# Analysis of Inertia-Weighted Redundancy Resolution

Final project of Robotics II

Ermanno Bartoli 1794103 Francesco Argenziano 1957976

July 26, 2021

# Contents

# 1 Introduction

Given a $M$-dimensional task $r = f(q)$, where $q = [q_1...q_N]^T$ is the set of joint variables of the robot, we say that the robot is **redundant** for that specific task if $M < N$, i.e. we have more d.o.f than needed to solve that particular task.
When the robot is redundant, this redundancy can be exploited to achieve some additional goals in addition doing the task: we can use redundancy to avoid kinematic singularities in the joint space, we can use it to distribute joint velocities and accelerations in order to avoid having a single joint that carries all the load of the task, we can optimize motion time, and we can also use it to minimize the energy consumption or the needed motion torques.
So, what we want to do is to find the $q(t)$ such that the robot is able to realize the task $r(t) = f(q(t)\ \forall t$ and at the same time it achieves, exploiting the redundancy, some other quality criterion.
One way to seek for the solutions is to work at the differential level, by using optimization. There exist both *local* and *global* methods: the former looks for a solution that minimizes an objective function and it's typically found online, the latter looks for the whole trajectory $q(t)$ which optimizes some criterion over the whole time interval and it's typically found online. We will consider only local methods because they're easier, consisting usually in solving a Linear-Quadratic problem.
Among the local methods, we focus our attention on the **Jacobian-based methods**: we look for a solution to $\dot{r} = J(q)\dot{q}$ in the form $\dot{q} = K(q)\dot{r}$, where $J \in \mathbb{R}^{M\times N}$ and $K \in \mathbb{R}^{N\times M}$ $N > M$, and $K(q)$ is the generalized inverse of $J(q)$, i.e. it holds $J(q)K(q)J(q) = J(q)$. One common choice to $K(q)$ is using the **pseudoinverse** $J^{\#}(q)$.
This approach can be also extended to the second differential order, i.e. working at the level of **accelerations** instead of the level of **velocities**. In fact, by further differentiating $\dot{r} = J(q)\dot{q}$ we get $\ddot{r} = J(q)\ddot{q} + \dot{J}(q)\dot{q}$, which can be rearranged in the form $J(q)\ddot{q} = \ddot{r} - \dot{J}(q)\dot{q}$. Since $\ddot{r}$ is given at the time $t$ and so it is the pair $(q, \dot{q})$, the right-hand side of that equation can be rewritten in the form $J(q)\ddot{q} = \ddot{x}$, where $\ddot{x} \triangleq \ddot{r} - \dot{J}(q)\dot{q}$, which is a known vector.
So, in this form, the problem if formally equivalent to the one at the first differential order, with acceleration in place of velocity commands.
Once we have this acceleration-level description, we can also take into account the dynamic model of the robot, and this analysis will be the core of our work. In fact, the goal of this project is to study and analyze the performances of a 3R robot (both in presence and absence of gravity) that tries to solve a $M = 2$-dimensional task, while at the same time tries to minimize an objective function that depends on the inertia matrix of the robot raised to different powers $k$.

# 2 Theoretical Treatment

Now we will describe the problem in a more formal way.
We consider a 3R robot ($N = 3$), redundant to a M=2-dimensional task. The following

differential relations hold:

$$r = f(q) \qquad r \in \mathbb{R}^M, q \in \mathbb{R}^N$$

$$\dot{r} = J(q)\dot{q} \qquad J(q) = \frac{df(q)}{dq}, J \in \mathbb{R}^{M \times N}$$

$$\ddot{r} = J(q)\ddot{q} + \dot{J}(q)\dot{q} \qquad \dot{J}(q) = \frac{dJ(q)}{dt}$$

The last equation can alternatively be rewritten as:

$$J(q)\ddot{q} = \ddot{x} \qquad \ddot{x} \triangleq \ddot{r} - \dot{J}(q)\dot{q}$$

The robot dynamic model is defined as

$$M(q)\ddot{q} + c(q, \dot{q}) + g(q) = \tau$$

where:

- $M(q) \in \mathbb{R}^{N \times N}$ is the inertia matrix of the robot. It' symmetric and positive definite $\forall q$;
- $c(q, \dot{q}) \in \mathbb{R}^N$ are the centrifugal and Coriolis terms;
- $g(q) \in \mathbb{R}^N$ are the gravity terms;
- $\tau \in \mathbb{R}^N$ are the input torques, provided by the motors.

We can write it in a more compact way:

$$M(q)\ddot{q} + n(q, \dot{q}) = \tau \qquad n(q, \dot{q}) \triangleq c(q, \dot{q}) + g(q)$$

What we want to do is to solve the task while, at the same time, minimize the following objective function in the framework of LQ optimization.

$$\min H = \frac{1}{2}||\tau - \tau_0||^2_{M^{-k}} = \frac{1}{2}(\tau - \tau_0)^T M(q)^{-k}(\tau - \tau_0) \quad k = ... - 2, -1, 0, 1, 2, ...$$

A general LQ optimization problem is in the form

$$\min H(x) = \frac{1}{2}||x - x_0||^2 = \frac{1}{2}(x - x_0)^T W(x - x_0) \qquad \text{s.t. } Jx = y$$

and the solution to this problem is in the form

$$x = x_0 + W^{-1}J^T(JW^{-1}J^T)^{-1}(y - Jx_0)$$

Before we can apply this solution, we first have to rewrite our problem in order to go back to the general formulation. From now on we will drop the $q$ dependecies in order to simplify the notation. We call $V = M^{-k}$ and w substitute $\tau$ with the dynamic model. We get:

$$H = \frac{1}{2}(M\ddot{q} + n - \tau_0)^T V(M\ddot{q} + n - \tau_0) =$$

$$= \frac{1}{2}(\ddot{q} + M^{-1}(n - \tau_0))^T M^T V M(\ddot{q} + M^{-1}(n - \tau_0))$$

So, by comparing this formulation to the general one we get:

- $x = \ddot{q}$
- $W = M^T V M = M M^{-k} M = M^{-(k-2)}$
- $x_0 = -M^{-1}(n - \tau_0) = M^{-1}(\tau_0 - n)$
- $y = \ddot{r} - \dot{J}\dot{q}$

Now we can finally apply the general solution and we get:

$$\ddot{q} = M^{-1}(\tau_0 - n) + M^{k-2}J^T(JM^{k-2}J^T)^{-1}(\ddot{r} - \dot{J}\dot{q} - JM^{-1}(\tau_0 - n))$$

We express now in terms of command torques:

$$M^{-1}(\tau - n) = M^{-1}(\tau_0 - n) + M^{k-2}J^T(JM^{k-2}J^T)^{-1}(\ddot{r} - \dot{J}\dot{q} - JM^{-1}(\tau_0 - n))$$
$$\tau = \tau_0 + MM^{k-2}J^T(JM^{k-2}J^T)^{-1}(\ddot{r} - \dot{J}\dot{q} - JM^{-1}(\tau_0 - n))$$
$$\tau = \tau_0 + MJ^{\#}_{M^{-(k-2)}}(\ddot{r} - \dot{J}\dot{q} - JM^{-1}(\tau_0 - n))$$

where $J^{\#}_{M^{-(k-2)}}$ is the **weighted pseudoinverse** of $J$, i.e.

$$J^{\#}_{M^{-(k-2)}} \triangleq M^{k-2}J^T(JM^{k-2}J^T)^{-1}$$

We rearrange terms one more time in order to isolate the damping factor and we get to the final expression of our solution

$$\tau = MJ^{\#}_{M^{-(k-2)}}(\ddot{r} - \dot{J}\dot{q} + JM^{-1}n) + M(I - J^{\#}_{M^{-(k-2)}}J)M^{-1}\tau_0$$

In this way, we can simply recover also the case where the damping factor is not present. In fact, when $\tau_0 =$ the formula simply becomes:

$$\tau = MJ^{\#}_{M^{-(k-2)}}(\ddot{r} - \dot{J}\dot{q} + JM^{-1}n)$$

and in both cases, we can simply consider the case with gravity and without gravity by putting $n = c + g$ in the first case and $n = c$ in the second case. The final expression that we've got allow us also to make a nice consideration when the damping factor is present. In fact, since we are working with input torques, what we see is that first the damping factor is transformed into a joint acceleration by premultiplying it with the inverse of the inertia matrix, than it is projected into the null space of the Jacobian, and finally it is transformed back to a torque by premultiplying it with the inertia matrix.
One last observation we have to make is that, since we want to be sure that our robot follows the task, we have to include also a controller in order to recover task error during execution. We can do this by considering the control law:

$$\ddot{r} = \ddot{r}_d + K_D(\dot{r}_d - \dot{r}) + Kp(r_d - r)$$

where $K_D > 0$ and $K_P > 0$ are diagonal matrix gains. Regarding the damping factor, we've chosen it as

$$\tau_0 = -M\hat{K}_D\dot{q}$$

where $\hat{K}_D > 0$ is a $3 \times 3$ extension of the $K_D$ diagonal matrix gain, and this damping factor help us to stabilize the self motion in the null space.

# 3 Simulation Setup

## 3.1 3R robot

First of all we have determined the structure of the robot by defining the Denavit-Hartenberg parameters.

### 3.1.1 Robot structure

| | $\alpha$ | a | d | $\theta$ |
|---|---|---|---|---|
| link 1 | 0 | $a_1 = L$ | 0 | $q_1$ |
| link 2 | 0 | $a_2 = L$ | 0 | $q_2$ |
| link 3 | 0 | $a_3 = L$ | 0 | $q_3$ |

Table 1: In this table the DH parameters of the robot are reported.

| | $L$ | $\mathrm{dc}_i$ |
|---|---|---|
| link 1 | 1 | 1/2 |
| link 2 | 1 | 1/2 |
| link 3 | 1 | 1/2 |

Table 2: In this table are reported the lengths of the links and the distances of the center of masses

### 3.1.2 Dynamic Parameters

| | $m$ | $I$ |
|---|---|---|
| link 1 | 10 | $1/12 \cdot m1 \cdot L^2$ |
| link 2 | 10 | $1/12 \cdot m2 \cdot L^2$ |
| link 3 | 10 | $1/12 \cdot m3 \cdot L^2$ |

Table 3: In this table are reported the masses and the intertia values for the links

obs: Since the masses and the lengths of the links are the same, also the inertia is the same for every link

We set the diagonal Gain matrices $Kp = \begin{bmatrix} 400 & 0 \\ 0 & 400 \end{bmatrix}$ and $Kd = \begin{bmatrix} 40 & 0 \\ 0 & 40 \end{bmatrix}$

## 3.2 Simulink Model

The Simulink model can be divided in 2 parts, the first one has the task of generate the trajectory (more details are descripted in *generation of trajectories* section), while the

second part, which is the biggest one, performs all the computations and the integrations

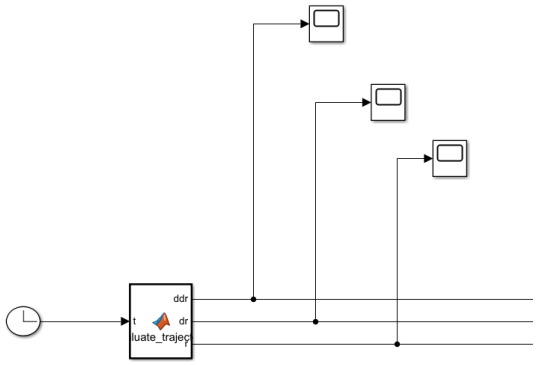


Figure 1: First part of Simulink Model

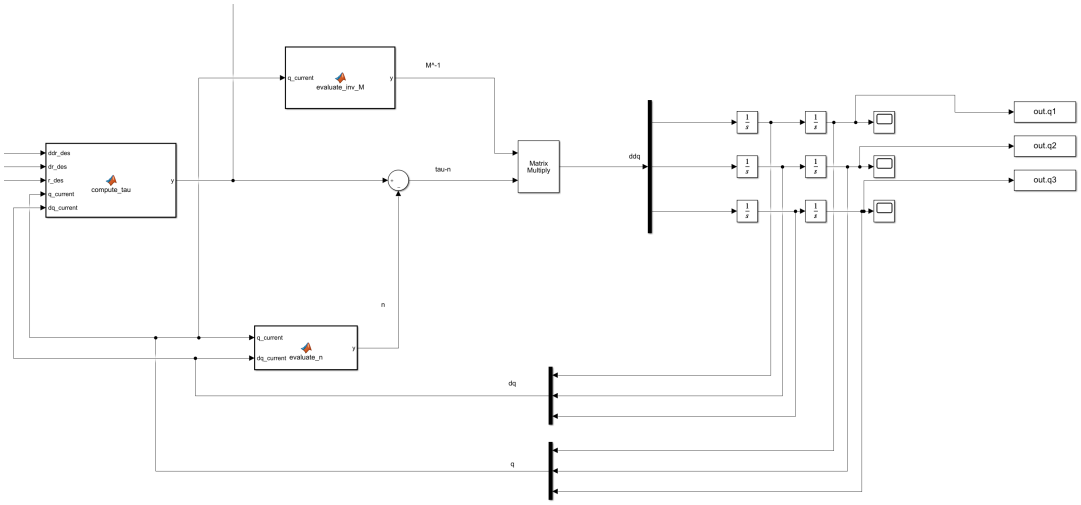


Figure 2: Second part of Simulink Model

### 3.3 Matlab Robotics System Toolbox

Matlab Robotics System Toolbox™ provides tools and algorithms for designing, simulating, and testing manipulators, mobile robots, and humanoid robots. The toolbox includes algorithms for collision checking, trajectory generation, forward and inverse kinematics. Anyway, since we were considering a 3R robot the forward kinematics was computed by hand, and we used it only to plot the trajectory that the robot had to follow and to display the motion.
Robotics System Toolbox also includes a library of commercially available industrial robot models that you can import, visualize, and simulate, but it's not the case of the 3R robot which was created by adding manually rigid bodies for the links, and rigidBodyJoints for the joints. Finally, we added collisionCylinders on the links to make the robot looks more realistic.

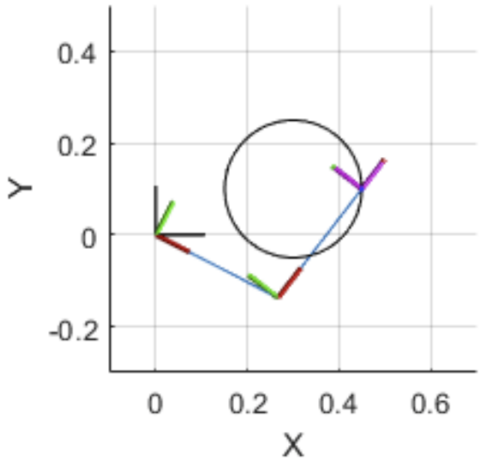


Figure 3: 2R robot **without** collisionCylinders

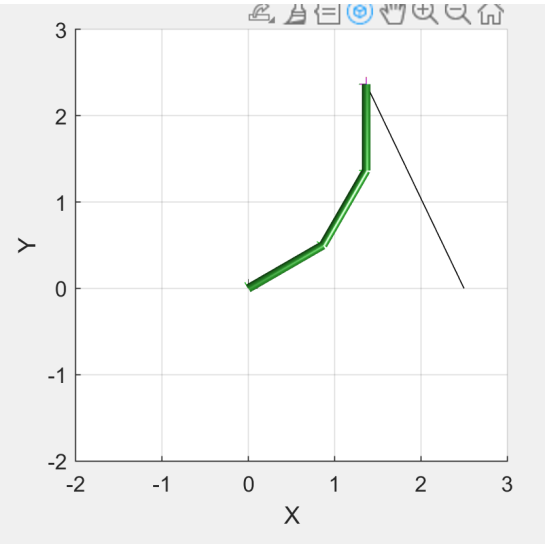


Figure 4: Our 3R robot **with** collisionCylinders

# 4 Simulations

In this section are showed, explained and commented a series of simulations that we handled, in order to check the behavior of the Robot.
As mentioned above, the robot we analyzed is a 3R robot; we considered all links of *equal length* ( $= 1$) and masses *uniformly distributed* over the entire link.
Moreover, in order to discuss as many different situations as possible, we generated different trajectories and we let our robot start from different configurations.
For our simulations we used Simulink to perform the computations, then we showed our results over some specific plots, and we plotted the motions by using Matlab Robotics toolkit, in a such way that for every simulation, the plot of the tau variables, the joint variables, and the video of the motion is available.
For every trajectory, we did several simulations with the same simulation parameters, just changing the value of $k$, so that also the power of the matrix used to weight the pseudoinverse changes. In this way, we could compare the robot behavior in the same situations and analyze which values of $k$ are preferable.
Some values of $k$ leads to very particular solutions:

- $k = 0$ it's the **minimum torque norm** solution, it's typically good for short trajectories, but for long trajectories it leads to torque explosions/oscillations (*whipping effect*);
- $k = 2$ it's the **minimum squared inverse inertia weighted torque norm** solution, it presents good performances in general;
- $k = 1$ it's the **minimum inverse inertia weighted torque norm** solution, that has a nice interpretation: this is the dynamic torque that needs to be applied when the force on the right-hand side of the equation (that we get by substituting $k = 1$ in the general solution, and that is premultiplied by $J^T$) is exactly the force that will allow us to execute the desired acceleration of a trajectory.

## 4.1 Generation of trajectories

The first relevant part of the simulations deals with the generation of the trajectory and it consists of the following steps.

- We defined different paths $r(s)$
- We defined the timing law $s(\tau)$, where $\tau$ is the normalized time, $\tau = t/T$
- We computed the first order derivative of $\dot{s} = \frac{ds}{dt}$
- We computed the second order derivative of $\ddot{s} = \frac{d\dot{s}}{dt}$
- We computed the first order derivative of $\dot{r} = \frac{dr}{ds} \cdot \dot{s}$

- We computed the second order derivative of $\ddot{r} = \frac{dr}{ds} \cdot \ddot{s} + \frac{d^2r}{\delta s^2} \cdot \dot{s}^2$

obs: We considered trajectories described by the following cubic in space:

$$s(\tau) = s_0 + \Delta s(a \cdot \tau^3 + b \cdot \tau^2 + c \cdot \tau + d), \quad \text{where} \quad \tau \in [0, 1]$$

Since we considered only rest-to-rest motions ($v(0) = v(1) = 0$), we get the following equalities: $c = 0$; $a + b = 1$, and $3 \cdot a + 2 \cdot b = 0$ which lead to the result:

- $a = -2$
- $b = 3$
- $c = 0$
- $d = 0$

The final cubic is semplified as follows:

$$s(\tau) = -2 \cdot \tau^3 + 3 \cdot \tau^2$$

$$\dot{s} = \frac{(6 \cdot t)}{T^2} - \frac{6 \cdot t^2}{T^3}$$

$$\ddot{s} = \frac{6}{T^2} - \frac{12 \cdot t}{T^3}$$

### 4.2 Results on a linear path

The first path we chose for the simulations is a linear path between a starting point and an ending point.
The initial configuration of the 3R robot is showed in the following table:

| **q1** | **q2** | **q3** |
|---|---|---|
| pi/6 | pi/6 | pi/6 |

So we chose the starting point in the exact position of the end-effector in the initial configuration.

$$start = \begin{bmatrix} cos(q1) + cos(q1 + q2) + cos(q1 + q2 + q3) \\ sin(q1) + sin(q1 + q2) + sin(q1 + q2 + q3) \end{bmatrix}$$

$$= \begin{bmatrix} cos(pi/6) + cos(pi/6 + pi/6 + cos(pi/6 + pi/6 + pi/6) \\ sin(pi/6) + sin(pi/6 + pi/6) + sin(pi/6 + pi/6 + pi/6) \end{bmatrix} = \begin{bmatrix} 1.3660 \\ 2.3660 \end{bmatrix}$$

Then we simply chose the final point with the following coordinates: $end = \begin{bmatrix} 2.5 \\ 0 \end{bmatrix}$.
Finally we had the equations:

$$r(s) = start + s \cdot (finish - start)$$
$$\dot{r} = (finish - start) \cdot \dot{s}$$
$$\ddot{r} = (finish - start) \cdot \ddot{s}$$

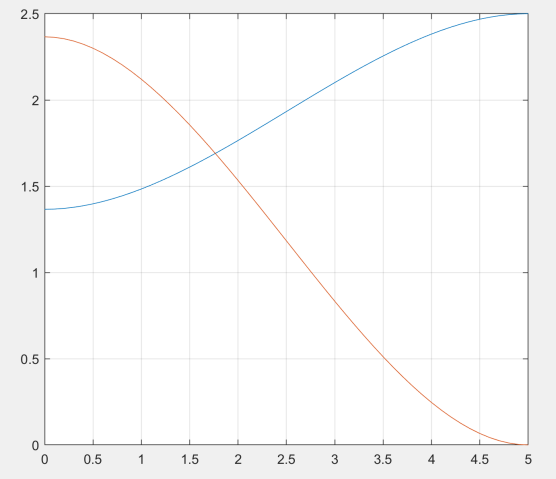


Figure 5: $r(t)$ , where $t \in [0, 5]$

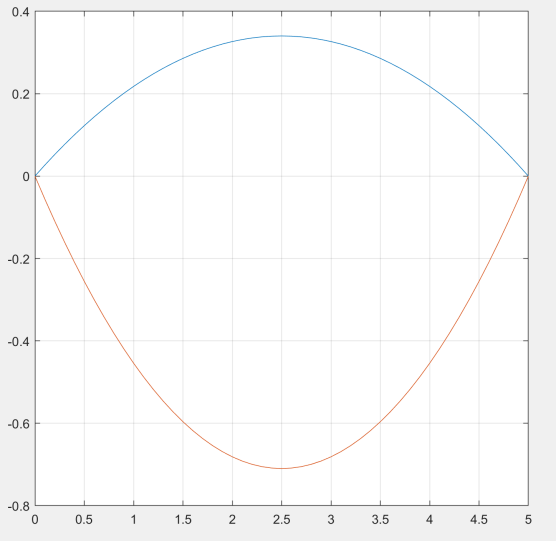


Figure 6: $\dot{r}(t)$ , where $t \in [0, 5]$

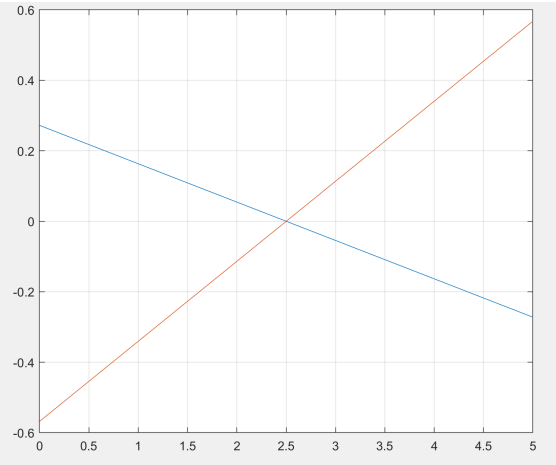


Figure 7: $\ddot{r}(t)$ , where $t \in [0, 5]$

### 4.2.1 Absence of gravity term

In this first set of results, the situation is quite stable, because we're almost in ideal conditions, in fact there is no gravity, and the path is really simple since it's linear.
Anyway some considerations can be made also at this point. As described in the previous section, not all values of k perform in the same way, in fact looking at figures 8, 9, 10, 11 the differences are evident. For k=1 and k=2 the tau values keep in a small range [-25; 15], while for values k=0 and k=-1 the are some spikes that make the $\tau$ values jump over 4000. This phenomenon will be much more evident in a limit case that we'll show in the *conclusion* section.

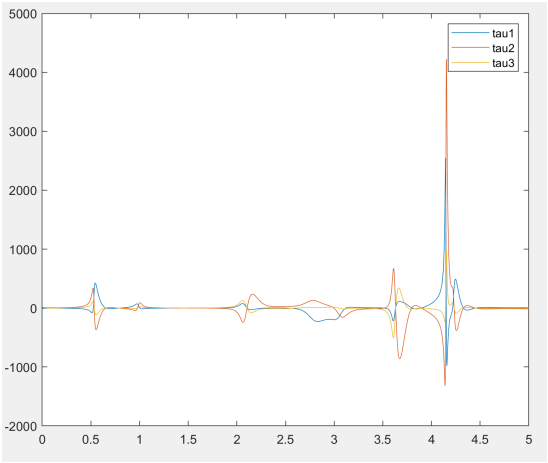


Figure 8: $\tau$ values for k=-1

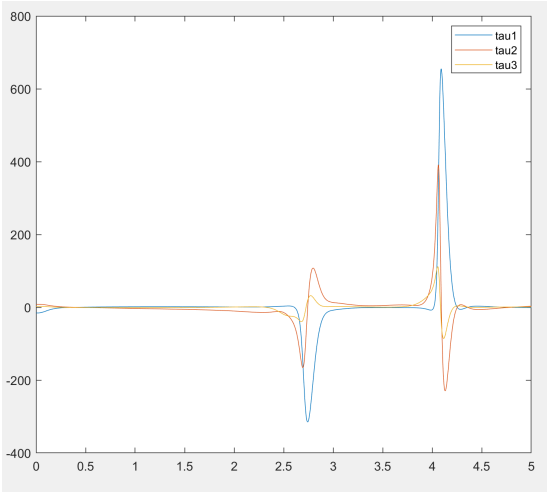


Figure 9: $\tau$ values for k=0

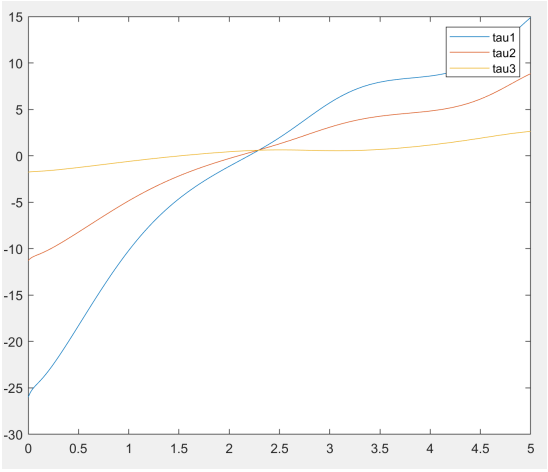


Figure 10: $\tau$ values for k=1

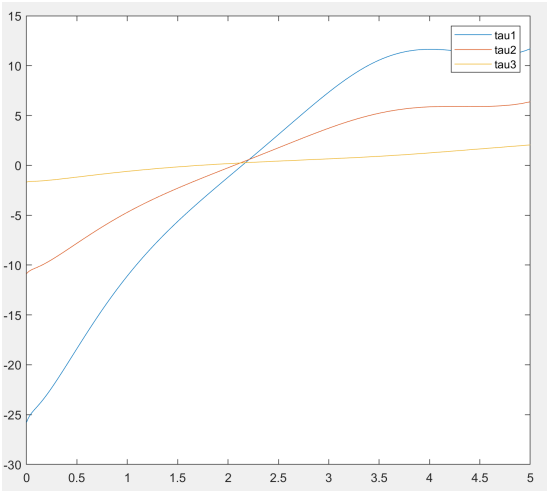


Figure 11: $\tau$ values for k=2

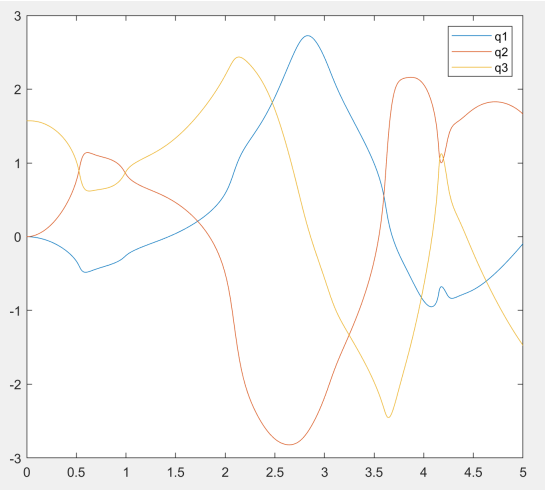


Figure 12: joint values for k=-1

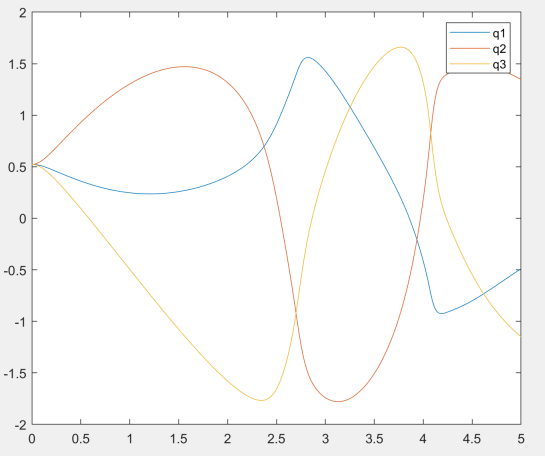


Figure 13: joint values for k=0

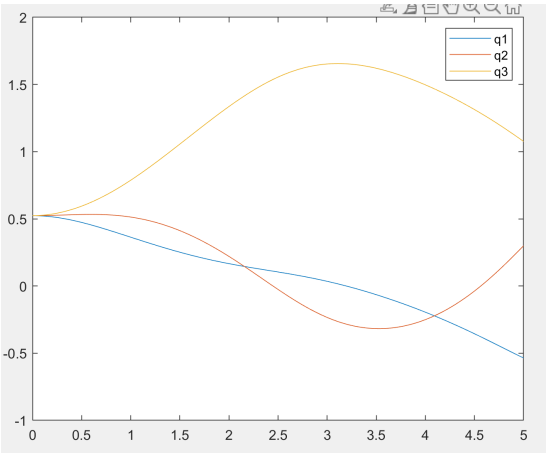


Figure 14: joint values for k=1

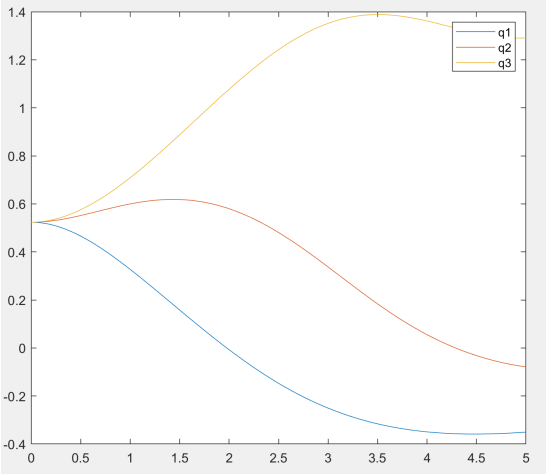


Figure 15: joint values for k=2

### 4.2.2 Presence of gravity term

With the presence of the gravity the situation is completely different. In fact we switch from an environment which is almost in ideal conditions to another one more similar to the real one. Also the figures are more complex in this set of results, and here is fundamental choose the right value of k, since values like $k = -1$ (figure 16) and $k = 0$ (figure 17) lead to values of $\tau$ which oscillate and reach high values.
Another aspect that has to be mentioned, is that with high values of $k$ (both positive and negative), the path is not well followed, since there are some moments of reconfiguration which lead to a disallignment from the trajectory.
Moreover this oscillation (both in $\tau$ values, but also in joint values) will be discussed more in depth when we'll introduce a damping factor which will be really helpful for this problem.

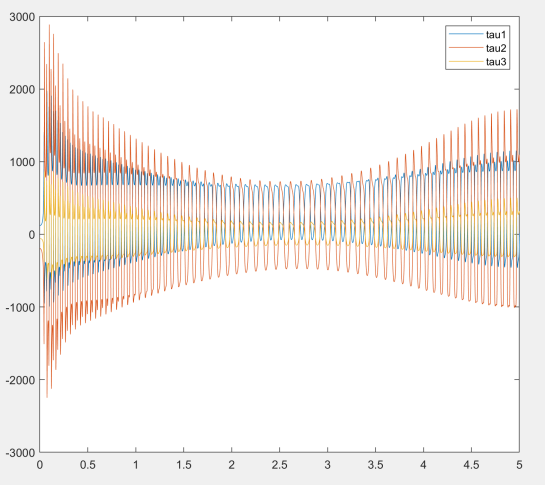


Figure 16: $\tau$ values for k= -1

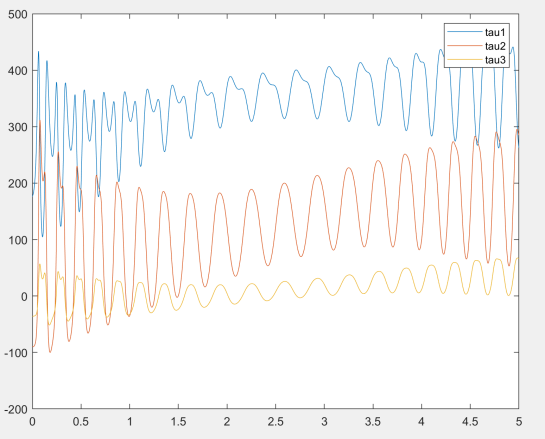


Figure 17: $\tau$ values for k=0

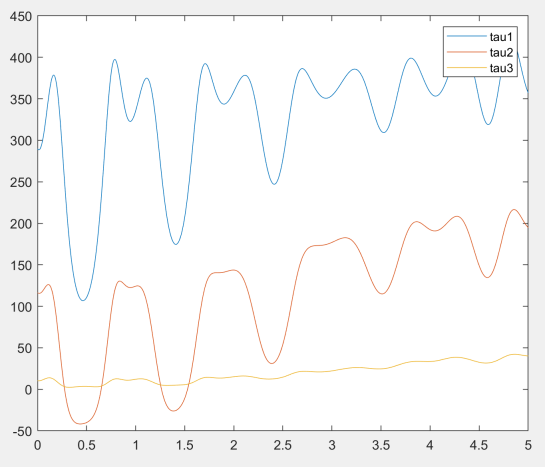


Figure 18: $\tau$ values for k=1

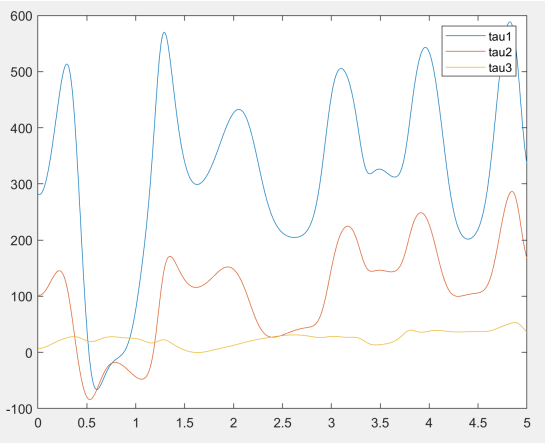


Figure 19: $\tau$ values for k=2

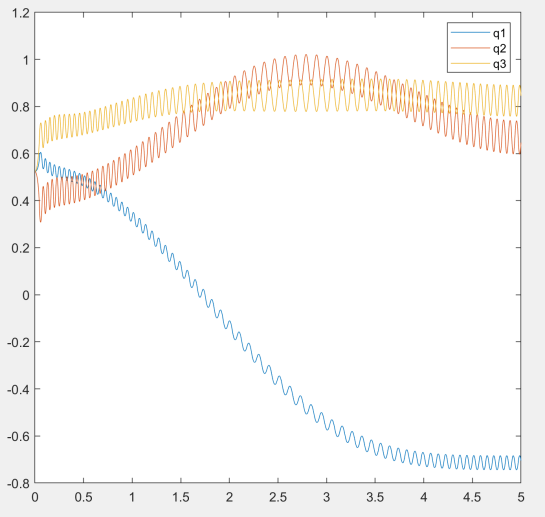


Figure 20: joint values for k=-1

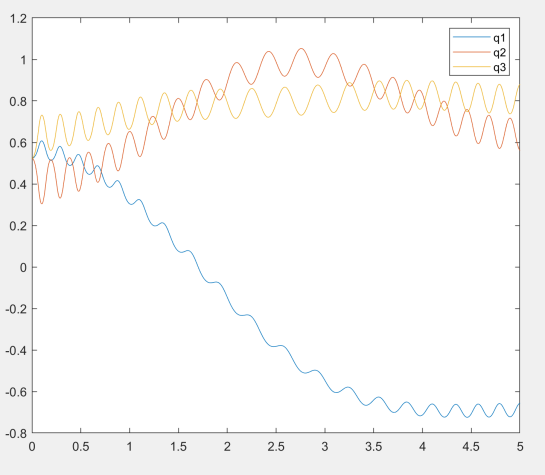


Figure 21: joint values for k=0

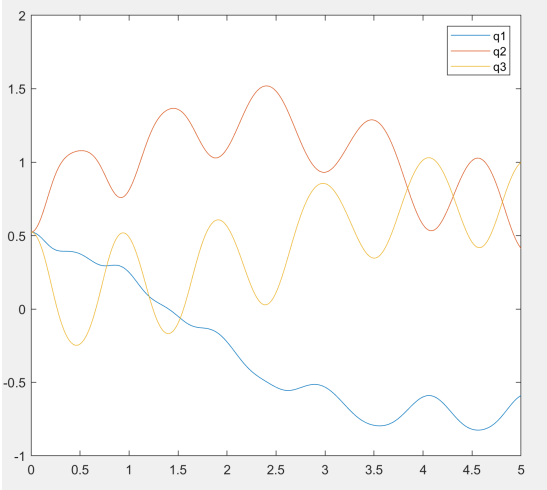


Figure 22: joint values for k=1

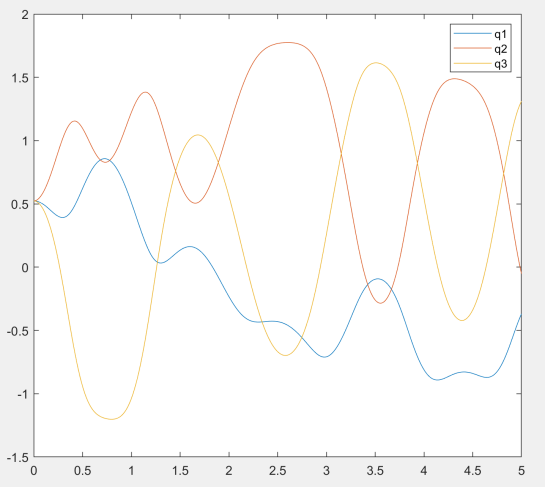


Figure 23: joint values for k=2

### 4.3 Results on a circular path

The second path we chose for the simulations is a circular path with center in $\begin{bmatrix}1.5\\1\end{bmatrix}$, and a radius of 0.5.
The initial configuration of the 3R robot for this path is showed in the following table:

| **q1** | **q2** | **q3** |
|---|---|---|
| 0 | 0 | $\pi/2$ |

*obs:* We positioned the end-effector at the starting point of the path in order to avoid high values of tau at the beginning of the simulation.

For this path we obtained the following equations:

$$r(s) = center + radius \cdot \begin{bmatrix} cos(2 \cdot s \cdot \pi) \\ sin(2 \cdot s \cdot \pi) \end{bmatrix}$$

$$\dot{r} = 2 \cdot \pi \cdot radius \cdot \begin{bmatrix} -sin(2 \cdot s \cdot \pi) \\ cos(2 \cdot s \cdot \pi) \end{bmatrix} \cdot \dot{s}$$

$$\ddot{r} = (2 \cdot \pi)^2 \cdot radius \cdot \begin{bmatrix} -cos(2 \cdot s \cdot \pi) \\ -sin(2 \cdot s \cdot \pi) \end{bmatrix} \cdot \ddot{s}$$

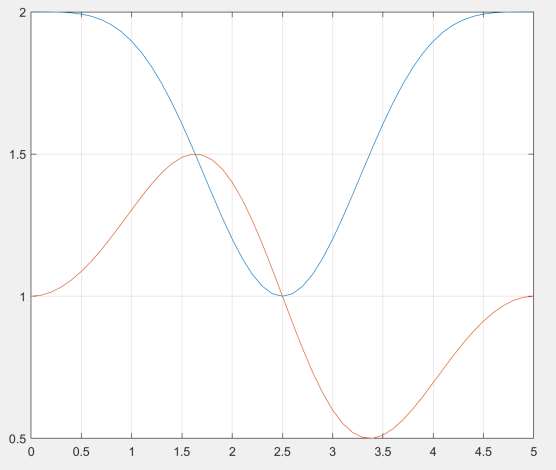


Figure 24: $r(t)$ , where $t \in [0, 5]$

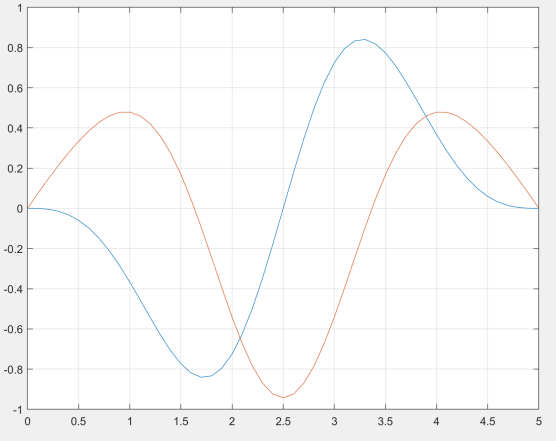


Figure 25: $\dot{r}(t)$ , where $t \in [0, 5]$

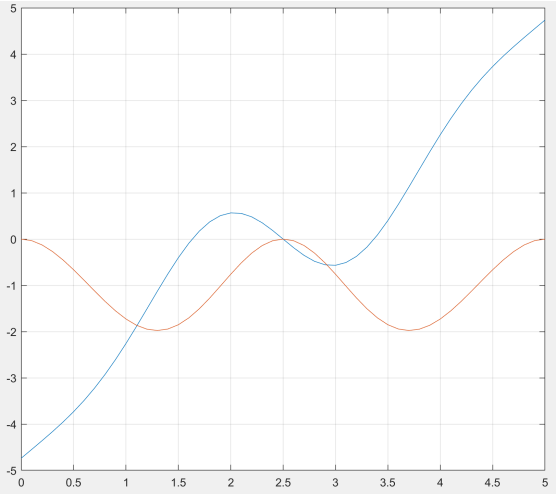


Figure 26: $\ddot{r}(t)$ , where $t \in [0, 5]$

### 4.4 Absence of gravity

In this set of results, the situation becomes more complex, because the trajectory is no longer a simple line, but a circular trajectory and it's harder to follow. The gravity is not present, in fact the values of $\tau$ are quite smoothed and specially for k=1 and k=2 they don't reach high values.
As in the previous results, k=0 and k=-1 cause spikes in the values of tau and they also reach high value, but more problems will come with the presence of gravity.

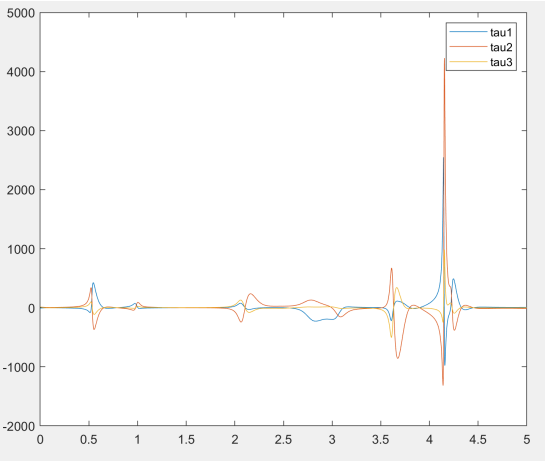


Figure 27: $\tau$ values for k=-1

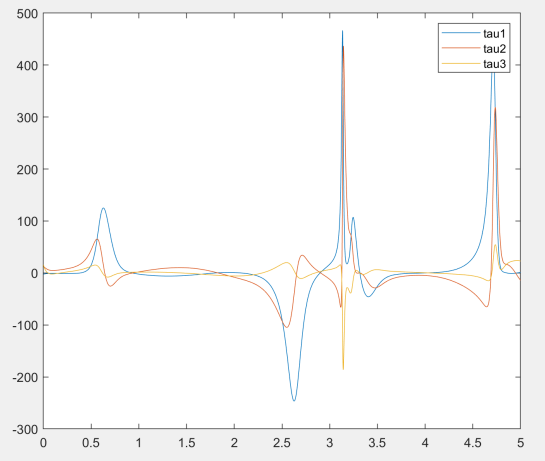


Figure 28: $\tau$ values for k=0

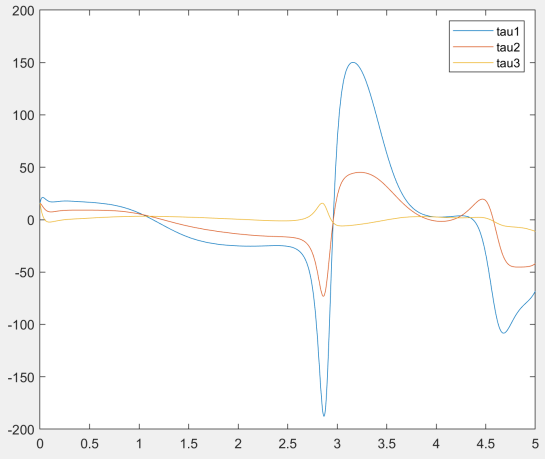


Figure 29: $\tau$ values for k=1

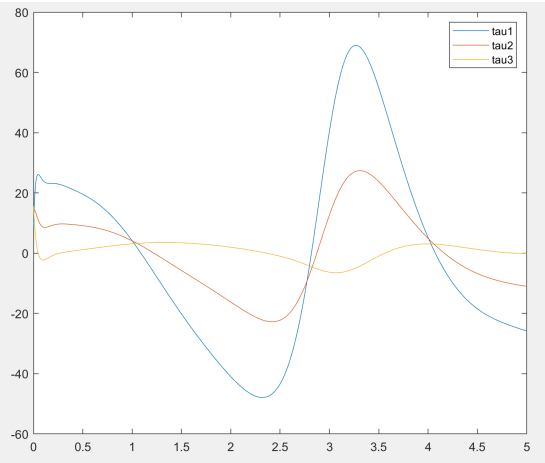


Figure 30: $\tau$ values for k=2

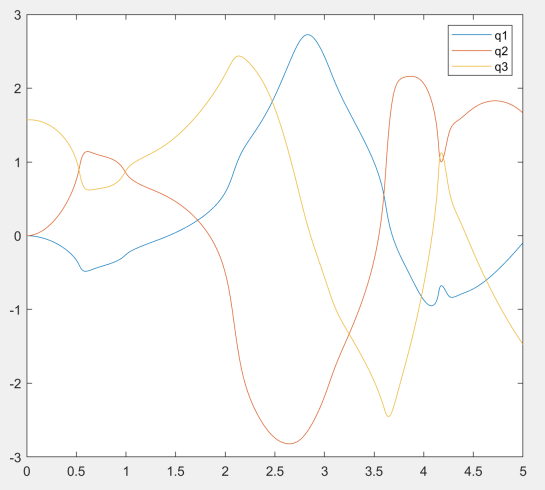


Figure 31: joint values for k=-1

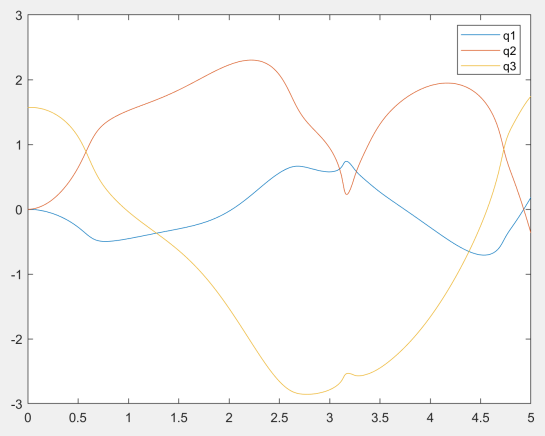


Figure 32: joint values for k=0

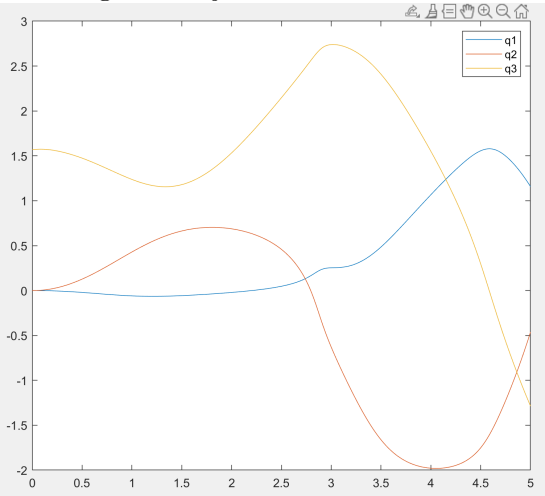


Figure 33: joint values for k=1

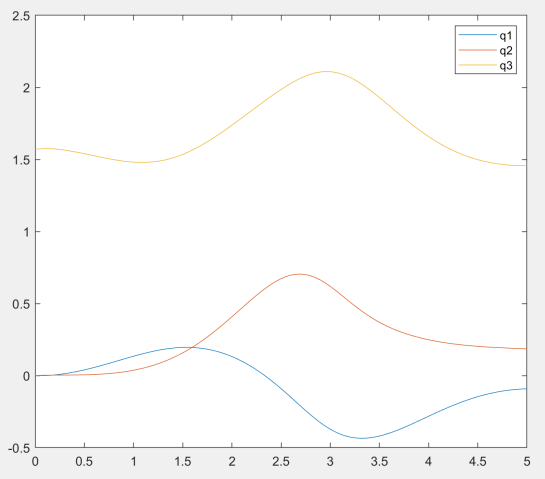


Figure 34: joint values for k=2

### 4.4.1 Presence of gravity

This set of results higlights some problems included with the presence of the gravity. In figures 35, 36, 37, 38 are showed the values of $\tau$ for these results.
What is easily noticeable is the complexity of the graphs, and the oscillatory trend of the functions.
In all the results, although evident oscillations in $\tau$ values and joint values (which are visible also during the simulations), the robot is able to following the circular path, so the solution obtained from the theory are consistent with the results.

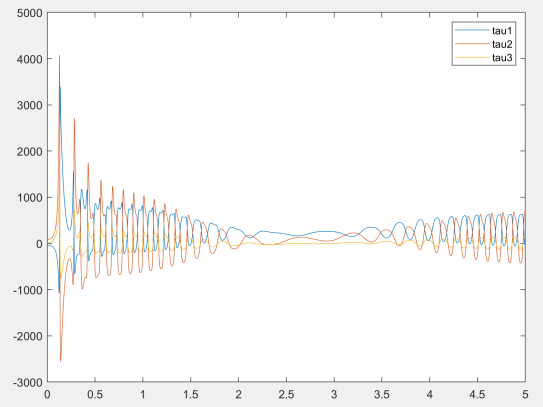


Figure 35: $\tau$ values for k=-1

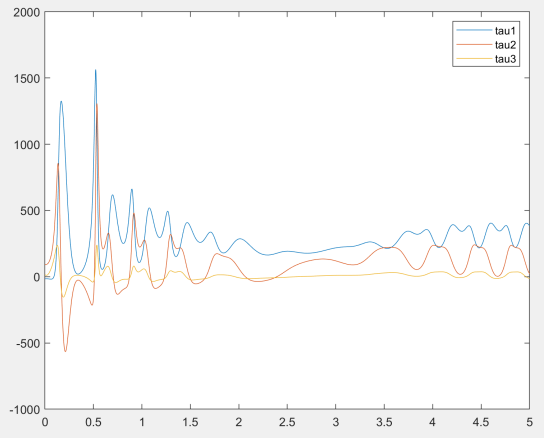


Figure 36: $\tau$ values for k=0

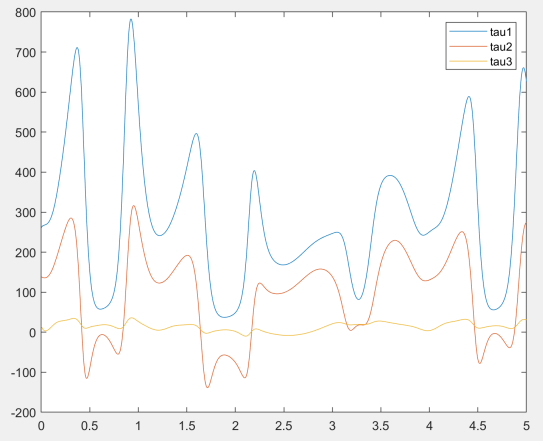


Figure 37: $\tau$ values for k=1

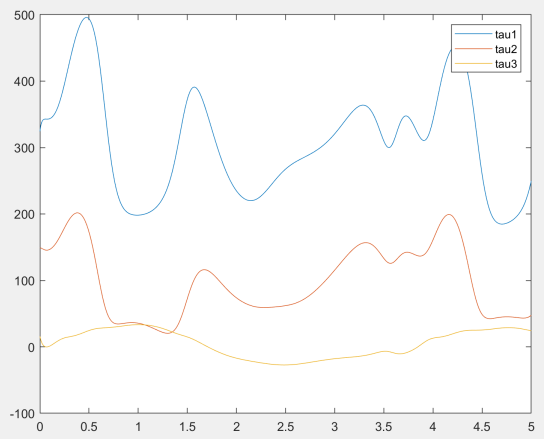


Figure 38: $\tau$ values for k=2

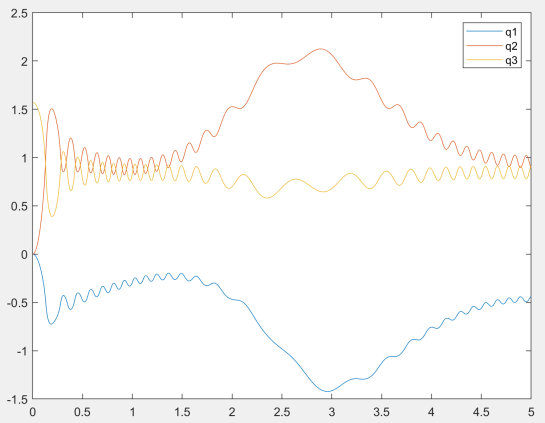


Figure 39: joint values for k=-1

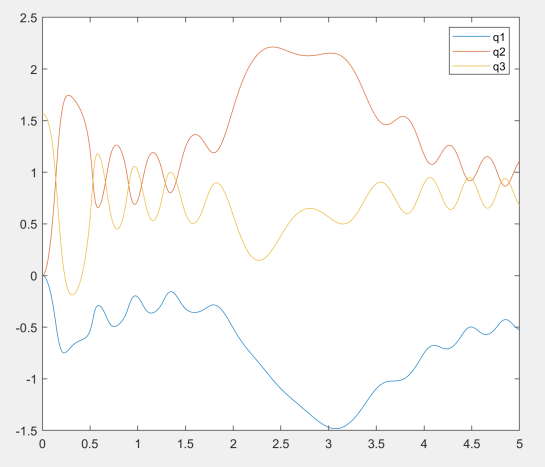


Figure 40: joint values for k=0

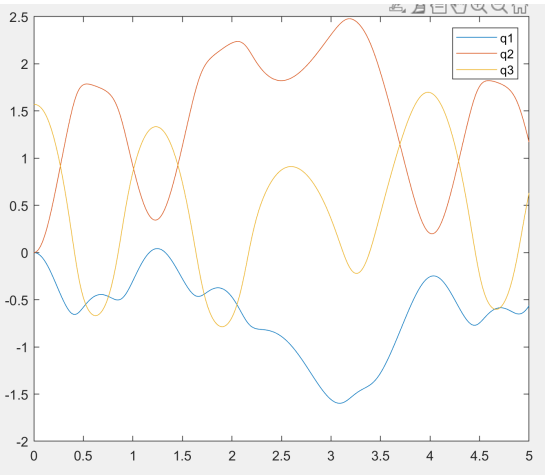


Figure 41: joint values for k=1

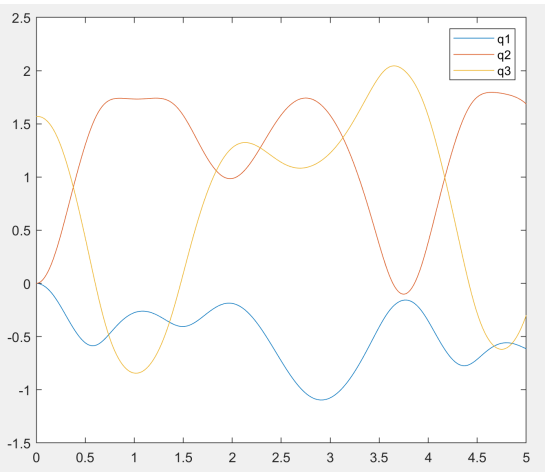


Figure 42: joint values for k=2

# 5 Damping factor

The results that we've obtained highlight some problems when the robot is put in an environment which is not in ideal conditions or when the task becomes enough complex; and these problems consist on oscillation trends and high values reached by $\tau$.
In order to analyze more in depth this problems, we built another path in such way that, once we noticed the presence of these problems, we could take some actions to solve them.
In the following result the robot was asked to hold the end-effector stuck in a point.
This task, which is trivial without the presence of gravity, becomes more interesting with the presence of it, because at every instant a $\tau$ which contrast the gravity have to be generated, and the end-effector has to be held to the point.

The goal point has coordinates $\begin{bmatrix}1.3660\\2.3660\end{bmatrix}$, the same starting point of the linear path, so also the initial configuration is the same.

The results we got is showed in the following graphs:

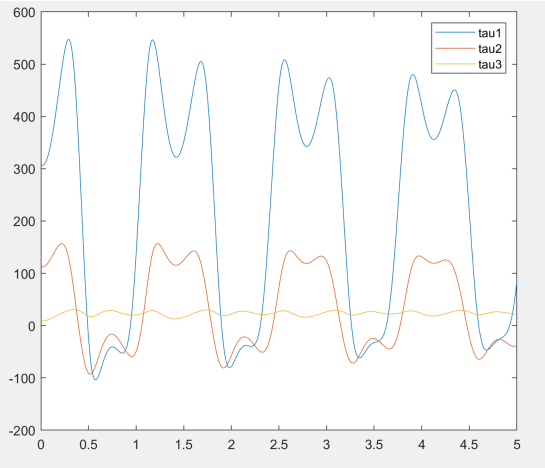


Figure 43: $\tau$ values for k=2

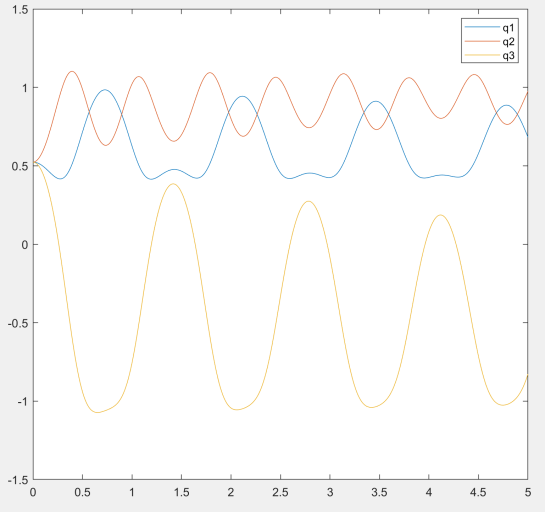


Figure 44: joint values for k=2

The graphs show some oscillations which, of course, are not needed for such a simple task. These oscillations are due to the redundancy of the robot for this task, because during all the simulation the end-effector is kept on the goal point, but the joints continue moving without a reason.
These are the reasons which convinced us to add a damping factor (described in the theory section), and after the addition of it what we expected was a totally different graph, without all these useless oscillations.

Finally the following result confirmed our intuitions.
The following figures show the result of the simulation.

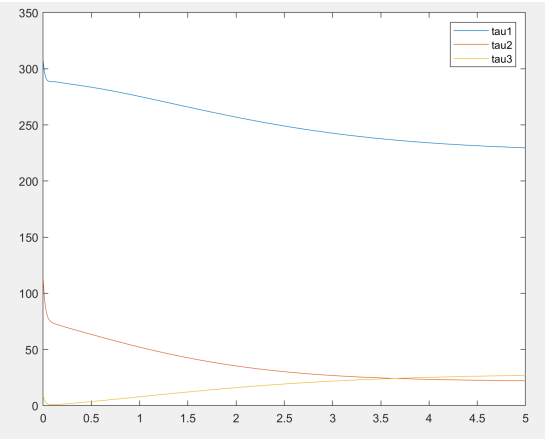


Figure 45: $\tau$ values for k=2

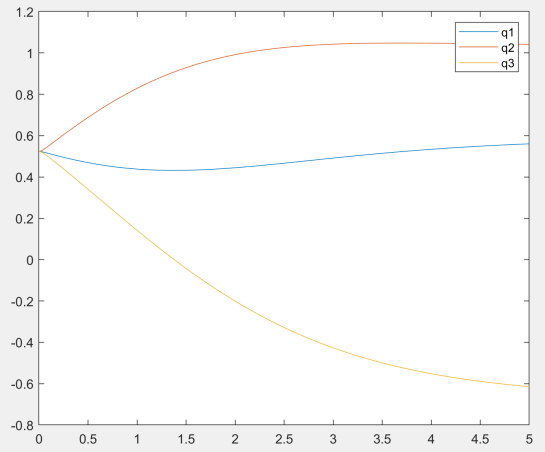


Figure 46: joint values for k=2

Figure 43 and figure 45 show 2 completely different graphs for the same task, of course the one in figure 45 is preferrable because it completely eliminate all the oscillations of the previous result.
A video of these 2 results just described is available on YouTube (1)

# 6 Conclusions and final comments

All the results we have obtained are extremely significative. In general, the presence of gravity affects really much the behavior of the robot in terms of $\tau$ values and oscillations because, although the path is correctly followed, high values of $\tau$ are reached.
The more the environment gets complex, the more important is the addition of a damping factor, which has the benefits just demonstrated over specific simulations.
One more observation is needed to complete the analysis. We've showed the results of 4 values of k, indicating k=2 as the preferred value, and k=-1 as the worst. We also hinted that more negative values of k can't lead to acceptable results (Figures 47, 48). Anyway it's not clear that this holds also for more positive values of k.
We observed that something tricky happens when high values of k are chosen (Figures 49, 50).
For the last simulation we tried to put the robot in the simplest environment possible (no gravity and linear path); here the values of tau seems to be close to 0, except for a single spike, so it seems to be a normal plot. Also the joints seem to move in a normal way. Anyway the robot does not follow the trajectory in any moment of the simulation. This show how also high (positive) values of k don't produce good results.

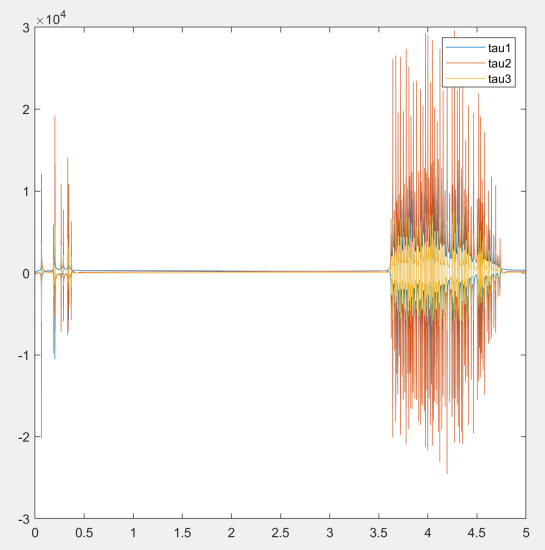


Figure 47: $\tau$ values for k=-3

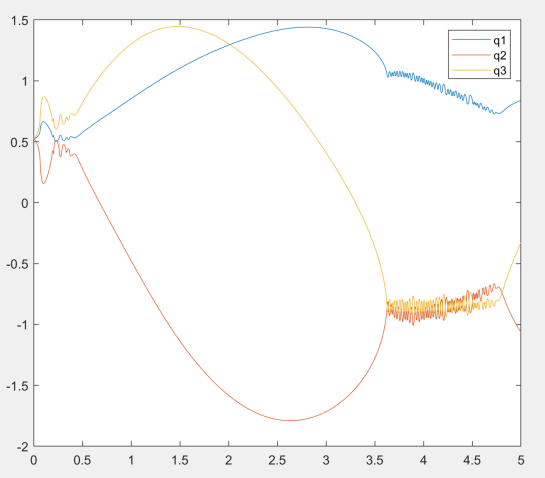


Figure 48: joint values for k=-3

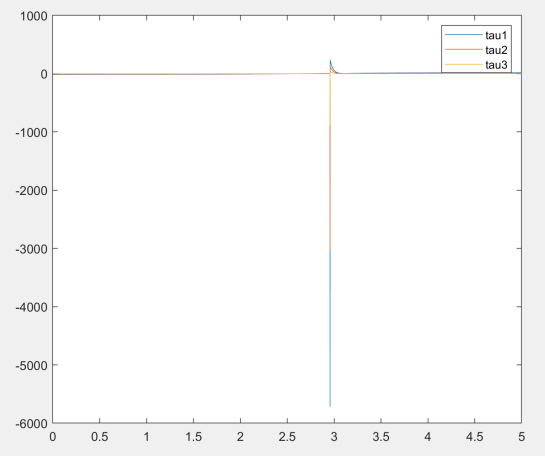


Figure 49: $\tau$ values for k=20

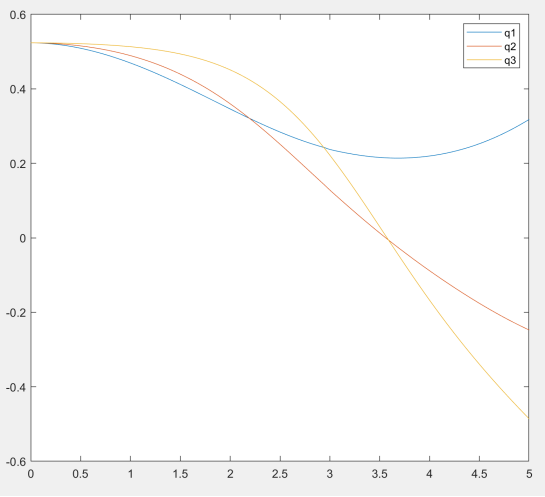


Figure 50: joint values for k=20

## References

[1] YouTube video: Benefits of a damping factor on a 3R robot under the effect of gravity,
`https://www.youtube.com/watch?v=81U0OS6AZ08`

[2] Prof. Alessandro De Luca slides for course of Robotics I,
`http://www.diag.uniroma1.it/deluca/rob1_en.php`

[3] Prof. Alessandro De Luca slides for course of Robotics II,
`http://www.diag.uniroma1.it/~deluca/rob2_en.php`