Online trajectory scaling for robot manipulators

- Patrizio Perugini 1844358
- Mattia Castelmare 1815675
- University ‘La Sapienza’

- Master degree in ‘Artificial Intelligence and Robotics’
- a.y. 2021 / 2022
- Robotics2 final project

Online trajectory scaling methods

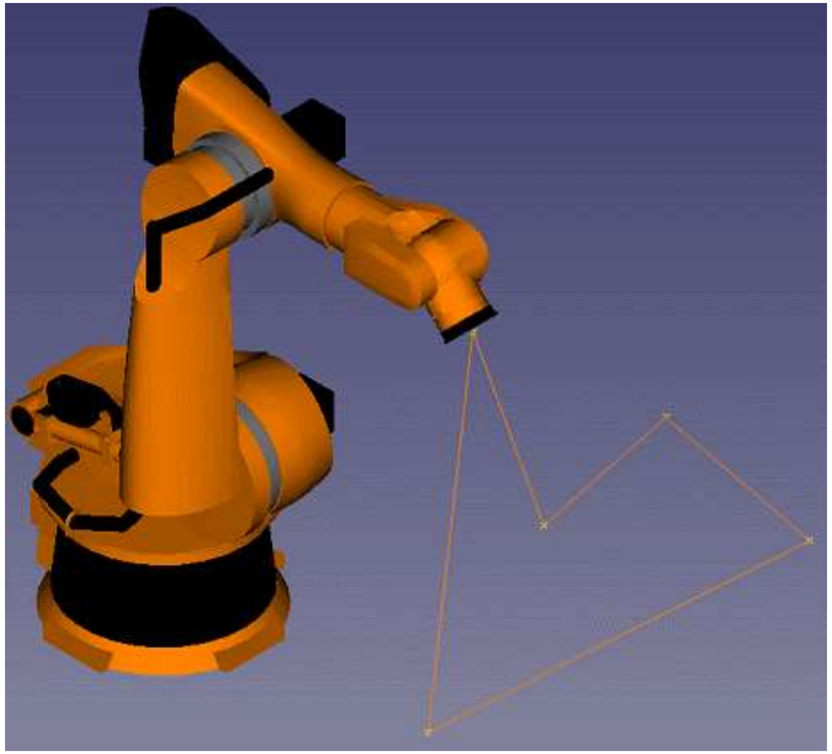
- Robots need to be endowed with some degree of autonomy to adapt their behavior to the state of the environment and other agents.

- It is a fundamental recquirements, for example in HRC applications (human robot collaboration), of the robot to **change** its **motion** at **runtime** (avoid collision and maximize process throughput).

- **Online modification of motion** can be performed mainly in 2 different ways:

  1) **Modifying** the **whole trajectory** that the robot has to follow (more complex: it requires to run path-planning algorithms and to check for collisions at runtime).
  2) **Modifying** only the **velocity profile** but keeping the original path (preferred: it requires only to slow down the execution of the trajectory along the same path).

- **Online trajectory scaling methods** deform the **original timing law** to satisfy the robot joint limits.



Online trajectory scaling methods

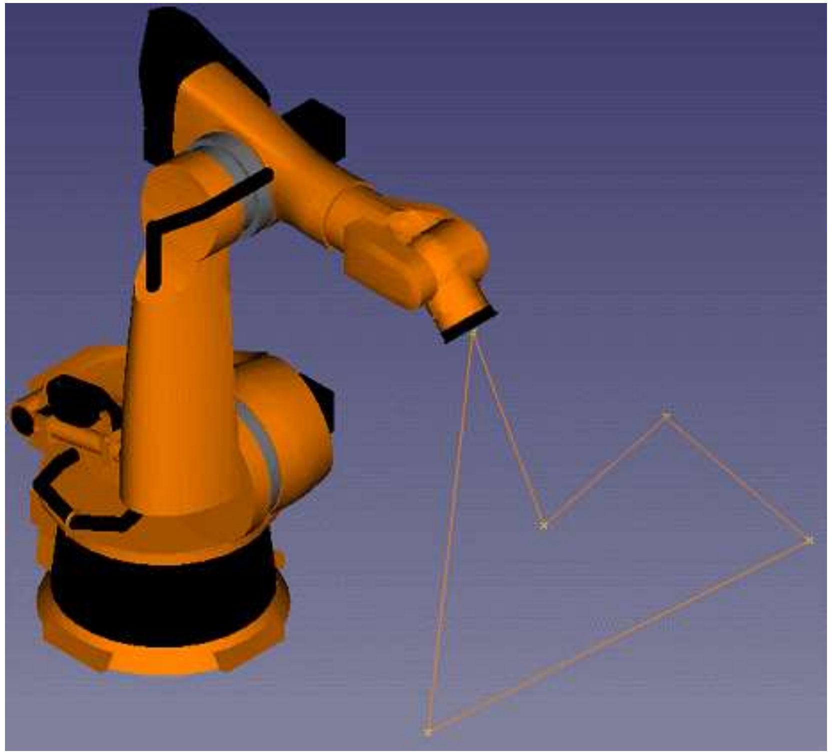
- In this way the **nominal motion** can be devised **without** rigorously **taking** the **robots limits** into **accounts** as the online algorithm will account for them at runtime.

- Trajectory scaling algorithms exploit the path-velocity decomposition of the task and use the **parametrization variable** of the **path curve** as an **additional degree** of **freedom** to meet the robot constraints.

- The **input** of the **algorithm**: **desired motion** to be followed.

- The **output**: the **scaled trajectory** (typically joint positions and velocities given to the robot low-level controller).

- **Novelty** in the **online trajectory scaling method**: until now all trajectory scaling methods slow down the nominal trajectory when it is too demanding: this result in a delay with respect the nominal task. Most of the algorithms in the literature do not show any **delay-recovering** feature or if they do this lead to jearky motion or overshoots.

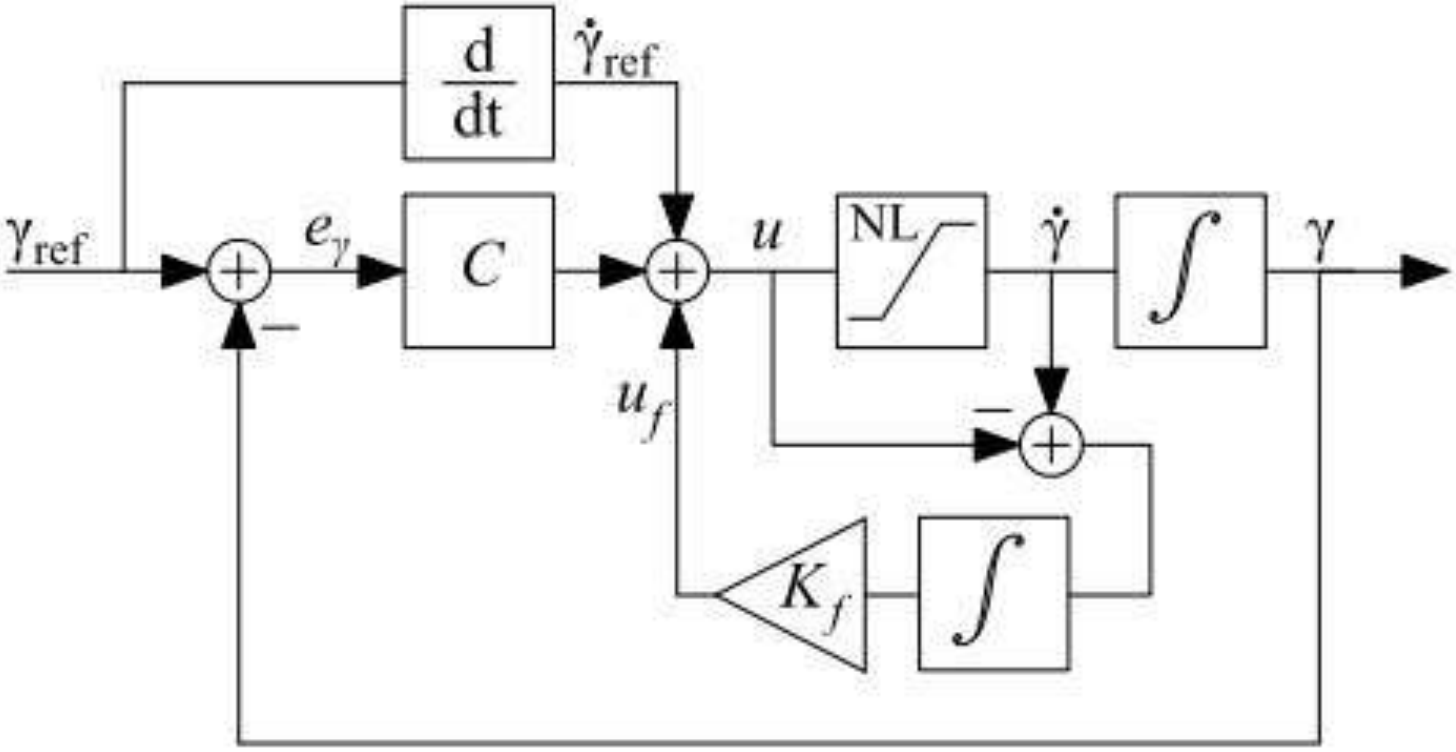


A NEW CONTROL SCHEME

- The new proposed control scheme is composed by 2 fundamental elements:

  1) *External control loop*

  2) *Saturation fly wheel block*



$$u = \dot{\gamma}_{ref} + C(e_\gamma) + K_f \int_0^{t_0} (\dot{\gamma} - u)dt$$

The control law

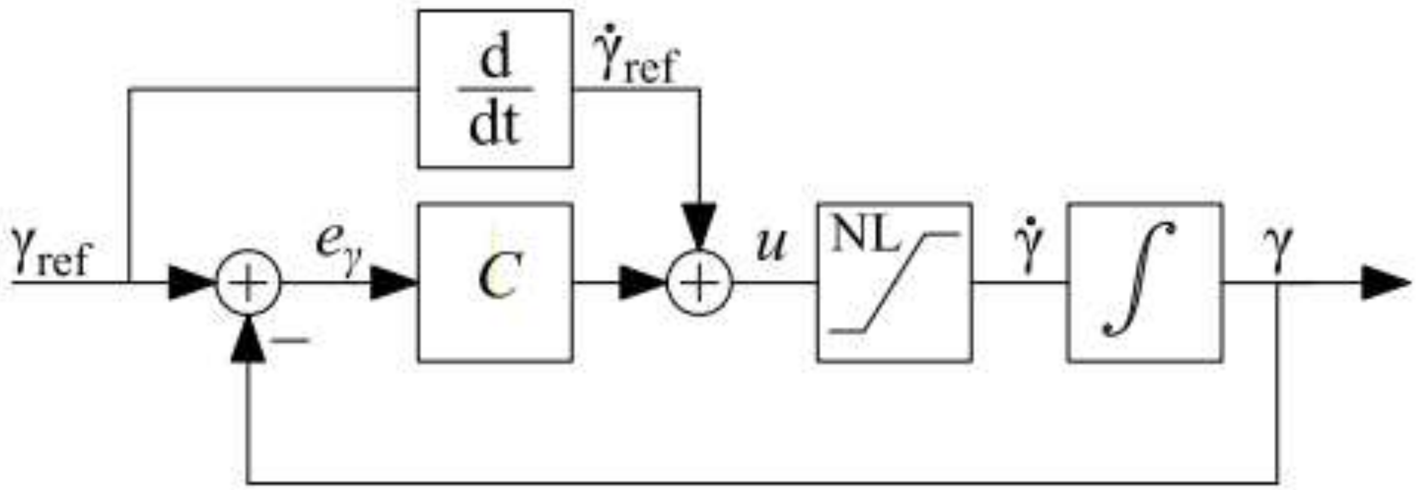
2 Fundamental elements:

- **External control loop:** control variable γ and reference signal γ_{ref} and the error is $e_\gamma = \gamma_{ref} - \gamma$.

  The feedfoward action is $\dot{\gamma}_{ref}$, when no slow down occurs it is the only input.

  When the scaling is activated, the external loop increases the control action to recover the delay.

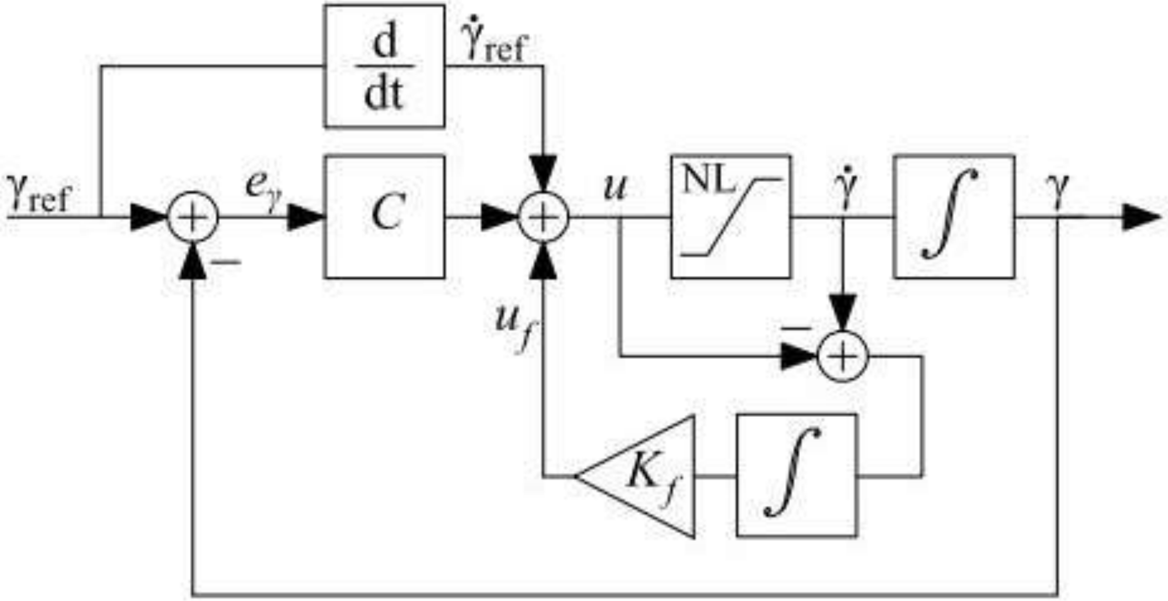
  Online control problem : **saturation block** between u and $\dot{\gamma}$.



The control law

2 Fundamental elements:

- **Saturation fly wheel block:** past history of the desired trajectory + behaviour algorithm = good results in preserving path feasibility. When slow-down phase ends large control error drives the system to saturation, to prevent this the control error should be bounded to a value that drives $\dot{\gamma}_{ref}$ to a saturated value of $\dot{\gamma}$. The inner control loop $u_f = K_f \int_0^{t_0} (\dot{\gamma} - u)\, dt$ will solve this problem.

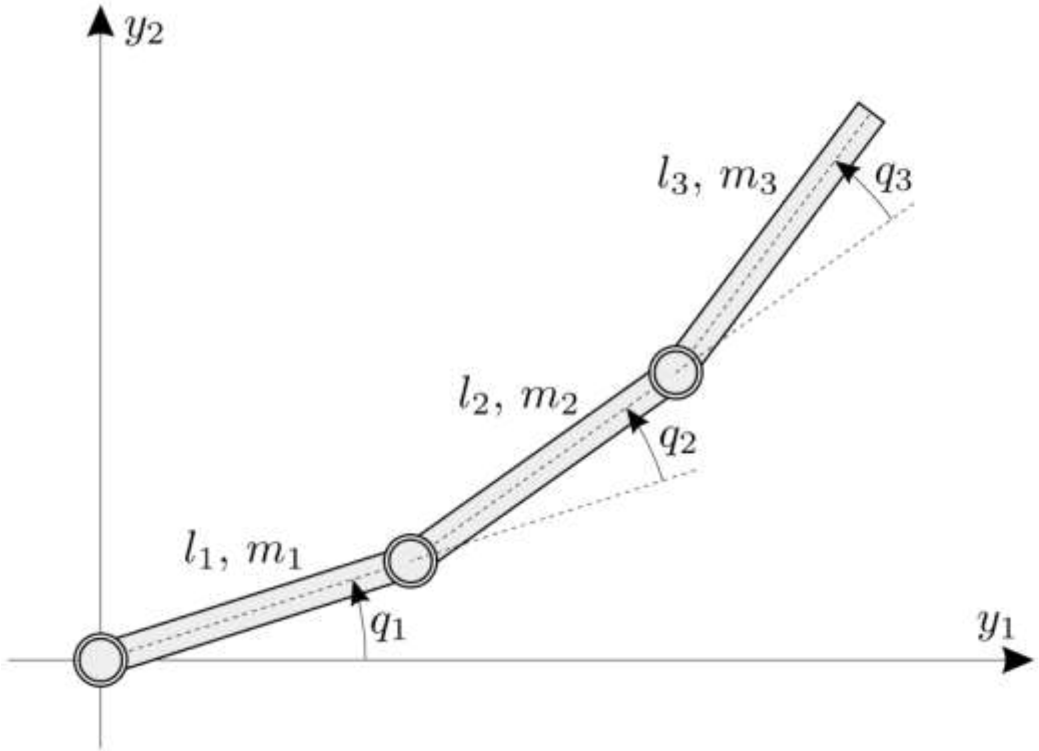


- The effect is to keep $\dot{\gamma}_{ref}$ close to $\dot{\gamma}$ during saturation and to accmumulate '**saturation intertia**' reduces the growth of $\dot{\gamma}$ that occurs when saturation ends.
- **Control input**:

$$u = \dot{\gamma}_{ref} + C(e_\gamma) + K_f \int_0^{t_0} (\dot{\gamma} - u)dt$$

Notice: the robot is commanded by velocity!

3R Robot



- In order to test the effectiveness of the control scheme just introduced we considered a 3R robot for our simulations.

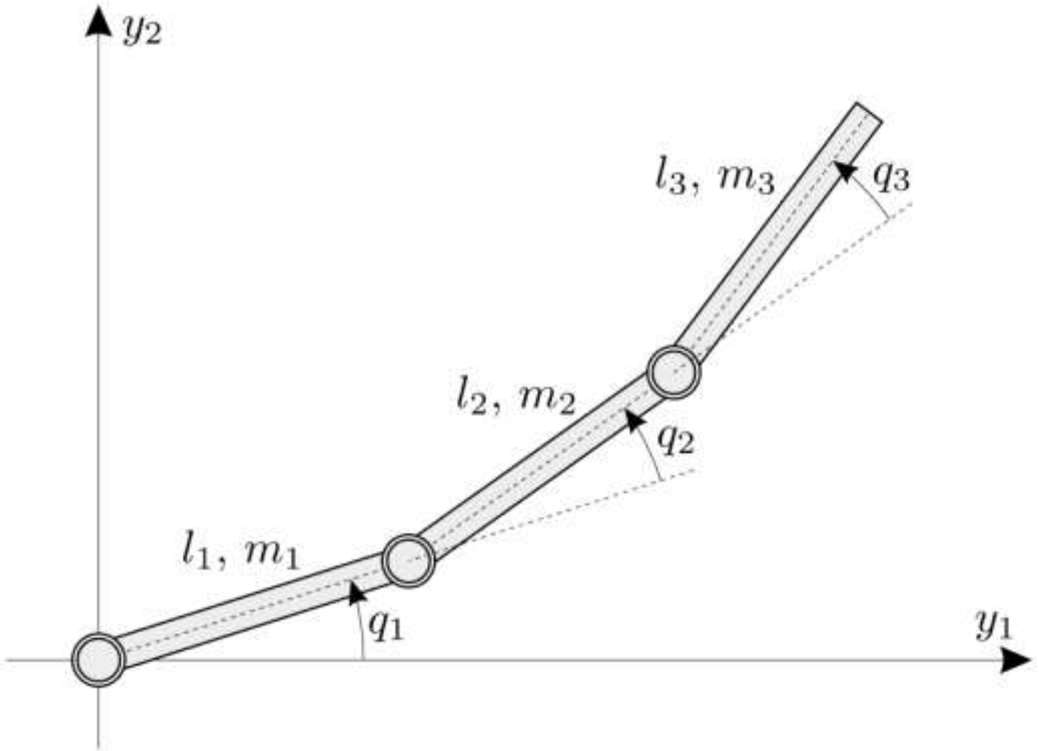
DH TABLE of the Robot

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

Unitary link length robot

$$L_1 = L_2 = L_3 = 1m.$$

3R Robot



To introduce at least one non ideality in our simulations we considered the robot joint limits as follow:

Velocities:

$-4rad/s <= \dot{q}_1 <= 4rad/s,$

$-4rad/s <= \dot{q}_2 <= 4rad/s,$

$-4.3rad/s <= \dot{q}_3 <= 4.3rad/s.$

Accelerations:

$-25rad/s^2 <= \ddot{q}_1 <= 25rad/s^2,$

$-15rad/s^2 <= \ddot{q}_2 <= 15rad/s^2,$

$-20rad/s^2 <= \ddot{q}_3 <= 20rad/s^2.$

Timing-law

- In this control law the **timing law** plays the role of the **control variable.**
- In our simulations we parameterized the **Cartesian path** with a **quintic polynomial** timing law to apply boundaries condition also for the acceleration.

$$\gamma(\tau) = a\tau^5 + b\tau^4 + c\tau^3 + d\tau^2 + e\tau + f \qquad \tau = t \,/\, T$$

- By applying a **rest to rest motion**, the following **formula** are simply obtained.

$$\gamma(\tau) = \gamma_{in} + \Delta\gamma(6\tau^5 - 15\tau^4 + 10\tau^3) \text{ with } \Delta\gamma = \gamma_{fin} - \gamma_{in}.$$

$$\dot{\gamma}(t) = (30t^2)/T^3 - (60t^3)/T^4 + (30t^4)/T^5.$$

$$\ddot{\gamma}(t) = (60t)/T^3 - (180t^2)/T^4 + (120t^3)/T^5.$$

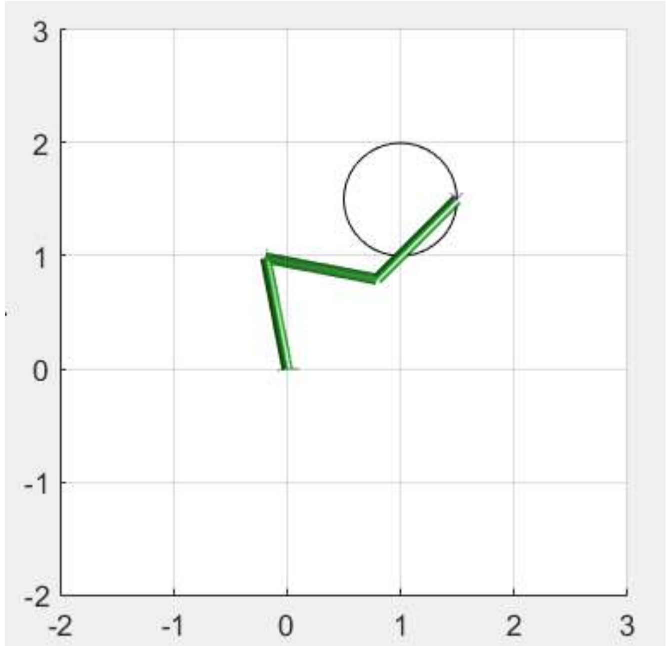
Circular trajectory

- The first trajectory we are going to consider is a circular trajectory where:

  - $R=0.5m$ and $X_0=1m$ $Y_0=1.5m$

  - Starting point (1.5, 1.5), endeffector's orientation=45°.

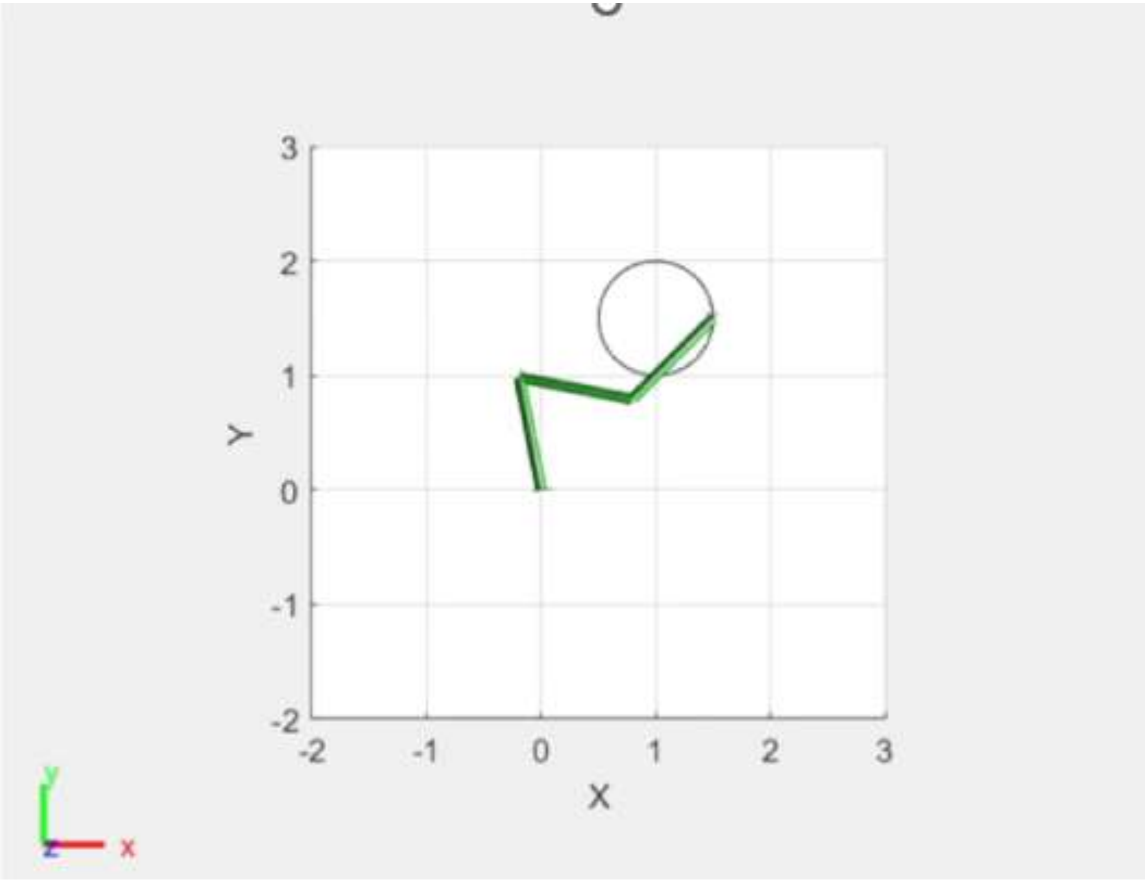
$$p(\gamma) = \begin{pmatrix} X_0 + Rcos(2\pi\gamma) \\ Y_0 + Rsin(2\pi\gamma) \\ \pi/4 \end{pmatrix} \quad \gamma(\tau) \in [0,1]$$



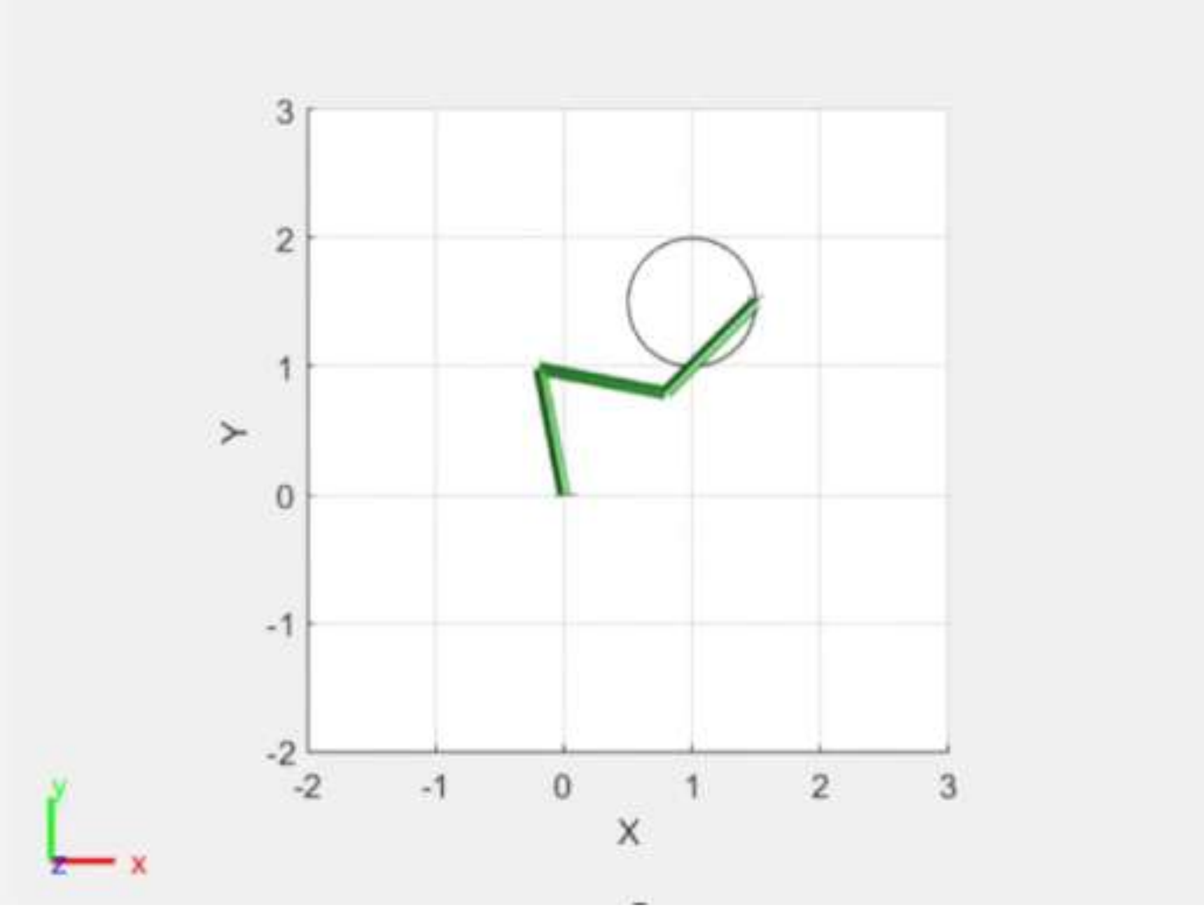
- We will carry out 3 different experiments:

1) With **execution time T=3.9** (the last time the joint limits are respected in the nominal condition).
2) With **execution time T=3.6** (here for the first time the scaling algorithm intervenes).
3) With **execution time T=3.4** (a more demanding case to appreciate the real effectiveness of the algorithm).

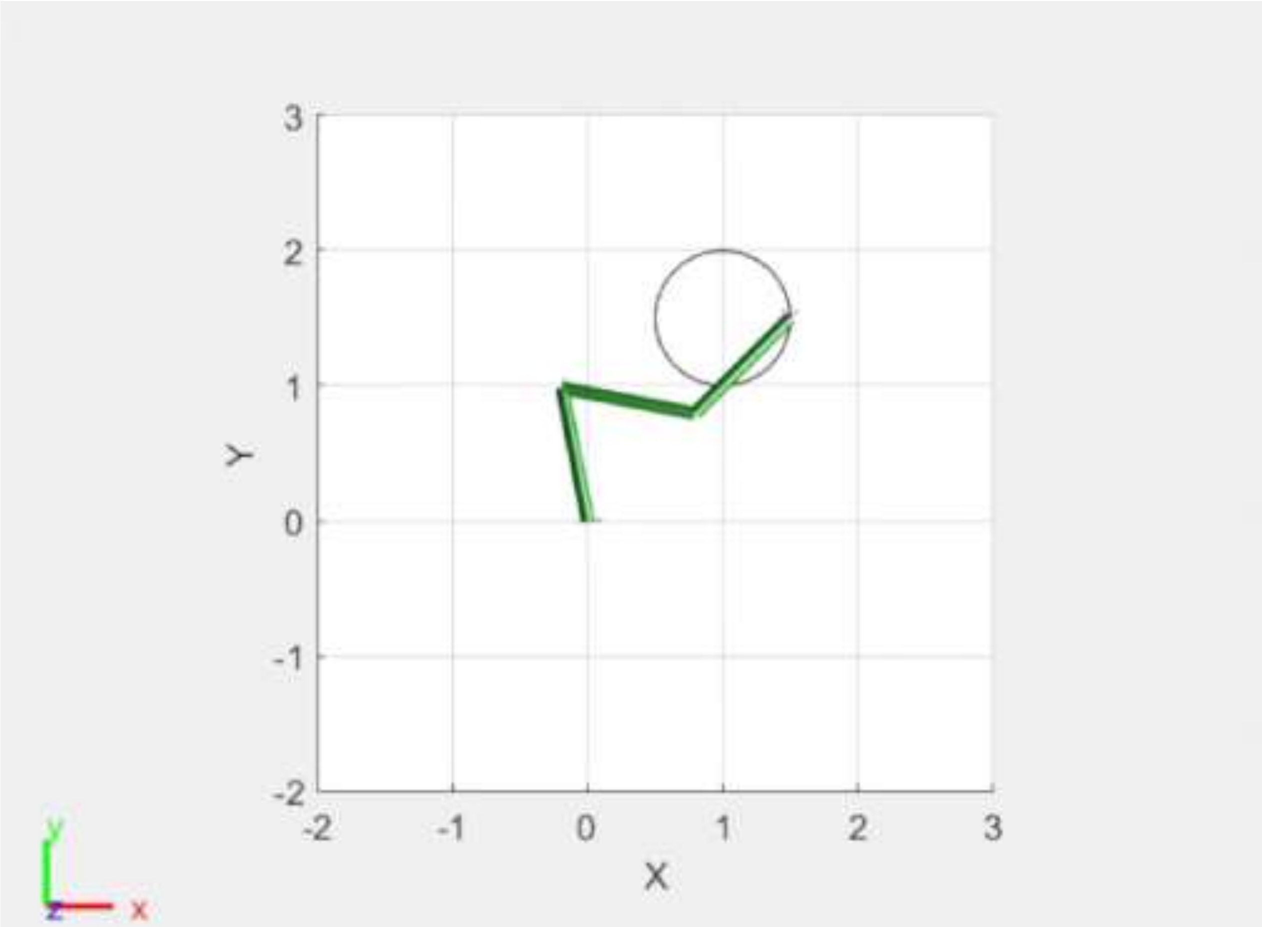
Circular trajectory - T=3.9s



Circular trajectory - <u>T=3.6s</u>

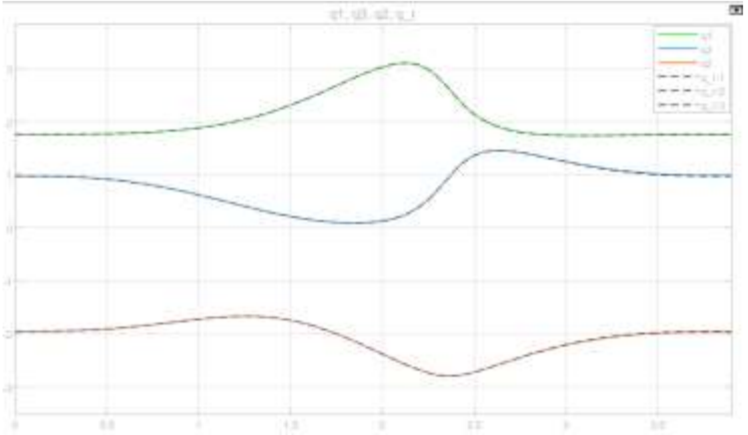


Circular trajectory - <u>T=3.4s</u>

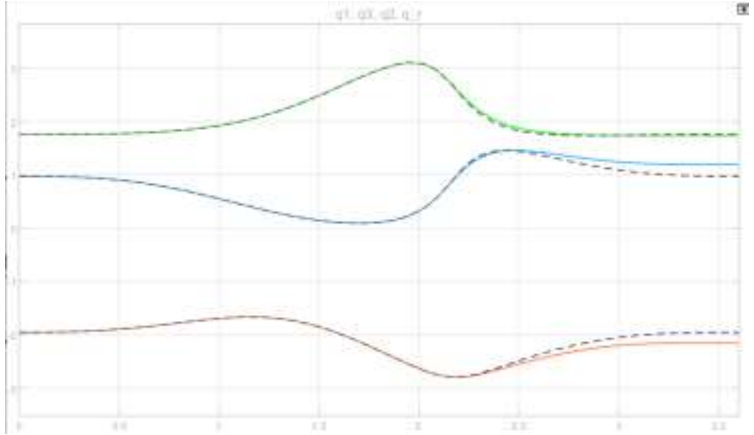


Position profiles

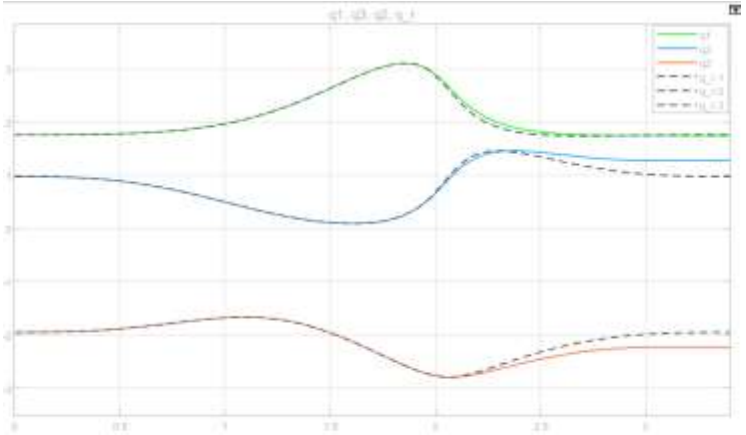
T = 3.9s



T = 3.6s

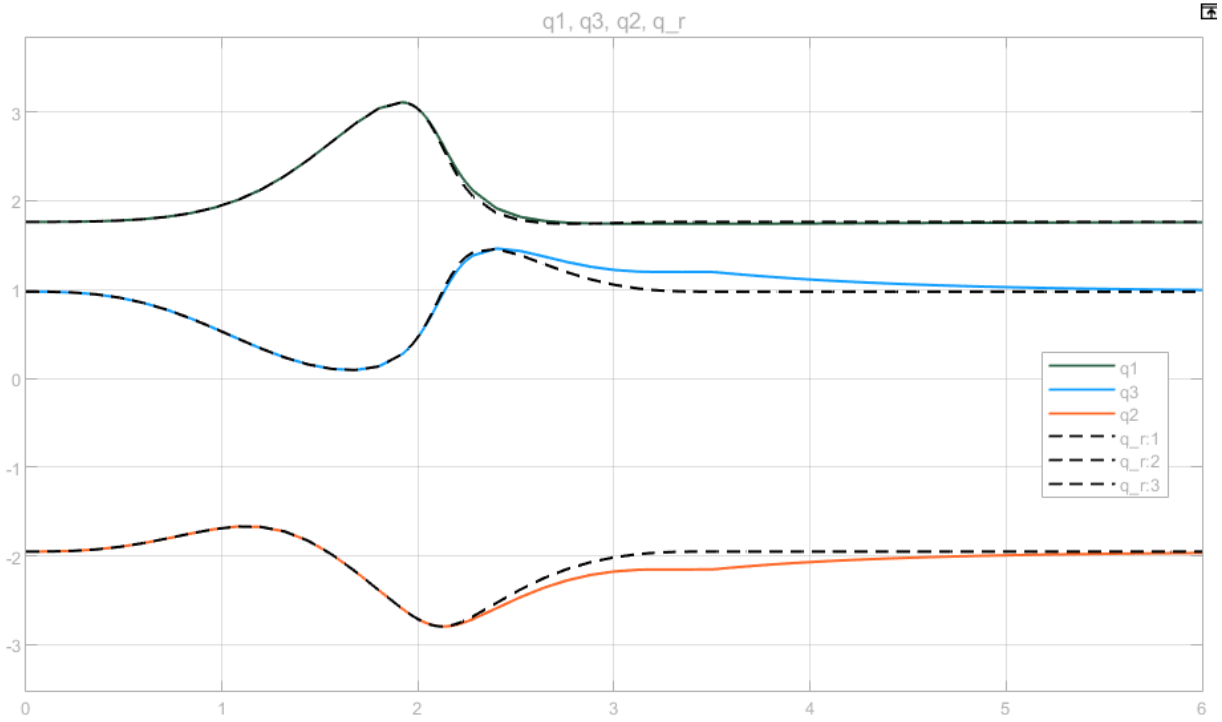


T = 3.4s



- By progressively reducing the time we can notice an **error** in **position** at the **end** of the **simulation**.
- This is not a real problem because the **error** is **not** in **space** but in **time**: this means that if we increase the time of the experiments the robot will close the path.

Solution for the error in time



- **Solution** to reduce the joint's error position at the end of the experiments:

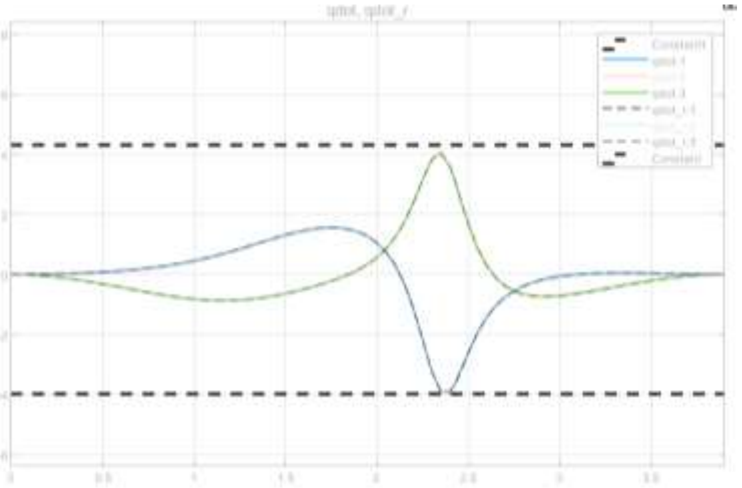
  1) For t>T we set as reference $\dot{\gamma}_{ref} = 0$ and $\gamma_{ref} = 1$

  2) We extend the execution time T for another interval of time Δt that allows the robot to close the path

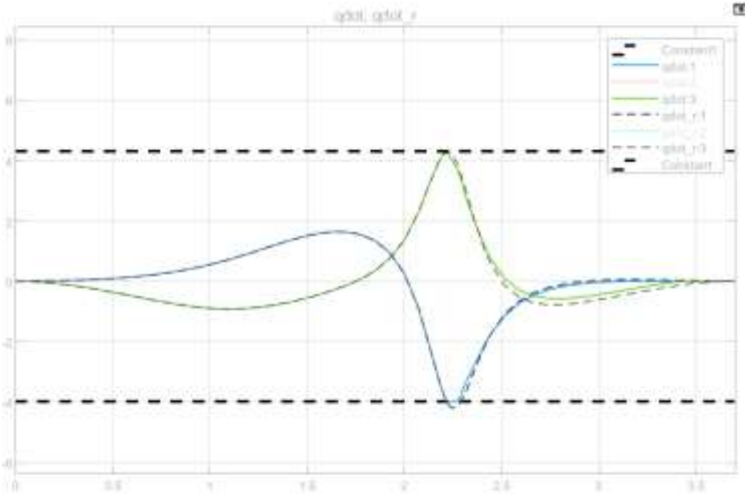
- We decided to show only the cases without this addition to clearly show the final error.

Velocity profiles

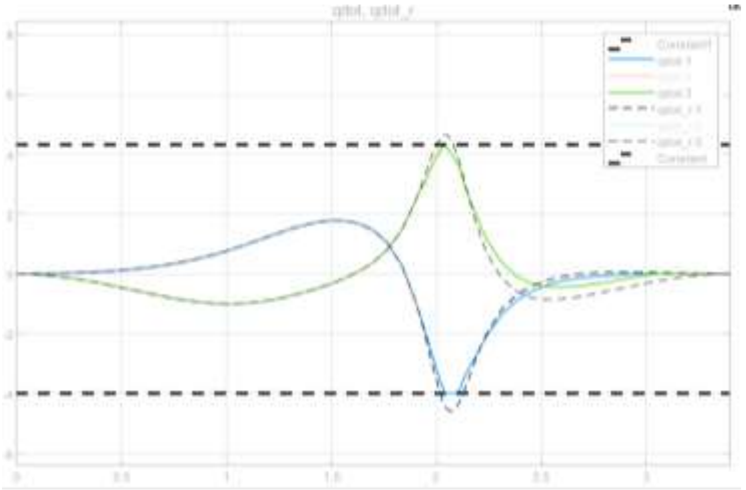
T = 3.9s



T = 3.6s



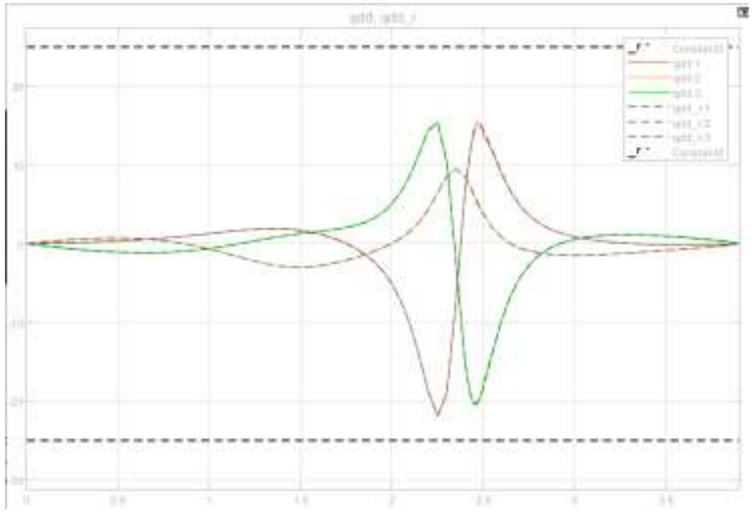
T = 3.4s



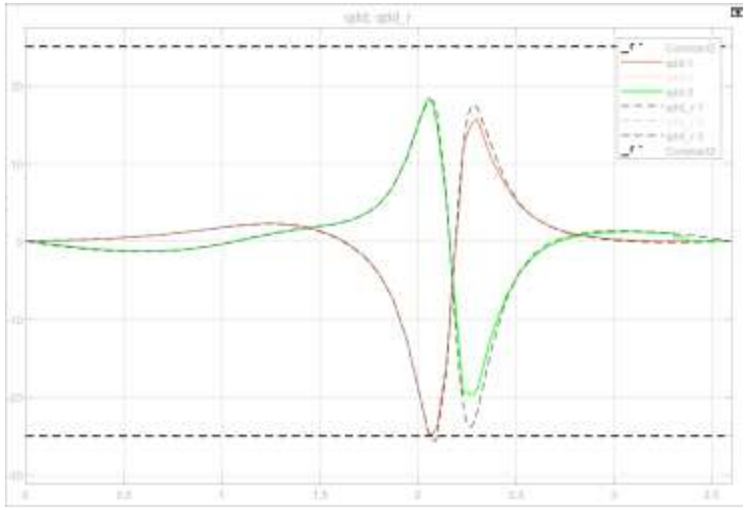
- Here it is clear how the algorithm works: when the **reference velocity** is **within** the **limits** it **doesn’t activate**.
- Instead, when the **boundaries** are **not respected (nominal condition too demanding)** it **modulates** the **velocity** in order to saturate it. This leads to a slow down but then when this phase ends the algorithm is able to recover the delay by speeding up the velocity.

Acceleration profiles

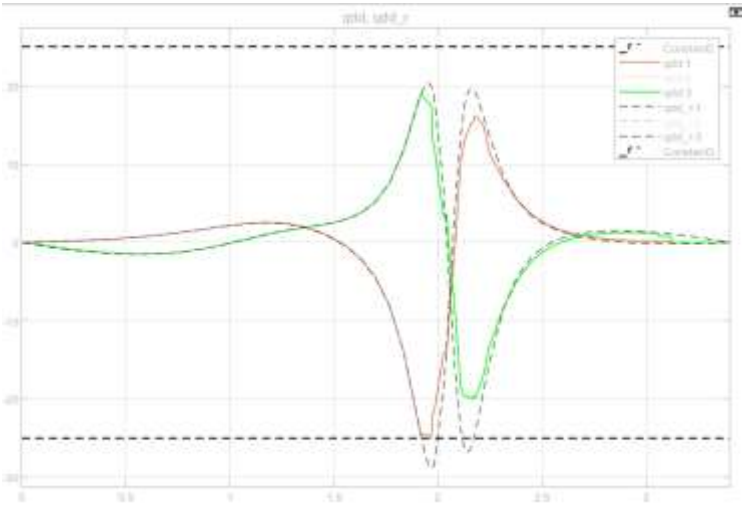
T = 3.9s



T = 3.6s



T = 3.4s

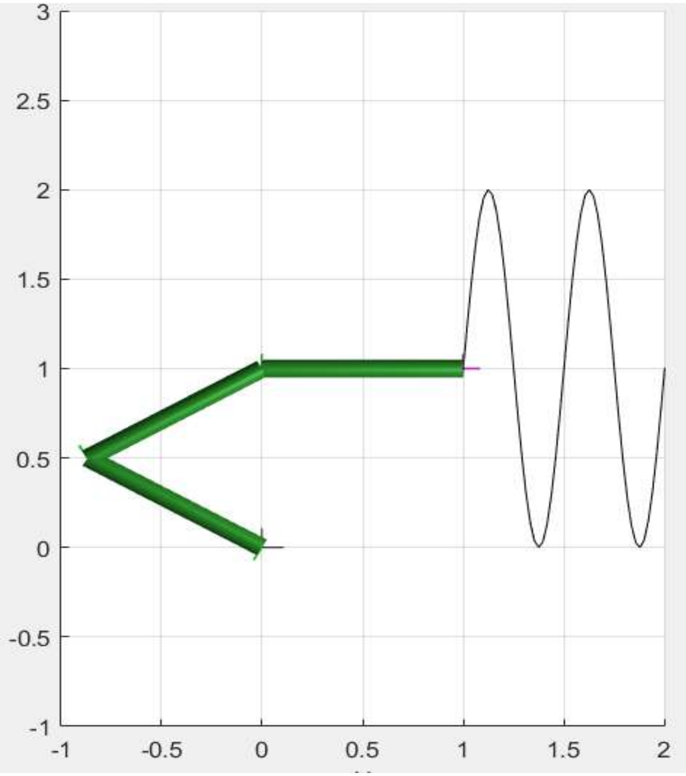


- Even in the acceleration profiles can be noted the intervention of the scaling algorithm.

Sinusoidal trajectory

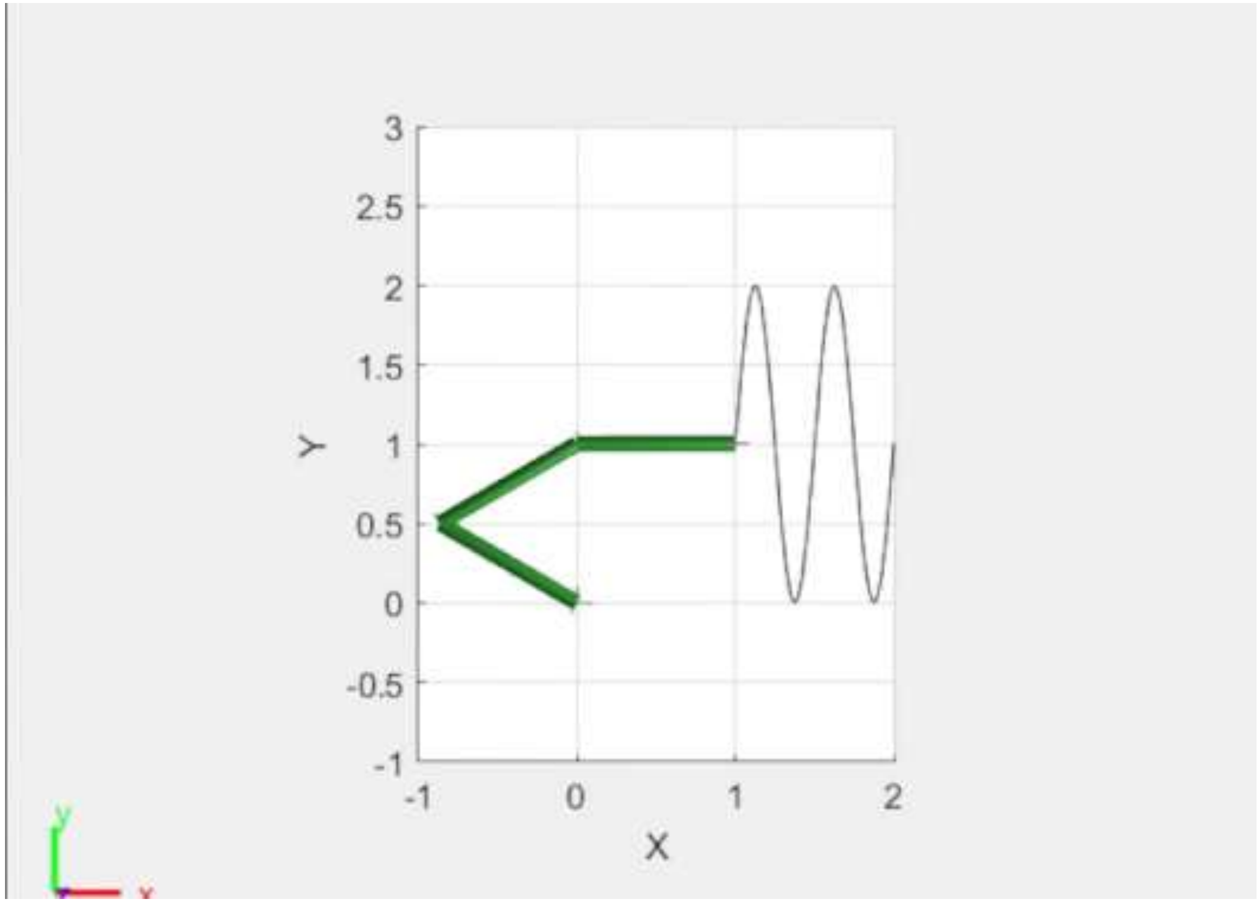
- The second trajectory we are going to consider is a **sinusoidal trajectory**.
- Also in this case the timing law is a **quintic polynomial** but the endeffector follows the timing law trend.

- We will carry out 3 different experiments by progressively shrinikg the time, the first one **T=12.7s** then **T=12s** and in the end **T=11.5s.**

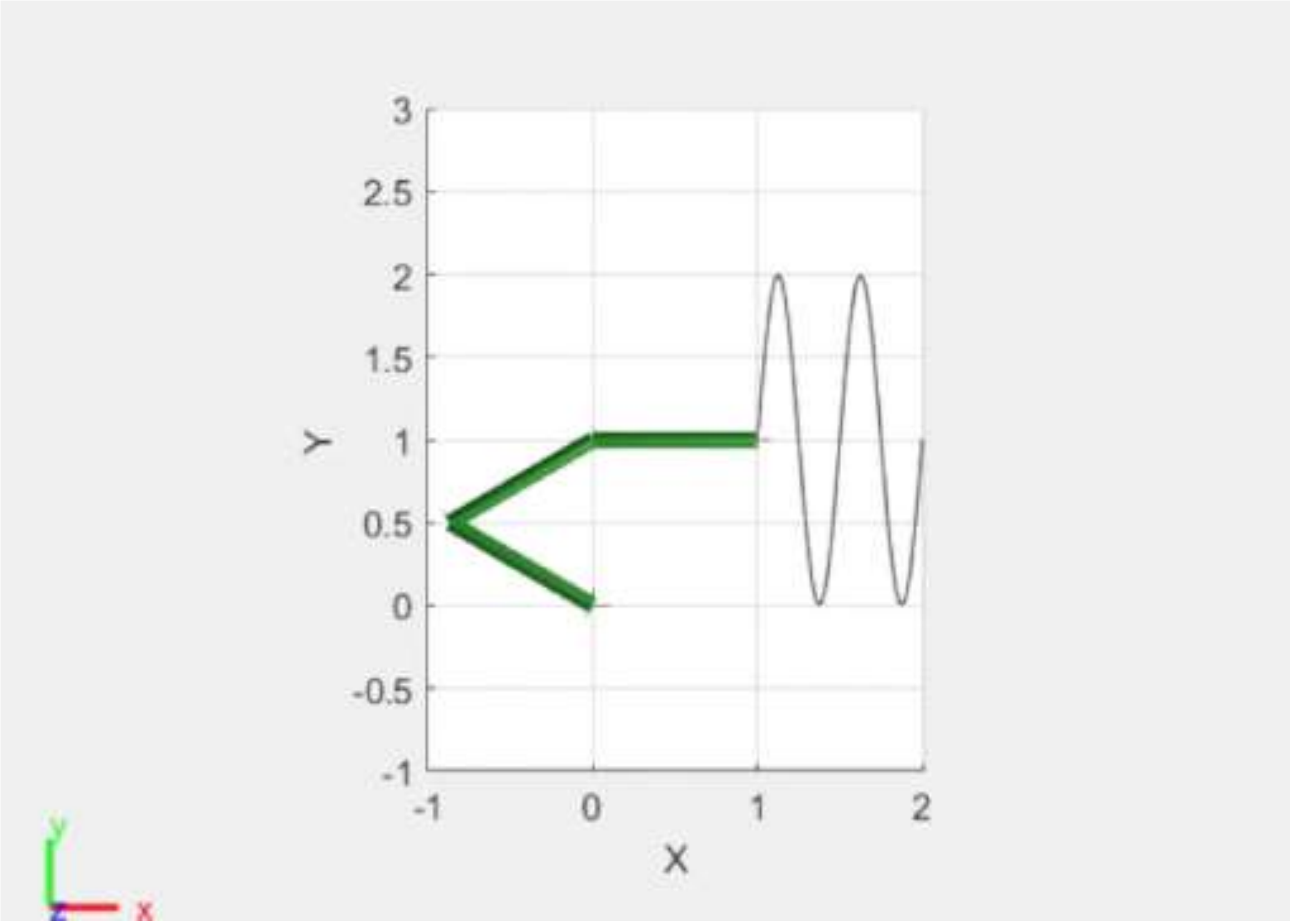


Path parameterization: $$p(\gamma) = \begin{pmatrix} \gamma + 1 \\ 1 + sin(4\pi\gamma) \\ \gamma \end{pmatrix} \quad \gamma(\tau) \in [0,1].$$

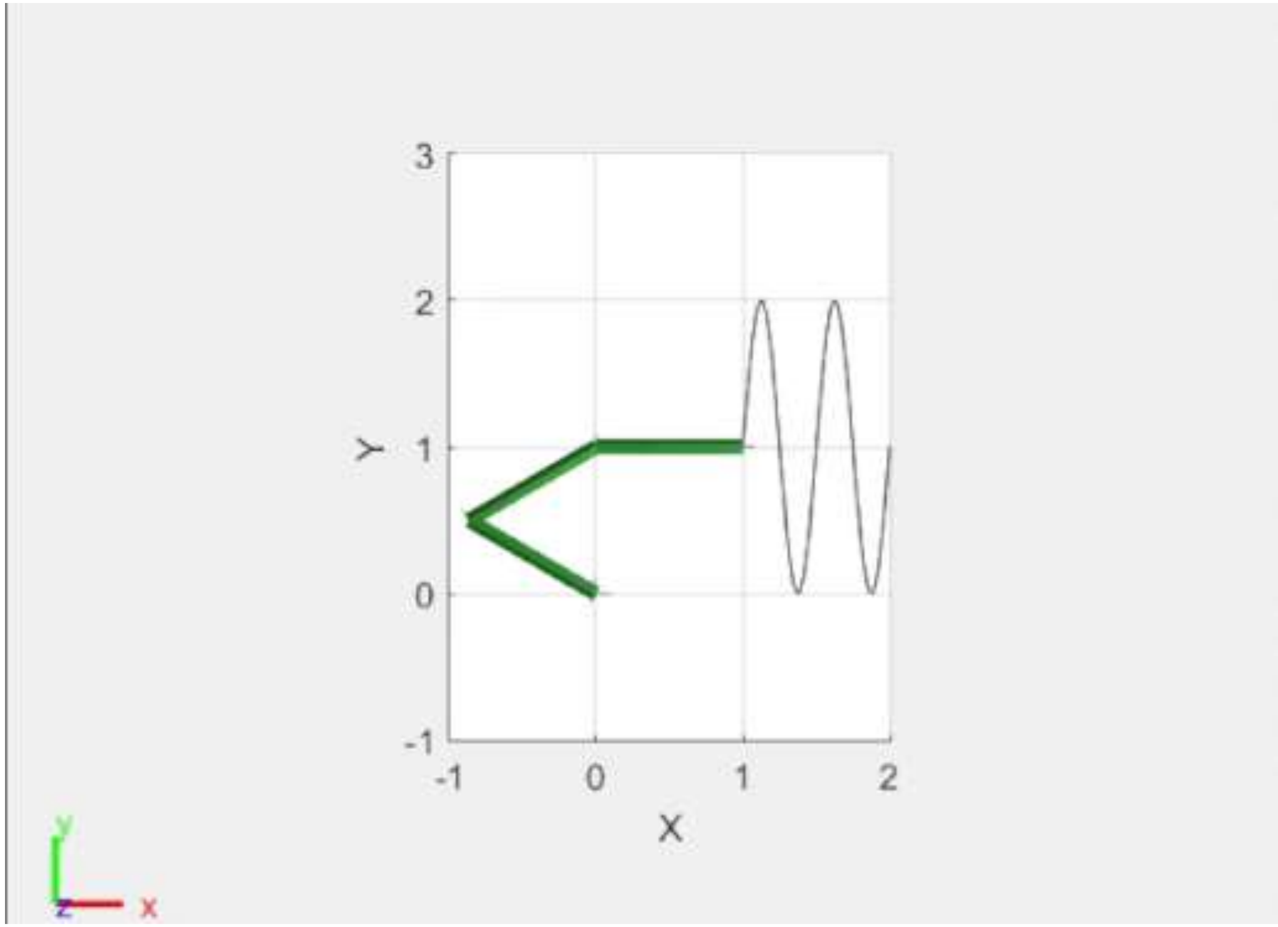
Sinusoidal trajectory - <u>T=12.7s</u>



Sinusoidal trajectory - T=12s

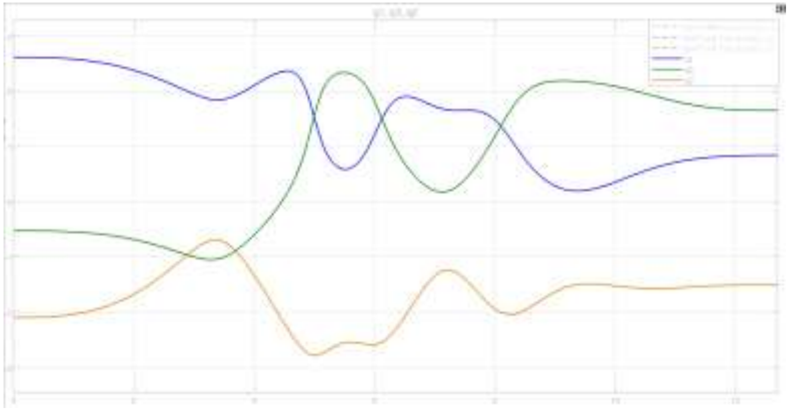


Sinusoidal trajectory - <u>T=11.5s</u>

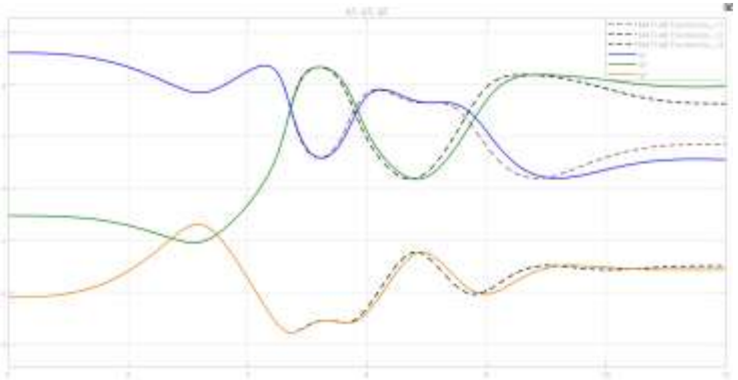


Position profiles

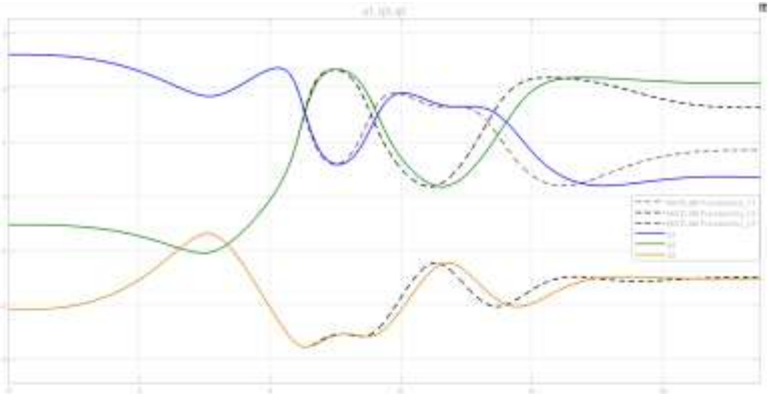
T = 12.7s



T = 12s



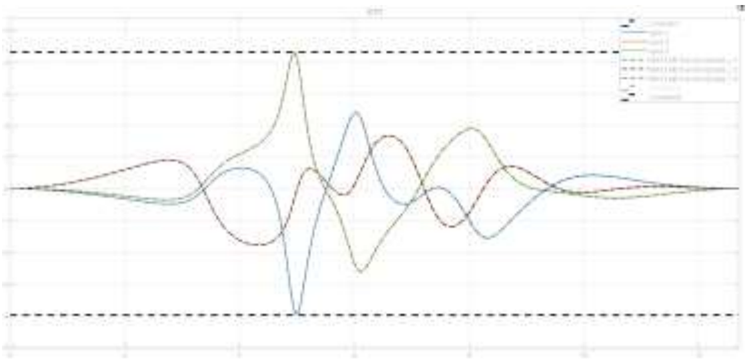
T = 11.5s



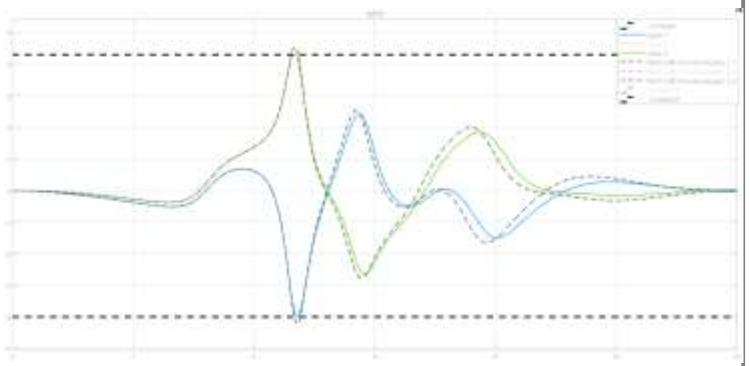
- Also, in this case we can notice that by shrinking the time we obtain an error in time but not in space.

Velocity profiles

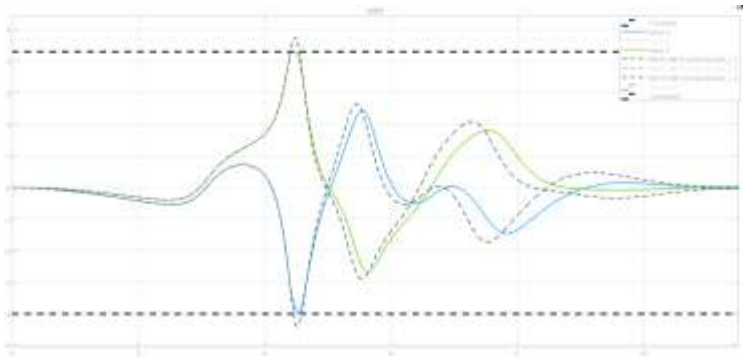
T = 12.7s



T = 12s



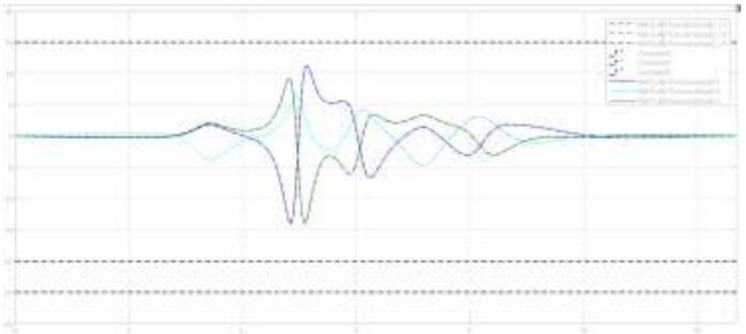
T = 11.5s



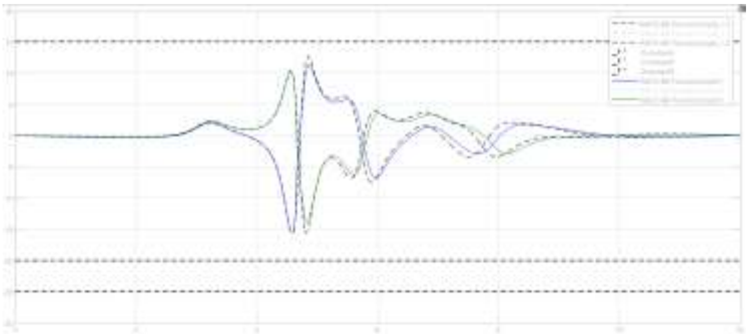
- In these velocity profiles it is really clear how the algorithm works: it modulates the velocity when the nominal one is too demanding and then the slowdown-phase ends.

Acceleration profiles

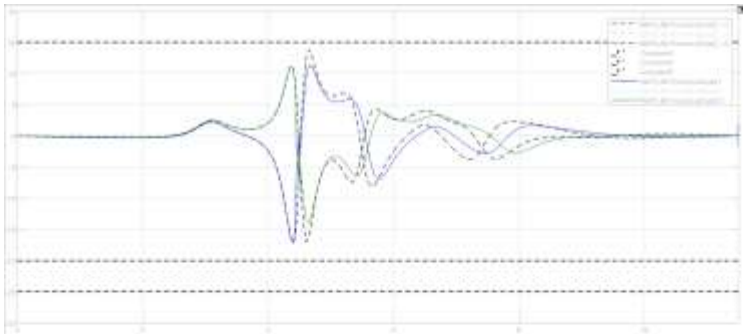
T = 12.7s



T = 12s



T = 11.5s



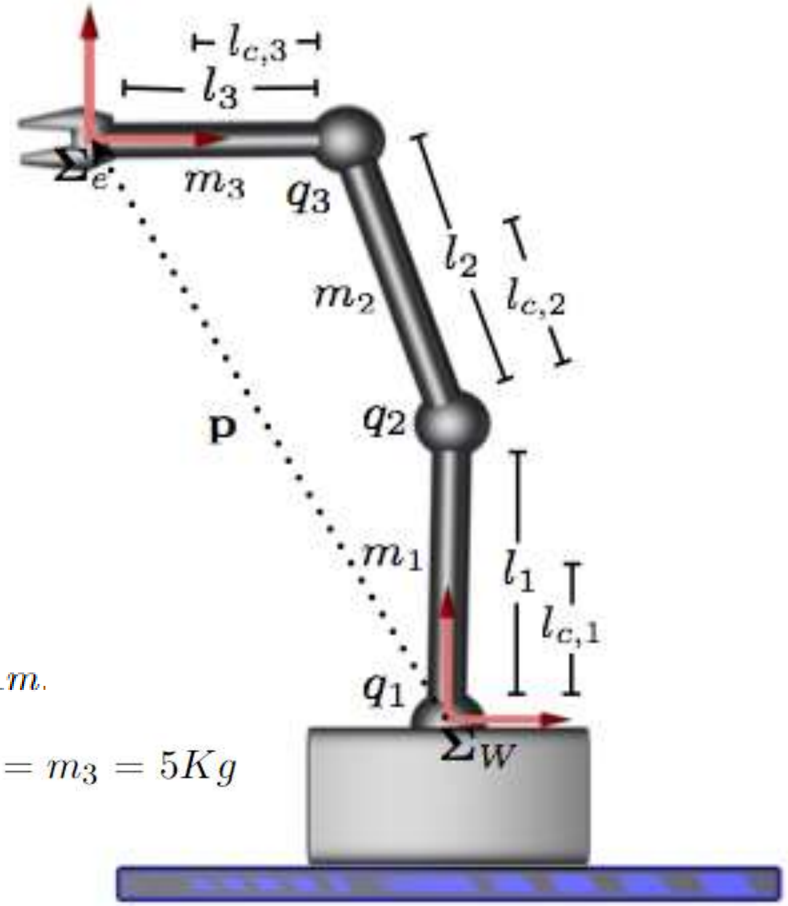
DYNAMICS

- Until now we haven't considered the **dynamic model** of the robot because we commanded it by velocity.
- The dynamic model is described in the below table.

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	$L_3/2$	$(m_3 * L_3^2)/12$

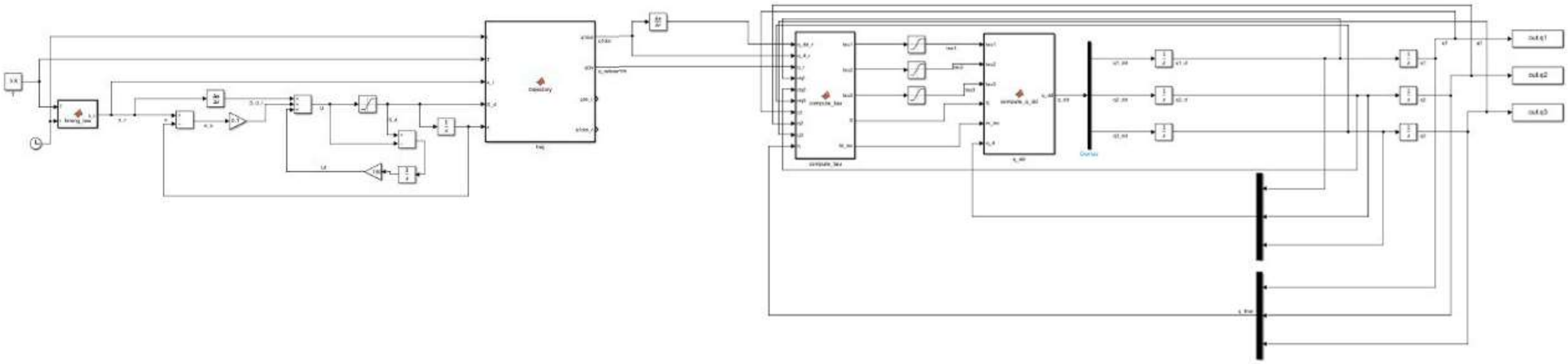
Unitary link length robot: $L_1 = L_2 = L_3 = 1m.$

Uniformly distribuited masses: $m_1 = m_2 = m_3 = 5Kg$



New control scheme

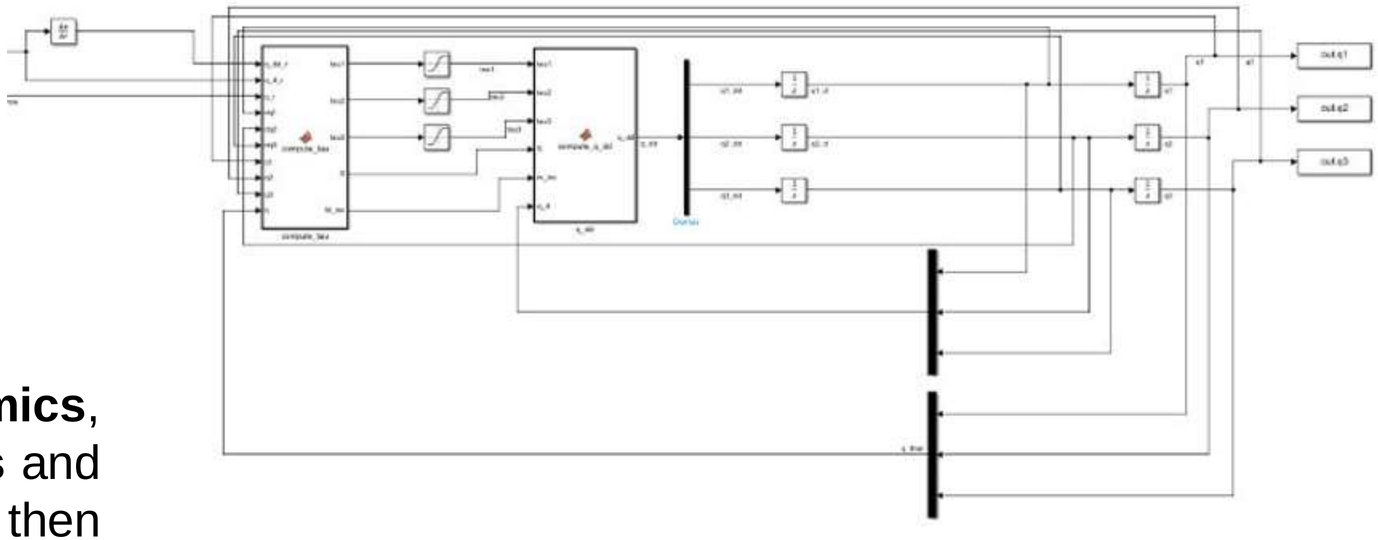
Control scheme **Robot dynamics part**



New control scheme

- We linked the first part with the robot dynamics in the following way:

Robot dynamics part



- The **output scaled** joint's **positions**, **velocities** and **accelerations** $q, \dot{q}$ and $\ddot{q}$ are now the **input** of the second controller as **nominal condition**

- The idea is to consider the **robot dynamics**, to introduce **boundaries** on the **torques** and to choose a **suitable control-law** and then to see if we are still able to perform the scaled profiles.

New control scheme

- **Control law:** feedback linearization (FBL) + PD + FFW that is a **trajectory controller.**

$$\tau = M(q)(\ddot{q}_r + K_p e + K_d \dot{e}) + S(q,\dot{q})\dot{q}_r, \text{ with } e = q_r - q \text{ and } \dot{e} = \dot{q}_r - \dot{q}.$$

$$K_p = \begin{pmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.1 \end{pmatrix} \; K_d = \begin{pmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 10 \end{pmatrix}.$$

- **Torque's boundaries**

$$-90Nm <= \tau_1 <= 90Nm$$

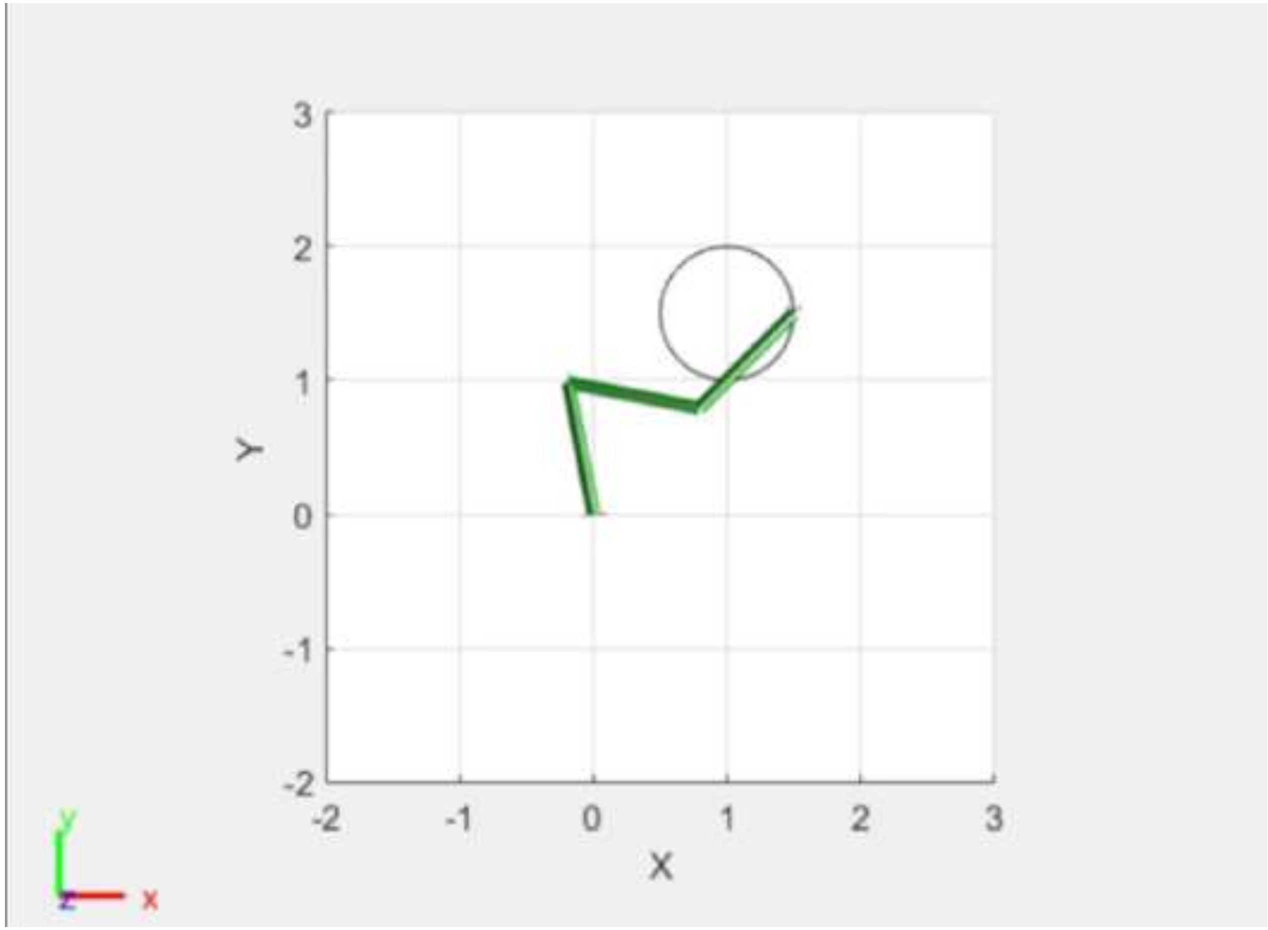
$$-60Nm <= \tau_2 <= 60Nm$$

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

- Now the **ideal case** is when the torques do **NOT saturate** and are able to execute the **nominal motion.**

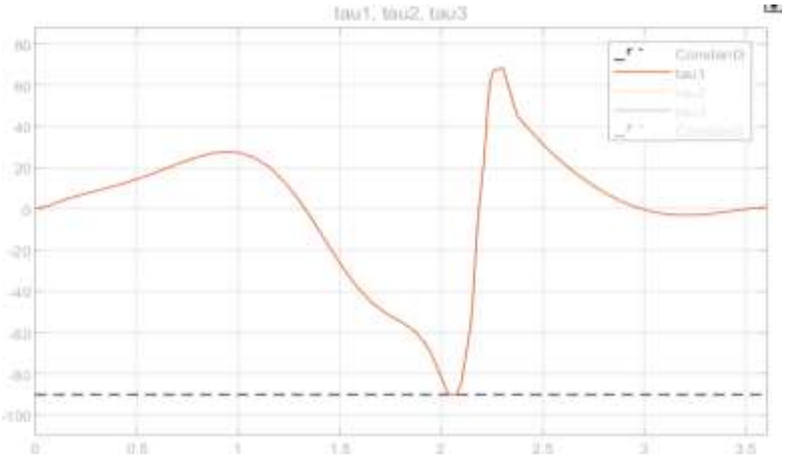
- The interesting case is performing the experiments with a **demanding time** in which the **torques saturate** and see the behavior of the new control law (i.e. if it is stil able to perform the scaled profiles or not).

T=3.6s – Dynamics circular case

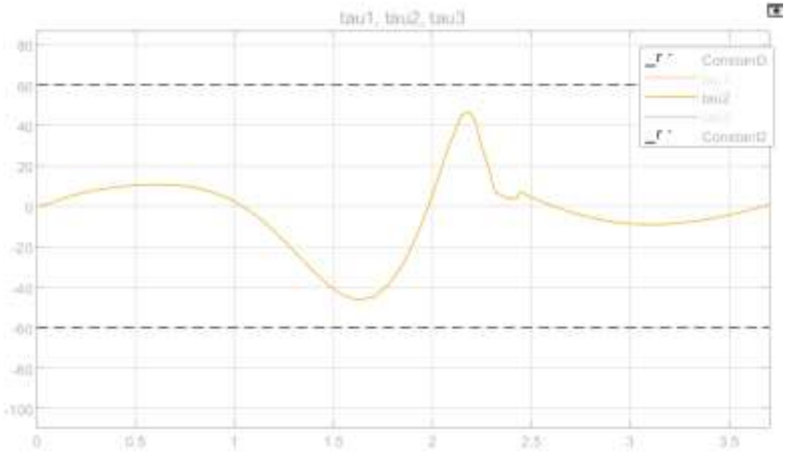


Torque profiles circular case - T=3.6s

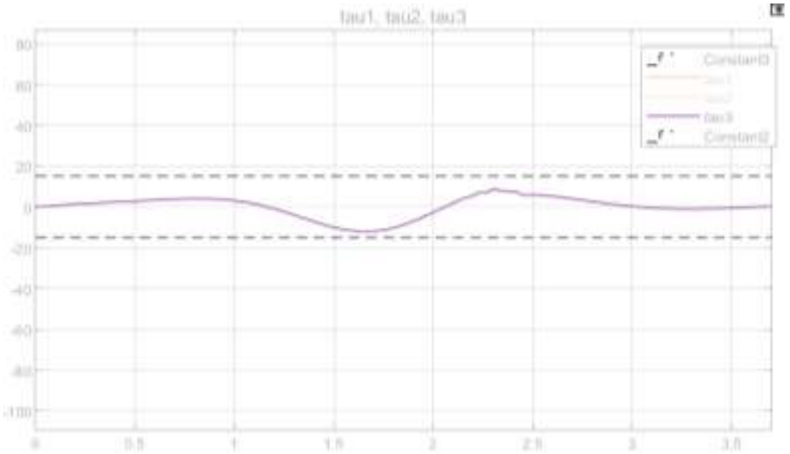
Torque Joint 1



Torque Joint 2



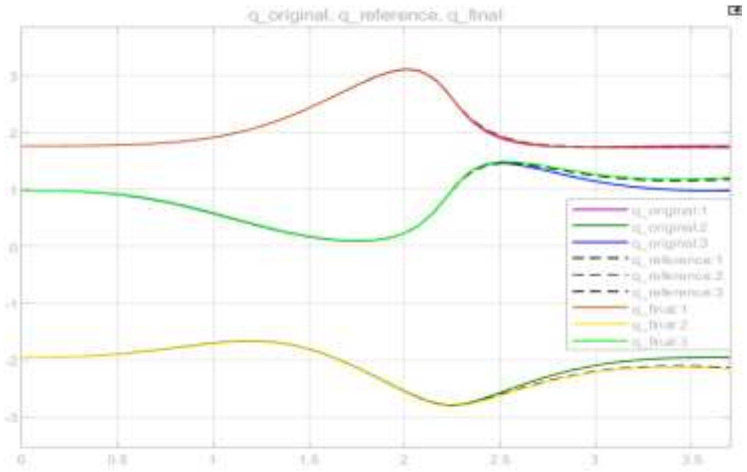
Torque Joint 3



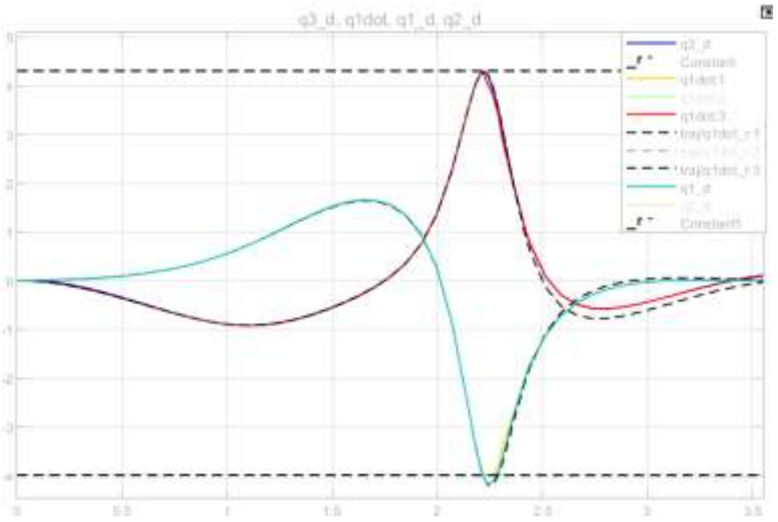
- Here we can see that the **torque** of the **Joint1 saturates** (because the nominal one goes beyond the limits), this leads to more non idealities.

Profiles dynamics circular case - T=3.6s

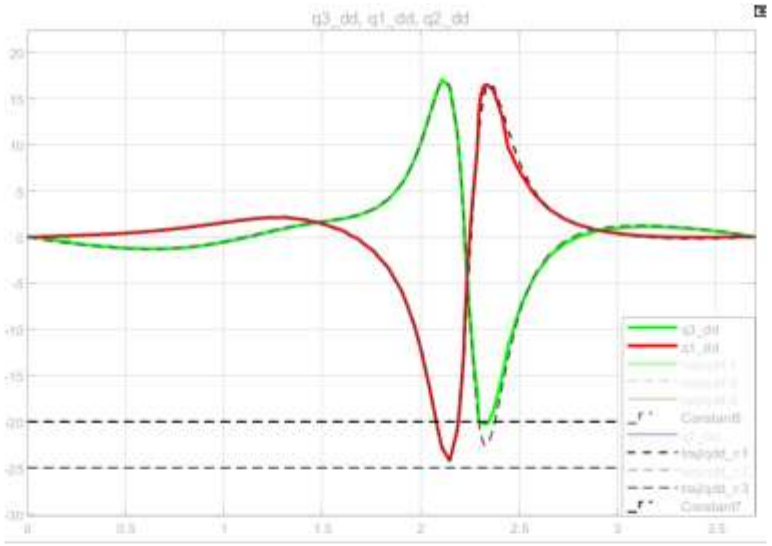
Position profiles



Velocity profiles



Acceleration profiles



- As we can see from the above graphs, despite the saturated torque of the first joint, the robot still manages to follow the scaled profiles, obviously with a larger, but derisory, error than in the previous case.

Sinuosidal case

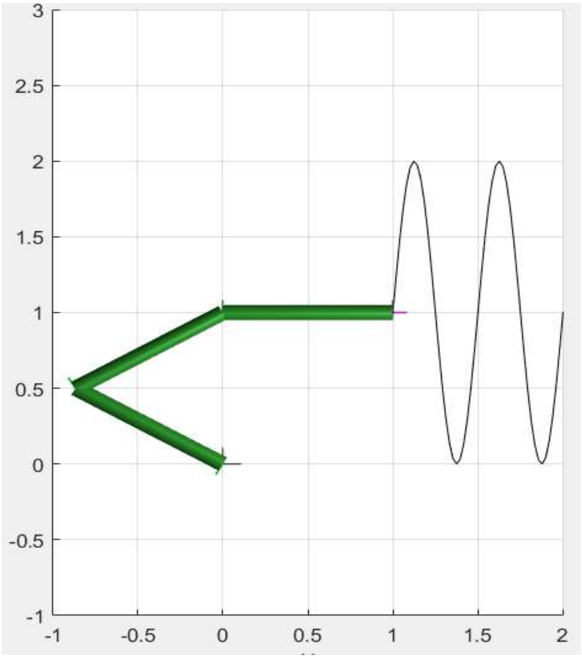
- For a didactic purpose for the experiments carried out with **sinusoidal trajectory** we imposed new torque boundaries because the original ones where widely respected even for very demanding times.

- **New torque boundaries**:

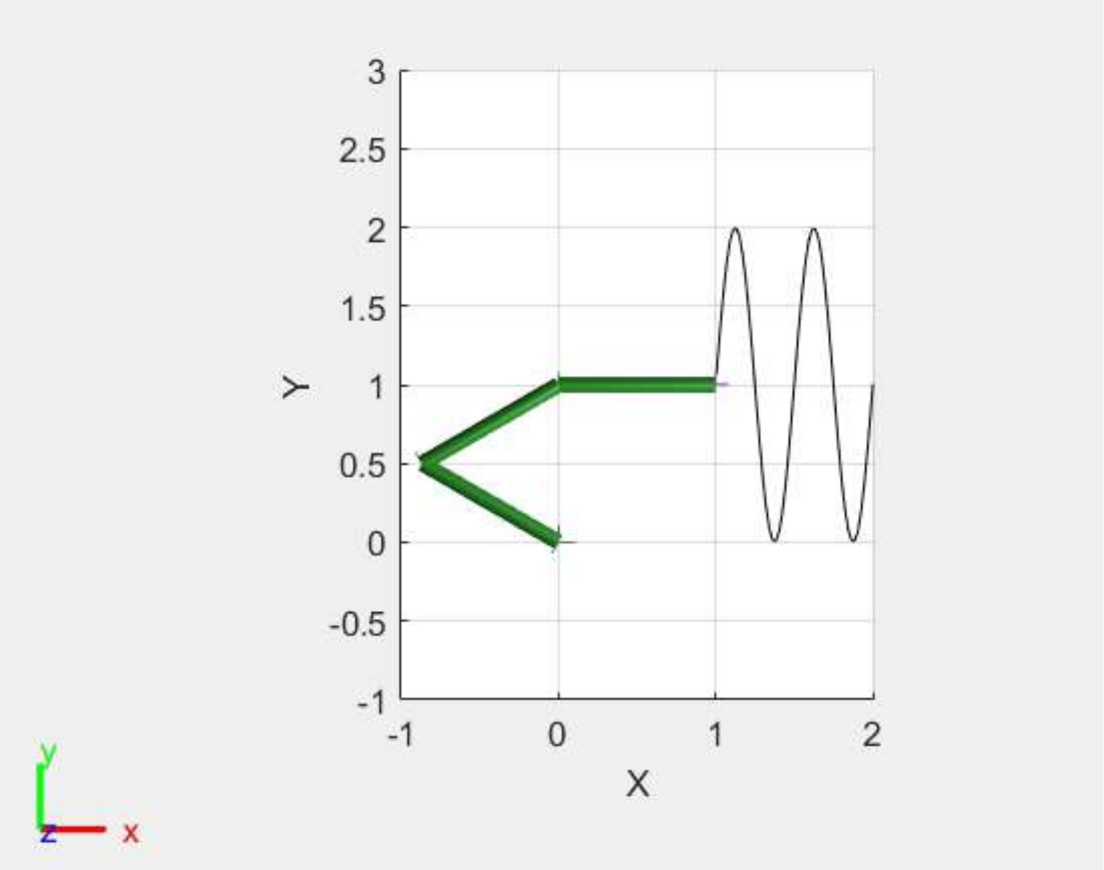
$$-45Nm <= \tau_1 <= 45Nm,$$

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

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

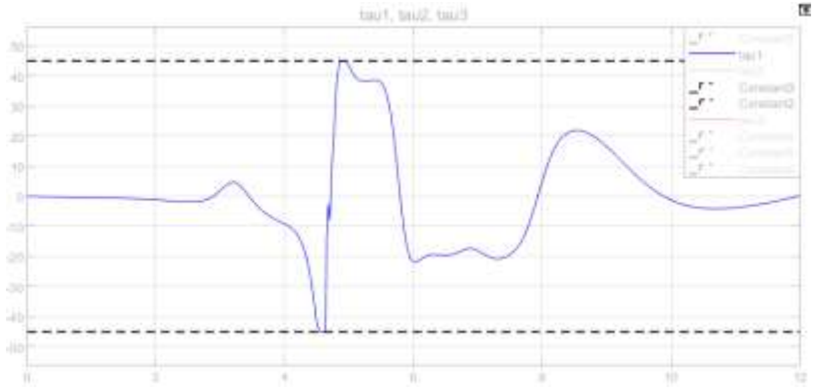


Dynamics sinusoidal case - T=12s

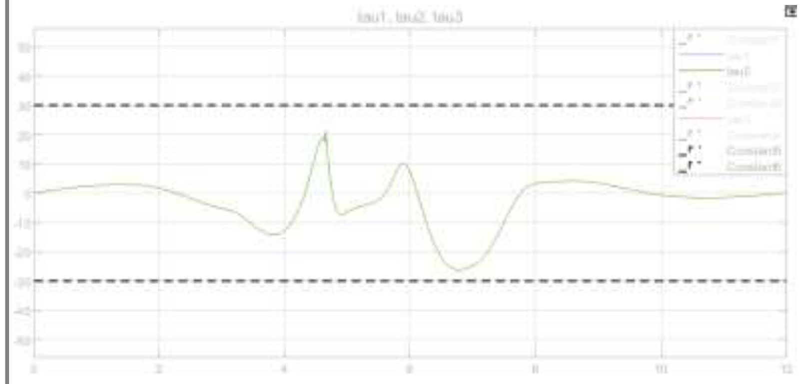


Torque profiles sinusoidal case - T=12s

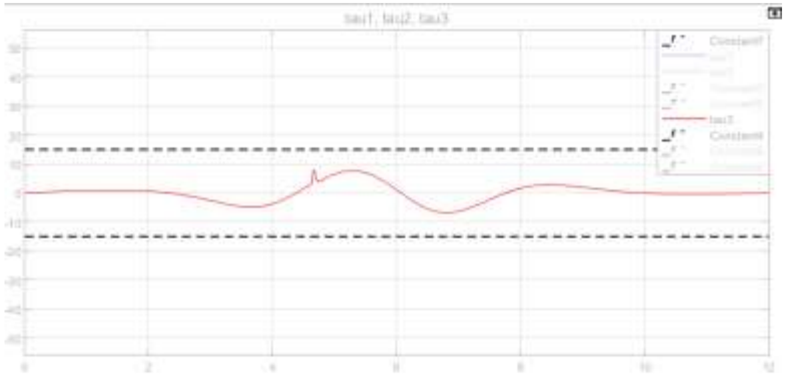
Torque Joint 1



Torque Joint 2



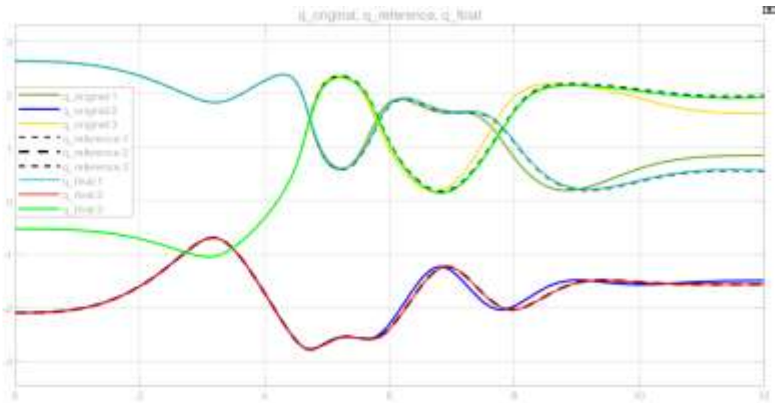
Torque Joint 3



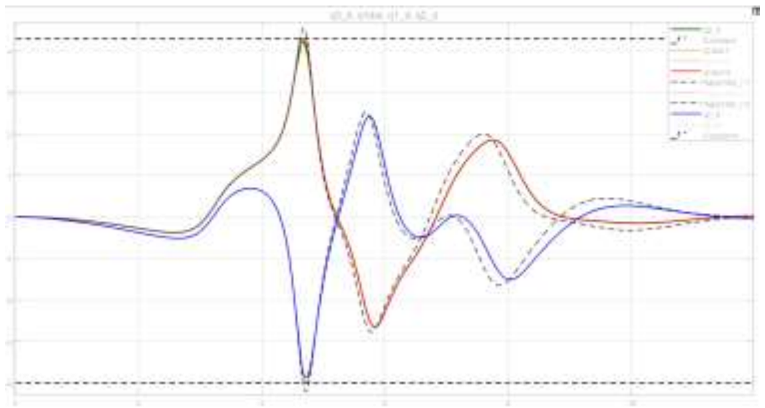
- As in the circular case we can see from the graphs that it is the **torque** of the **joint 1** that goes beyond the boundaries imposed.

Dynamics sinusoidal case - T=12s

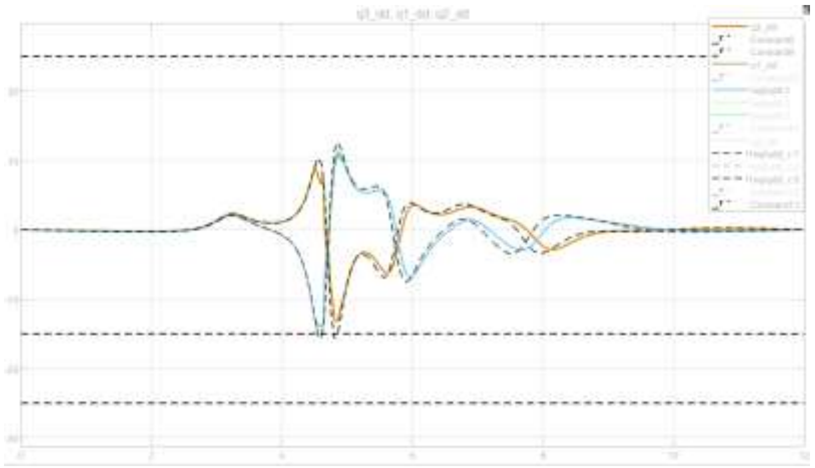
Position profiles



Velocity profiles



Acceleration profiles



- Even in this case the new control law is able to perform the scaled motion profiles with a **slight larger error** than before

Conclusions

- The experiments carried out have led to very satisfactory results in both cases (with and without considering the robot dynamics).
- We have proved the effectiveness of the online trajectory scaling method proposed: it activates when the boundaries are not respected and modulates the profiles in order to recover the delay introduced by the saturation.
- The position error obtained at the end of the simulations does not come from a lack of the algorithm. In fact, we have previously seen how this problem can be solved.
- Finally, the idea to consider the scaled profiles as the nominal case for the dynamic part and the choice of the control law (feedback linearization + PD +FFW) have been proved to be correct to demonstrate the effectiveness of the method.

Thanks for your attention!

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)

Mattia Castelmare 1815675
Patrizio Perugini 1844358