)/T^4 + (120t^3)/T^5$.

Experiments

From now on the following parameters will be taken into account: $X_0 = 1m, Y_0 = 1.5m$, $R = 0.5m, L_1 = L_2 = L_3 = 1m$.
In order to appreciate the results we will take into account 2 different values for T (that is the duration of the trial). In the first case the robot will be able to execute the path because no joint saturates while the scaling mechanism will come into place in the other example.

Now, We must define joint limits for the velocities and accelerations.

Velocity boundaries:

$-4rad/s <= \dot{q_1} <= 4rad/s,$

$-4rad/s <= \dot{q_2} <= 4rad/s,$

$-4.3rad/s <= \dot{q_3} <= 4.3rad/s.$

Acceleration boundaries:

$-25rad/s^2 <= \ddot{q_1} <= 25rad/s^2,$

$-15rad/s^2 <= \ddot{q_2} <= 15rad/s^2,$

$-20rad/s^2 <= \ddot{q_3} <= 20rad/s^2.$

Moreover in the formula $u = \dot{\gamma}_{ref} + C(e_\gamma) + K_f \int_0^{t_0}(\dot{\gamma} - u)dt$ we set the gain as $C = 0.1$ and $K_f = 100$.

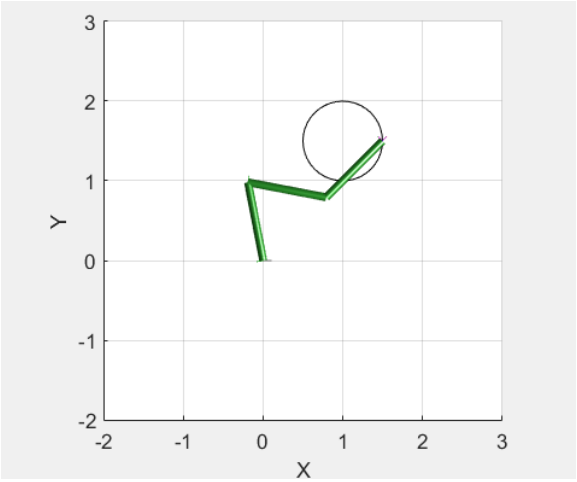

Figure 3: Circular trajectory

In the Figure 3 is shown the nominal trajectory to be performed.

Starting from now we will compare the reference positions, velocities and accelerations with respect to the one executed in order to explain the online scaling process.

T=3.9s

With this time we are perfectly able to perform the nominal trajectory without the need of the scaling because all the boundaries are respected. There will be no error at all not in position nor in velocity nor in acceleration.

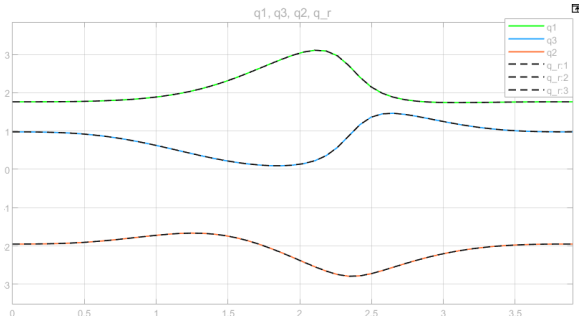

Figure 4: $q(t)$ T = 3.9s

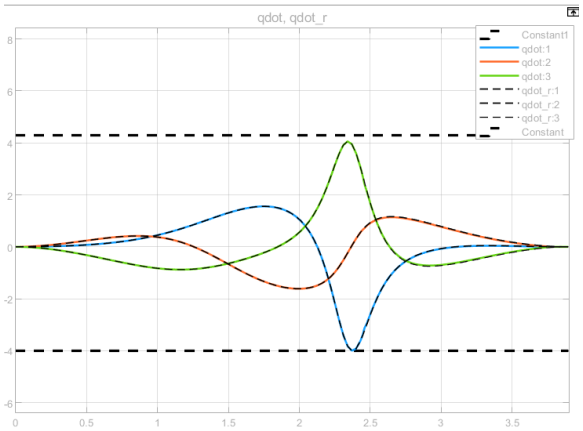

Figure 5: $\dot{q}(t)$ T = 3.9s

We can easily see that the position of the executed q is exactly the same of the nominal ones and that the velocities never exceed the bounds. It's important also to consider what happens to the acceleration. We don't expect the acceleration to scale in this case since the time requested for the motion it's not demanding.

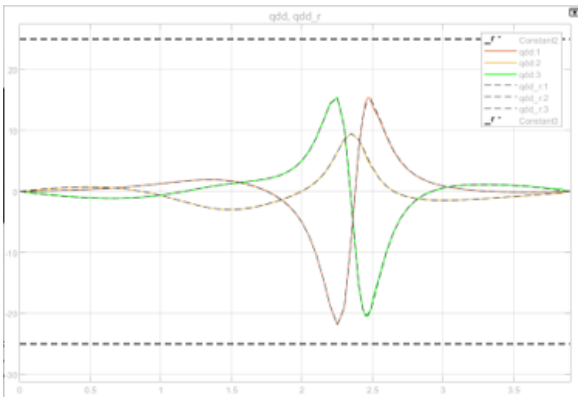

Figure 6: $\ddot{q}(t)$ T = 3.9s

T=3.5s

For the sake of clarity we will not show all the joints, in order to explain the scaling in detail and we'll take into account only the more interesting one. The nominal motion is represented in the following graphs by the dotted lines while the executed one with the other colored lines.

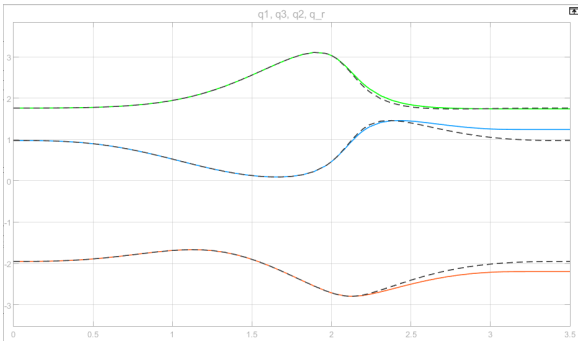

Figure 7: q(t) T = 3.5s

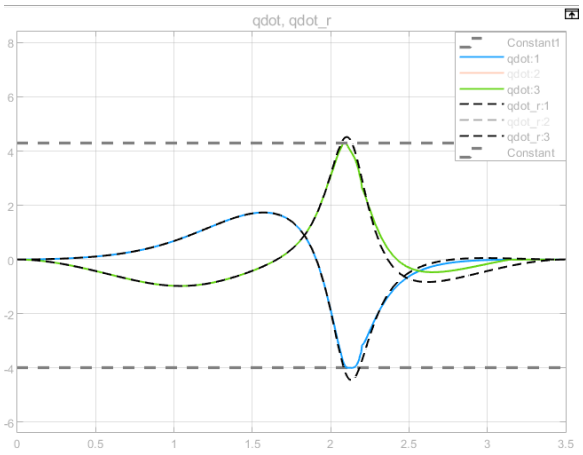

Figure 8: $\dot{q}(t)$ T = 3.5s

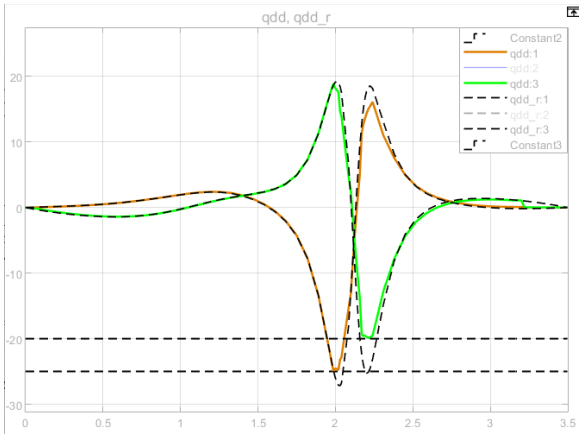

Figure 9: $\ddot{q}(t)$ T = 3.5s

In this case we can see the activation of the online scaling. Figure 8 clearly shows how joint 3 and 1 would saturate and here the control law acts. It modifies the velocity profiles when the nominal velocity goes beyond the limit bringing the actual one at saturation, this will cause an overall slowdown in the execution of the path. When the slowdown phase ends, the saturation fly-wheel block recovers the delay by accelerating and exceeding the nominal velocity.

Figure 7 shows a noticeable error in the joints position at the end of the simulation. This is true but the trajectory is always followed, the error is only in the time and not in space. The robot tries its best to execute the desired trajectory while staying always in the path in given total time.

Figure 10 shows the final position of the robot (and so the error in space) which is achieved without never going out of the desired Cartesian path.

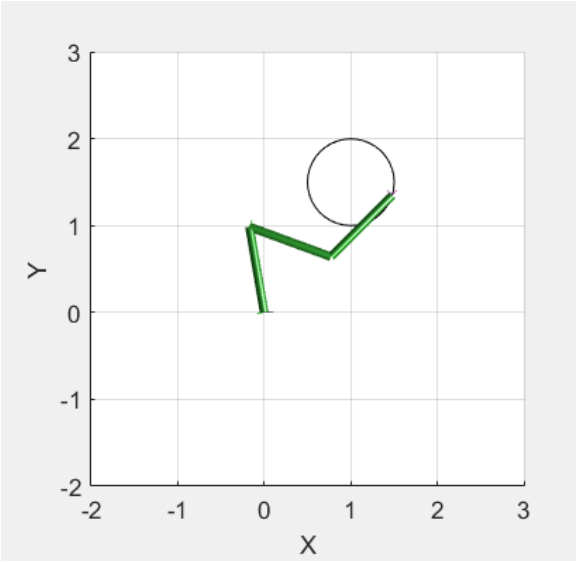

Figure 10: position error T = 3.5s

It is possible to close the path if we decide to increase the time of the experiment. In particular we can execute the trajectory in the nominal time and then continue for a time Δt until the end of the path is reached. After the nominal time T we have $\gamma_{ref} = 1$ and $\dot{\gamma}_{ref} = 0$ as reference, in this way our robot will proceed at a velocity for another Δt that allows it to reach the desired final position.

Here follows an example of what would happen to the q(t).

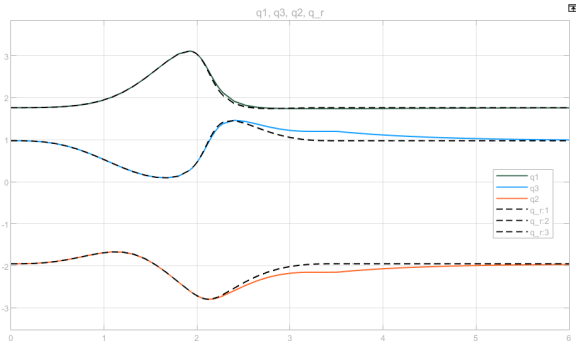

Figure 11: q(t) T = 6s

2.2 Sinusoidal trajectory

The other trajectory that we chose for testing our control law is a sinusoidal trajectory parameterized in the following way:

$$p(\gamma) = \begin{pmatrix} \gamma + 1 \\ 1 + sin(4\pi\gamma) \\ \gamma \end{pmatrix} \gamma(\tau) \in [0,1].$$

Also in this case we chose γ as a quintic polynomial to impose a rest to rest motion with zero initial and final acceleration.

Experiments

As we have done before we consider the length of the links as follow $L_1 = L_2 = L_3 = 1m$ and the same boundaries of the accelerations and velocities.

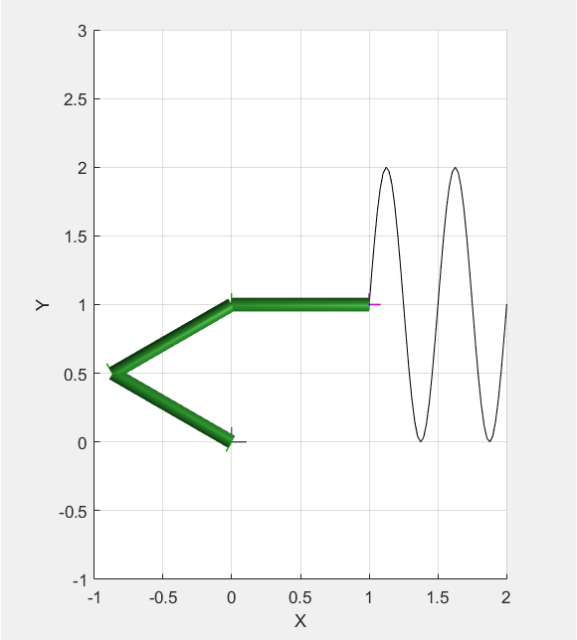

Figure 12: Sinusoidal trajectory

In order to see the activation of the scaling due to the control law we are going to present two different cases: in the first one we ask the robot to perform the path in a time in which there is not saturation of the joint velocities and accelerations, so it follows the sinusoidal trajectory as it would in an ideal case. In the second case we will decrease the simulation time to appreciate the scaling mechanism.

T = 12.7s

At this execution time we can observe that the scaling doesn't intervene and the positions, accelerations and velocities respect the boundaries and have the same profiles of the nominal ones.

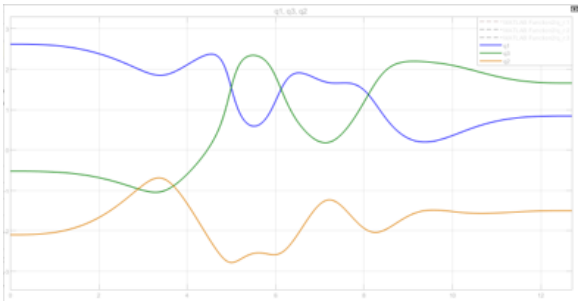

Figure 13: q(t) T = 12.7s

As we can see from the graphs below due to the not activation of the scaling we don't have any error in the position, velocity and acceleration.

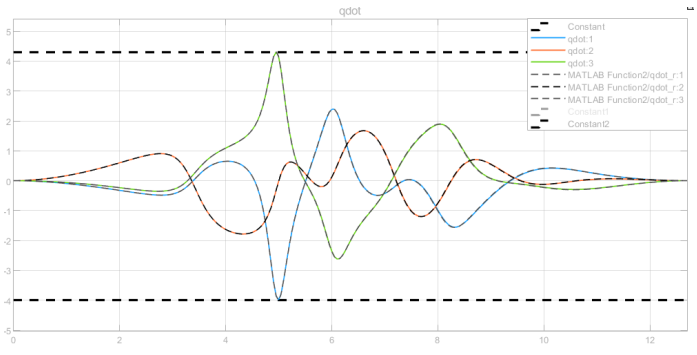

Figure 14: $\dot{q}(t)$ T = 12.7s

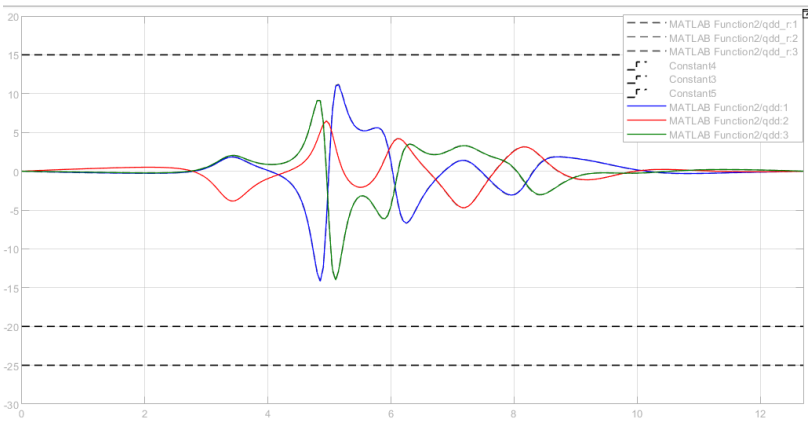

Figure 15: $\ddot{q}(t)$ T = 12.7s

T = 12s

We have shrunk the execution time in order to actually see the different profiles and the effect of the scaling algorithm.

Also in this case, as Figure 16 shows, due to the shorter time of the duration of the experiment we obtain at the end of the simulation a noticeable error in the positions, but as we already have proved this error is only in time and not in space.

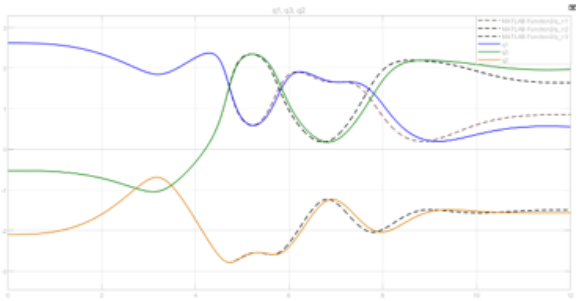

Figure 16: q T = 12s

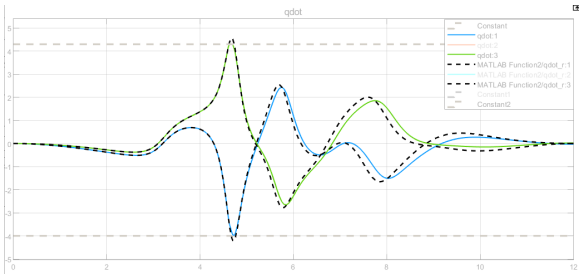

Figure 17: $\dot{q}$ T = 12s

To not have confusing graphs, we have only reported the velocities that reach saturation. In Figure 17 we can clearly see the effectiveness of the scaling: the dotted profiles are ideal velocities but they don't respect the bounds imposed. The green and blue ones, instead, are the real velocities executed by our controllers, they saturate and respect the bounds. Moreover the scaled velocities slow down when they cannot follow the desired velocity and then they recover the delay when the slow down phase ends by speeding up.

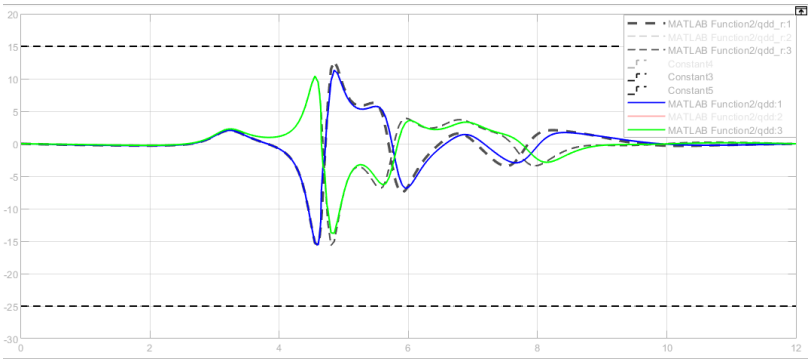

Figure 18: $\ddot{q}$ T = 12s

Also in the acceleration profiles we can recognize the same scaling mechanism just described.

3 Dynamics

In this new section we are going to consider also the dynamic model of our 3R robot. In addition to saturation of the velocities and accelerations we will consider the saturation for the torques that now are the commanded input of the robot. This means that we are adding more non-idealities since we don't assume to have a low level controller which is able to perform the requested velocity. We have built two different control problems: the first is the one previously explained where the scaling happens, the second is the new controller which handles the dynamic. The output of the first problem $q, \dot{q}$ and $\ddot{q}$ (the scaled positions, velocities and accelerations) are now the input of the new controller which is built in cascade. The aim is to see if the robot still manages to perform the scaled profiles with the torque bounds imposed and with a suitable control law.

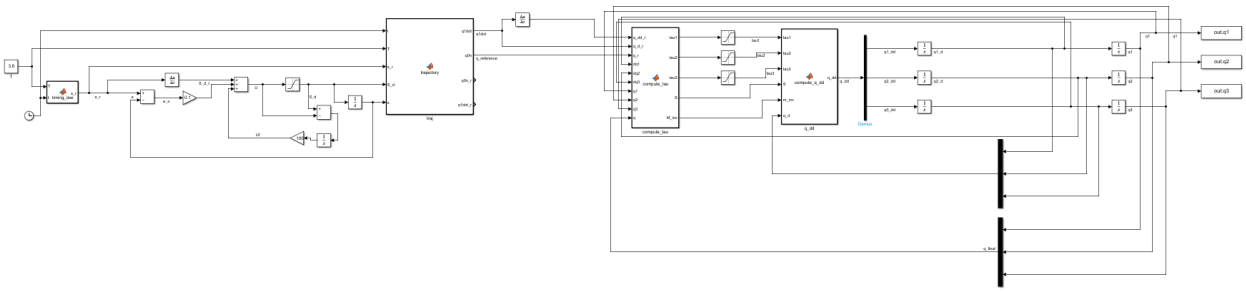

Figure 19: Low level controller + high level controller

In order to calculate the torque we used the following trajectory controller:
$\tau = M(q)(\ddot{q}_r + K_p e + K_d \dot{e}) + S(q, \dot{q})\dot{q}_r$, with $e = q_r - q$ and $\dot{e} = \dot{q}_r - \dot{q}$.
Going through the above formula we have:
- $M(q)$ which is the inertia matrix of the robot calculated in the actual configuration,
- $S(q, \dot{q})$ is the factorization of $c(q, \dot{q})$ in which there are the Coriolis and Centrifugal terms,
- K_p and K_d are respectively the gain associated to the error in position and to the error in velocity,
- $\dot{q}_r, \ddot{q}_r$ are the reference accelerations and velocities.

To compute $M(q)$ and $S(q, \dot{q})$ we used the following dynamic parameters:

i	m	dc	I
1	m_1	$L_1/2$	$(m_1 * L_1^2)/12$
2	m_2	$L_2/2$	$(m_2 * L_2^2)/12$
3	m_3	$L_3/2$	$(m_3 * L_3^2)/12$

Where $m_1 = m_2 = m_3 = 5Kg$ and $L_1 = L_2 = L_3 = 1m$. I represents the inertia moment of the links with respect the z axis and dc represents the CoM's distance of each link from the joint.

3.1 Circular trajectory with robot dynamics

All the parameters remain the same as before but we have to define the torque boundaries:
$-90Nm <= \tau_1 <= 90Nm$,
$-60Nm <= \tau_2 <= 60Nm$,
$-15Nm <= \tau_3 <= 15Nm$.
The gains K_p and K_d are set as follow: $K_p = \begin{pmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.1 \end{pmatrix}$ $K_d = \begin{pmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 10 \end{pmatrix}$.

When the time is not demanding and the torques don't saturate then we are able to execute the desired trajectory since it behaves as a double integrator.
The interesting part to analyze is when the torques saturate: in this case the control scheme comes into place, so we have analyzed the case in which the execution time is T=3.7s. The dotted lines are the scaled profiles (now the nominal one) and the other ones are the one executed by the robot. In Figure 21 we showed also the nominal joints position of the previous case (low level controller).

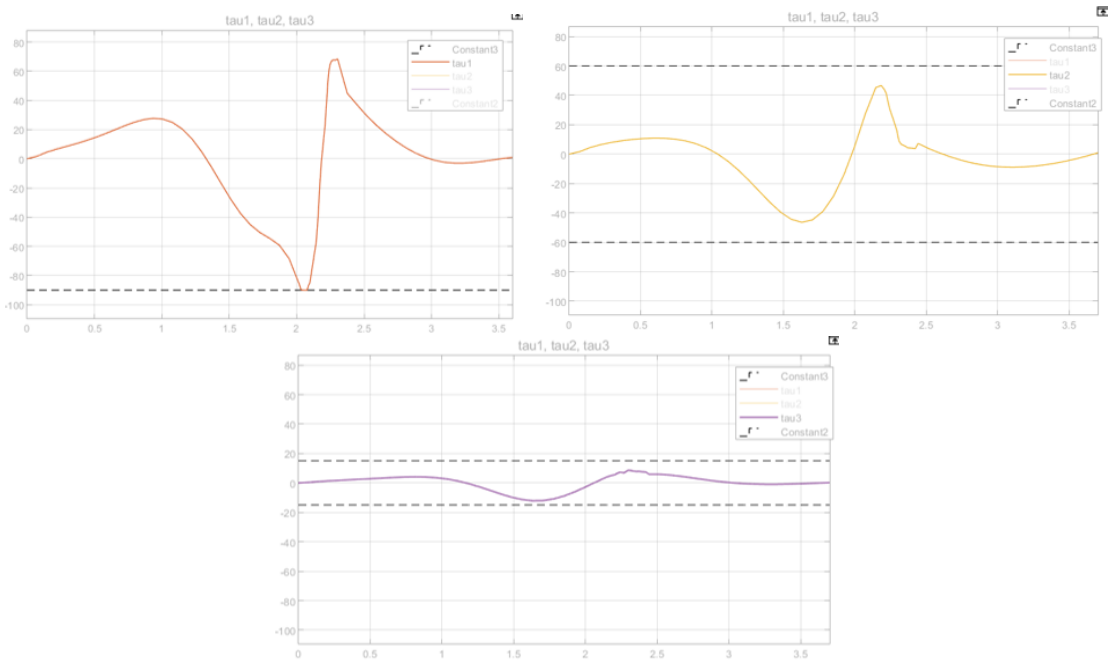

Figure 20: τ_1, τ_2, τ_3 T=3.7s

It's possible to appreciate from the above graphs how the torques saturate and so this lead to an increased error in position, velocity and acceleration. We can notice that the new profiles are not so different from the reference ones. We'd like to remember that the reference are different from the original, the former are the ones in the ideal case while the latter are the ones when the scaling mechanism intervenes.

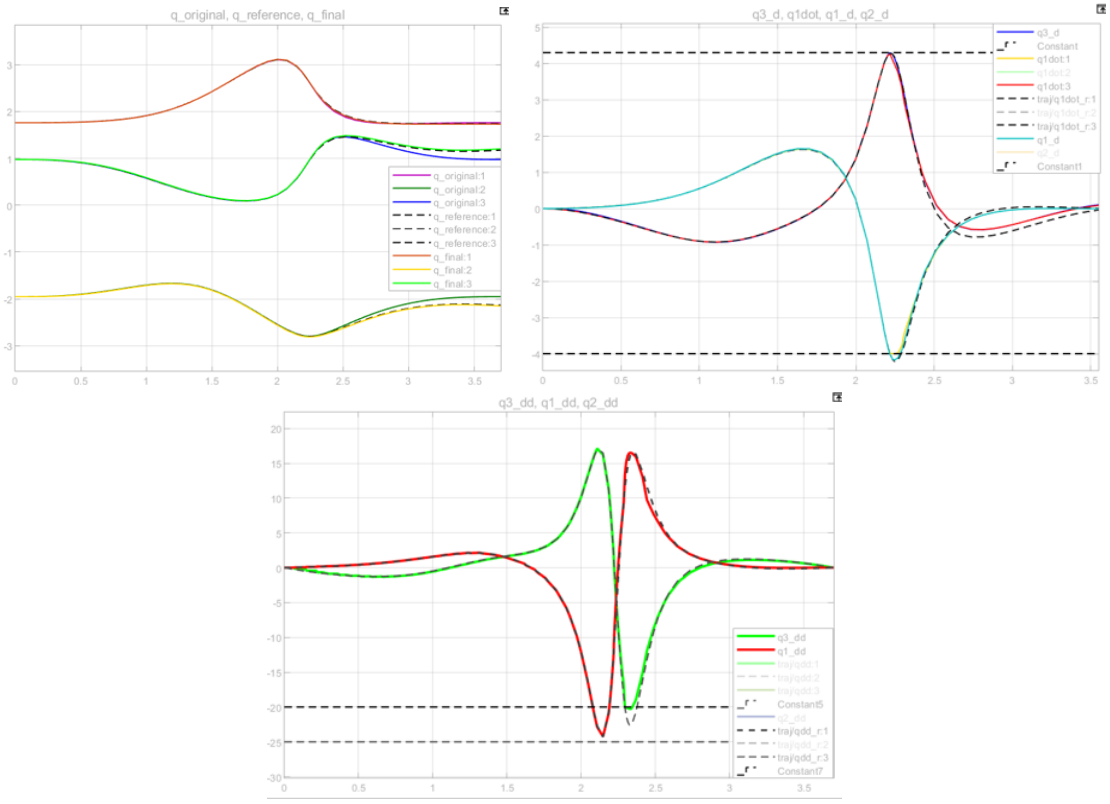

Figure 21: $q(t)$ $\dot{q}(t)$ $\ddot{q}(t)$ T = 3.7s

3.2 Sinusoidal trajectory with robot dynamics

In this case the settings are the same as the previous example but for didactic purpose we reduced the bounds of the torque as follow:

$-45Nm <= \tau_1 <= 45Nm,$

$-30Nm <= \tau_2 <= 30Nm,$

$-15Nm <= \tau_3 <= 15Nm.$

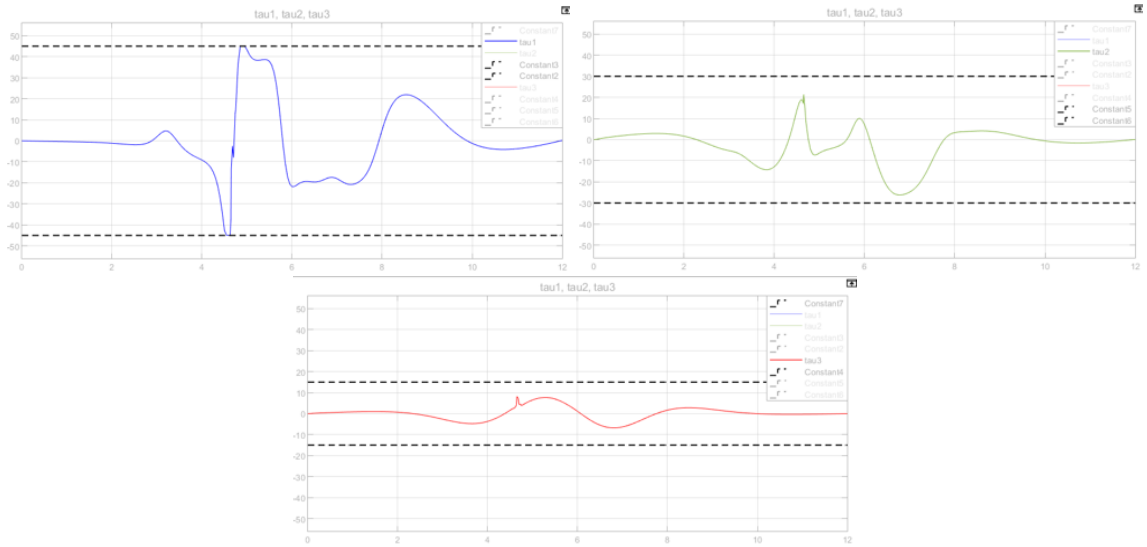

Figure 22: τ_1, τ_2, τ_3 T=12s

Also in this case when we do the simulations with a demanding time, the torques saturate but the controller can still execute the path within the acceleration and velocity bounds.

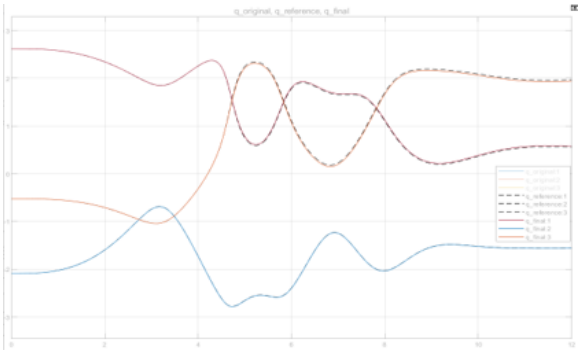

Figure 23: $q(t)$ T = 12s

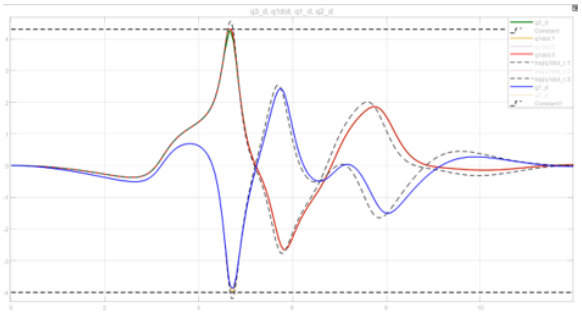

Figure 24: $\dot{q}(t)$ T = 12s

As shown in the graphs, also in this case the robot is able to perform the path while changing the velocity and acceleration profiles according to the boundaries imposed and by recovering the delay introduced in the slowdown phase.

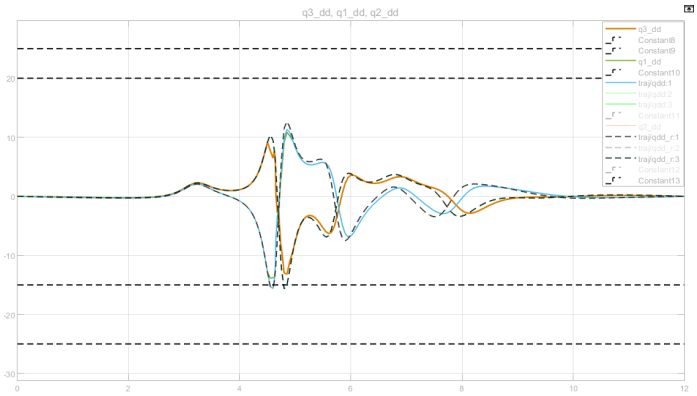

Figure 25: $\ddot{q}(t)$ T = 12s

4 Conclusions

We have proved the effectiveness of the online scaling algorithm. In the case of an ideal controller once the scaling happens the control law is able to scale the velocity and acceleration profiles and there is no error in the cartesian space, since the only error that we have is in time. By extending the time of the simulation we can easily close the path. Finally, once we consider also the dynamic model and the saturation of the torques more non idealities come into place and the error increases if the time is very demanding. We managed to mitigate this problem by choosing a feedback linearization + PD + FF as a control law that allows us to replicate the results previously obtained.

References

[1] M. Faroni, R. Pagani, G. Legnani, "Real-time trajectory scaling for robot manipulators," Proc. 17th Int. Conf. on Ubiquitous Robotics, pp. 533-539, 2020 (with video)