Robotics II
Final Project

Marco Pennese 1749223, Veronica Vulcano 1760405

Contents

1 Introduction

The goal of our work is to perform the dynamic identification of a robot, estimating its dynamic parameters/coefficients by solving an optimization problem.
As we know, it is very important to have an accurate dynamic model because it allows us to implement better control laws for achieving many tasks.

In particular, we have dealt with the problem of torque sign lack, as in some real robot (eg. KUKA KR5). In fact, when we know only the absolute value of torque and we don't have the knowledge about the torque sign, we can't use traditional methods for dynamic identification.
For this reason, we have developed an algorithm able to estimate torque signs so that we could proceed with the identification using well known strategies.

In our work, we first studied a simple 1R robot under gravity; then, we moved to a more complex robot, a 3R spatial robot. For both of them, we are now reporting the procedure we followed, the simulations and the results of our experiments.

The repository of our work is available here `https://github.com/MarcoPenne/Robotics_Project`.

2 1R arm under gravity

We start our work by focusing on a simple robot arm (pendulum) under gravity.

2.1 Dynamic model

The dynamic model of this kind of robot is:

$$u = gmd\sin(q) + (I + md^2)\ddot{q} = \begin{bmatrix}\sin(q) & \ddot{q}\end{bmatrix}\begin{bmatrix} gmd \\ I + md^2 \end{bmatrix}$$

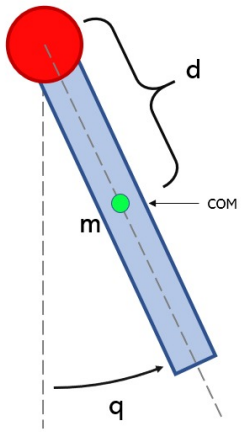

Figure 1: Robot arm model: m is the mass of the link, d is the position of the center of mass, I is the moment of inertia when rotating around the center of mass

We generate some trajectories from which we can compute the torques and build the Y matrix for a certain time instant. Then, we stack all the Y in a unique matrix $\overline{Y}$ (regressor matrix) and we do the same also for the measured torques obtaining a vector $\overline{\tau}$. $\overline{Y}$ and $\overline{\tau}$ are the inputs of the PBRP algorithm.

2.2 PBRP algorithm

If we had torque signs, we would apply directly the Penalty-Based Parameters Retrieval (PBRP) algorithm. The aim of this algorithm is to get a physically consistent set of parameters; in order to do so, we need to solve the following optimization problem:

$$\min_{p_k} \phi(p_k) = \|\overline{Y}\pi(p_k) - \overline{\tau}\|^2$$

where $\overline{Y}$ is the stacked regressor, $\overline{\tau}$ is the stacked torque measurements and $\pi(p_k)$ is the coefficients vector computed from the current parameters vector p_k.

In order to obtain a physically consistent set of parameters, we should put constraints on the mass and on the inertia. Since we are dealing with a 1 dof robot, the only constraints we have is that both mass and inertia should be positive.

$$m > 0, I > 0$$

In general, it is possible to provide a set of lower and upper bounds for the dynamic parameters based on a priori knowledge. For instance, the center of mass must be inside the convex link of the robot. However, the aim of our work is to estimate the dynamic coefficients without torque signs. So, we cannot apply this algorithm directly, but we need a preliminary analysis in which we predict what is the most likely torque sign.

2.3 Tree of solutions

First of all, we take the absolute values of the generated torques, so to pretend that we haven't this kind of information.
We can imagine that neighboring torques have the same sign. So, we have that sequent torque values usually have the same sign, then it starts to decrease its absolute value, approaching zero, until it crosses zero and change sign. As a consequence, we have to "mark" the function whenever it crosses 0. In order to do this, we should tolerate a threshold; we compute this threshold by discarding the 10% of samples closer to 0. At this point, we discard all the samples under threshold; we are left with several segments so that in each of them the sign of the torque is either positive or negative.

$$\begin{bmatrix} Y(q(t_1), \dot{q}(t_1), \ddot{q}(t_1)) \\ \vdots \\ Y(q(t_n), \dot{q}(t_n), \ddot{q}(t_n)) \end{bmatrix} \begin{bmatrix} gmd \\ I + md^2 \end{bmatrix} = \begin{bmatrix} \tau(t_1) \\ \vdots \\ \tau(t_n) \end{bmatrix}$$

From this formula, we infer that it is possible to treat all the segments one indipendently from the other, because each torque at a given time instant has its own Y's row. We can rewrite the previous formula so as it fits our problem (in the case of n segments):

$$\begin{bmatrix} Y_{seg_1}(t_{i_1}, ..., t_{f_1}) \\ \vdots \\ Y_{seg_n}(t_{i_n}, ..., t_{f_n}) \end{bmatrix} \begin{bmatrix} gmd \\ I + md^2 \end{bmatrix} = \begin{bmatrix} \tau_{seg_1}(t_{i_1}, ..., t_{f_1}) \\ \vdots \\ \tau_{seg_n}(t_{i_n}, ..., t_{f_n}) \end{bmatrix}$$

While dealing with a single segment, we take the corresponding Y and τ blocks. So, for the k^{th} segment the problem becomes:

$$Y_{seg_k}(t_{i_k}, ..., t_{f_k}) \begin{bmatrix} gmd \\ I + md^2 \end{bmatrix} = \tau_{seg_k}(t_{i_k}, ..., t_{f_k})$$

By taking the segments independently, it may be the case in which Y_{seg_k} has not full rank. So, we need something to deal with this possibility and always have well defined problems. To compute the order in which the segments will be considered, we choose as heuristic function the one that computes the condition number of Y_{seg_k}. In fact, a problem with a high condition number is said to be ill-conditioned; this means that the solution to the problem is difficult to find. We can define a threshold for the condition number; if the condition number of all Y_{seg} is over this threshold, we start considering all the possible pairs; if we are still over-threshold, then we consider the triplets and so on. At this stage, we have a well defined problem from which we can start predicting the torque signs; any problem with fewer segments would not be as good as the one we have.

After that, we have to predict the torque sign for each segment. The previous step returns the order in which segments should be considered so as the condition number decreases. The idea is: we take the first segment which could be either positive or negative, so we have two possible solutions; then, we take the following segment in the sequence which again could be either positive or negative, so at this stage we have four possible solutions (by considering the previous ones), and so on. This is an exponential problem, so if we have n segments we would have 2^n possible solutions. However, it is possible to prevent the tree from being completely expanded because we can recognize in advance when a choice of sign lead to a wrong solution; in fact, when the PBRP algorithm is applied to a torque with the wrong sign, the loss should be higher than the one with the correct sign. So, at each level of the tree, we expand only the best nodes which are the ones with the lower loss until that segments (in our case, we decided to expand the best 5 nodes). For all the optimization problems of the tree we use pattern search algorithm because it is faster.

At the end of this tree algorithm, we will obtain the best combination of signs for the segments sequence. So, we apply the PBRP algorithm to this sequence with a number of runs equal to 3 using simulated annealing.

2.4 Experiments

Experiment 1 For our experiments, we start with a very simple trajectory in which the robot starts from $q = 0$, then it goes to the configuration $q = \frac{\pi}{4}$ and finally it returns to the initial configuration. In order to obtain such trajectory, we interpolate these points with a cubic spline. With this trajectory, the resulting torques are always positive, so it will be easier for the algorithm to compute the sign.

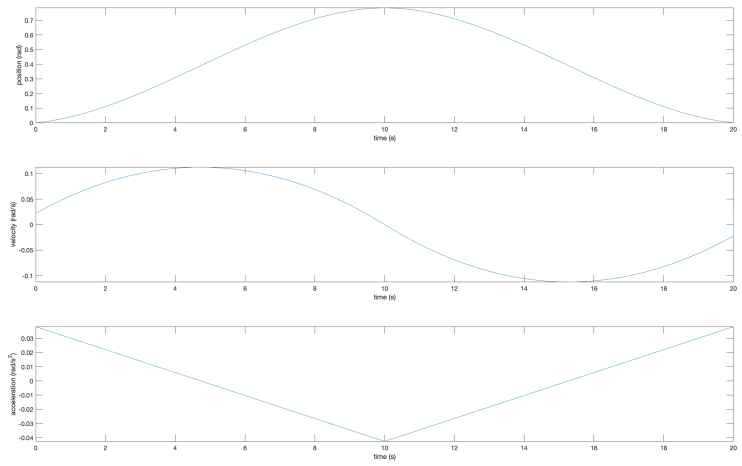

Figure 2: Plot of position, velocity and acceleration; the last two are obtained by symbolic differentiation

From these values, we obtain the torques by computing the Y matrix, the dynamic coefficients π and multiplying them. The dynamic parameters are up to our choice, so we invented their values in order to compute the dynamic coefficients.

At this point, we stack all the Y and τ together. Then, we take the absolute values of τ and send them to the tree algorithm. Finally, we send $\overline{Y}$ and $\overline{\tau}$ with the estimated signs to the PBRP algorithm.

We report the results:

	Ground truth	Retrieved
$m\ [kg]$	5	5.8590
$d\ [m]$	0.5	0.4267
$I\ [kg \cdot m^2]$	0.4208	0.6041

Table 1: Dynamic parameters

We remember that:

$$\boldsymbol{\pi} = \begin{bmatrix} \pi_1 \\ \pi_2 \end{bmatrix} = \begin{bmatrix} gmd \\ I + md^2 \end{bmatrix} = \begin{bmatrix} 24.5250 \\ 1.6709 \end{bmatrix}$$

Even if there is a slightly difference between the ground dynamic parameters and the estimated ones, we obtain the same dynamic coefficients. As a measure of the error, we take the norm of the difference between the ground values of the dynamic coefficients and the estimated ones; in this experiment, we obtain a really small error equal to $1.6410 \cdot 10^{-5}$.

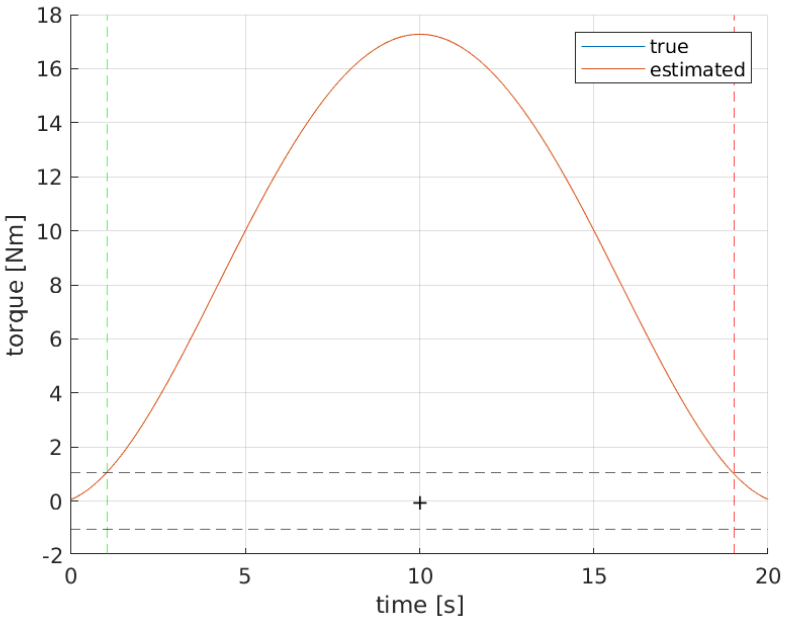

Figure 3: Plot of the reconstructed torque vs nominal torque. The gray dashed lines represent the threshold; the green dashed lines represent the start of the segment, while the red dashed lines represent the end of the segment. Samples under threshold are discarded. The signs estimated by the algorithm are reported for each segment.

Experiment 2 Then, we move to a slightly more difficult trajectory, in which the robot starts from $q = 0$, then it goes to the configuration $q = \frac{\pi}{4}$, then to $q = -\frac{\pi}{4}$ and finally it returns to the initial configuration. Again, we interpolate these points with a cubic spline. With this trajectory, the resulting torques are positive in the first half period and negative in the second one.

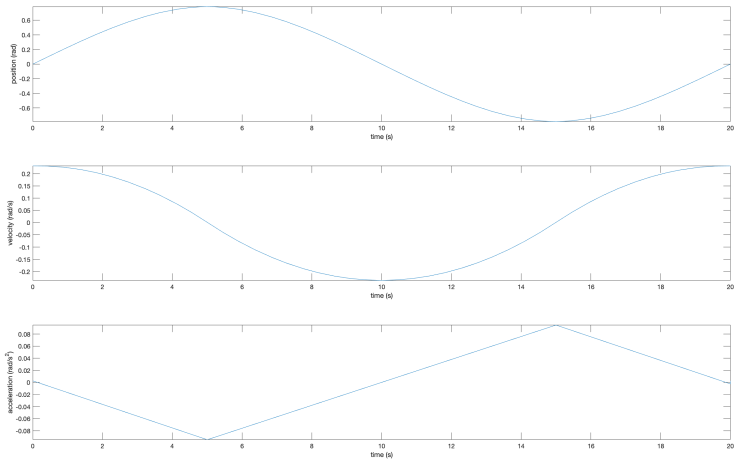

Figure 4: Plot of position, velocity and acceleration; the last two are obtained by symbolic differentiation

We follow the same procedure as before and we obtain:

	Ground truth	Retrieved
$m\ [kg]$	5	6.3695
$d\ [m]$	0.5	0.3926
$I\ [kg \cdot m^2]$	0.4208	0.7316

Table 2: Dynamic parameters

In this case, we don't obtain the exactly same values of the dynamic coefficients; however, the error is really small ($4.2554 \cdot 10^{-2}$).

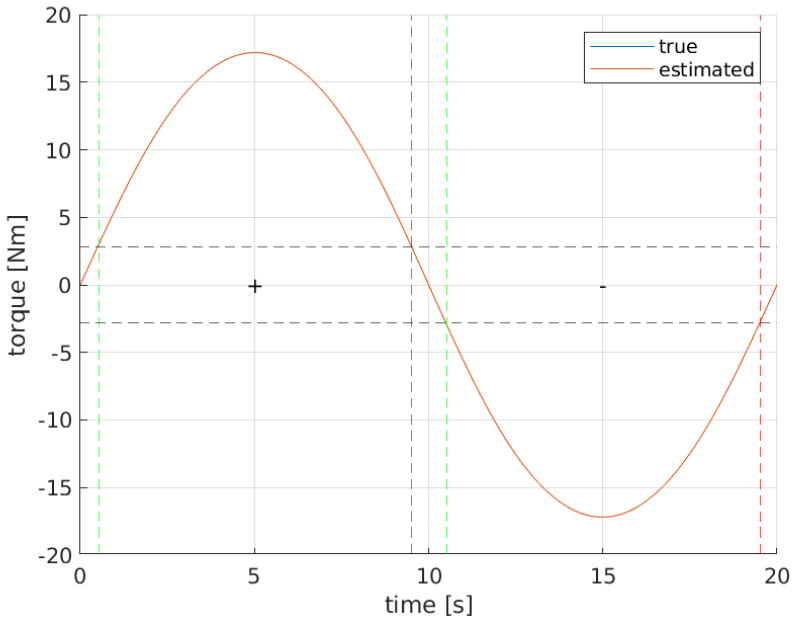

Figure 5: Plot of the reconstructed torque vs nominal torque. The gray dashed lines represent the threshold; the green dashed lines represent the start of the segment, while the red dashed lines represent the end of the segment. Samples under threshold are discarded. The signs estimated by the algorithm are reported for each segment.

Experiment 3 In the last experiment for the 1-dof arm, we moved to a more exciting trajectory generated according to this formula:

$$q_j(t) = \sum_{l=1}^{L} \frac{a_{l,j}}{l\omega_f} \sin(l\omega_f t) - \frac{b_{l,j}}{l\omega_f} \cos(l\omega_f t) + q_{0,j}$$

We implemented a function called generate_exciting_traj() that generates a random exciting trajectory using the following parameters: $L = 5$ and $\omega_f = 0.2\pi$.

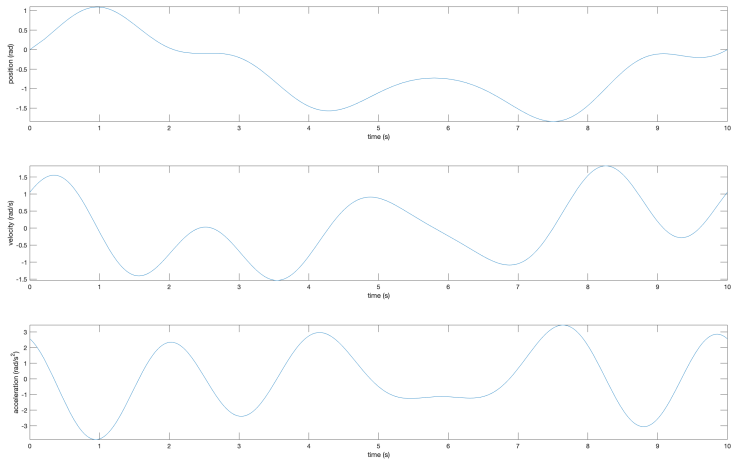

Figure 6: Plot of position, velocity and acceleration; the last two are obtained by symbolic differentiation

We follow the same procedure as before and we obtain:

	Ground truth	Retrieved
$m\ [kg]$	5	7.5379
$d\ [m]$	0.5	0.3317
$I\ [kg \cdot m^2]$	0.4208	0.8417

Table 3: Dynamic parameters

With this experiment, we obtain almost the same dynamics coefficients with an error equal to $2.0259 \cdot 10^{-6}$.

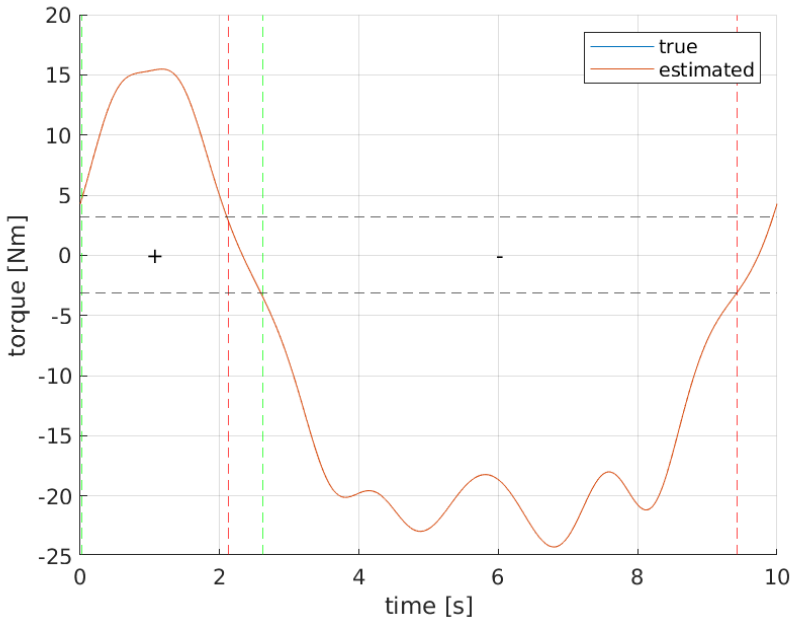

Figure 7: Plot of the reconstructed torque vs nominal torque. The gray dashed lines represent the threshold; the green dashed lines represent the start of the segment, while the red dashed lines represent the end of the segment. Samples under threshold are discarded. The signs estimated by the algorithm are reported for each segment.

2.5 Validation

We validate the results obtained in the third experiment on a new never seen trajectory:

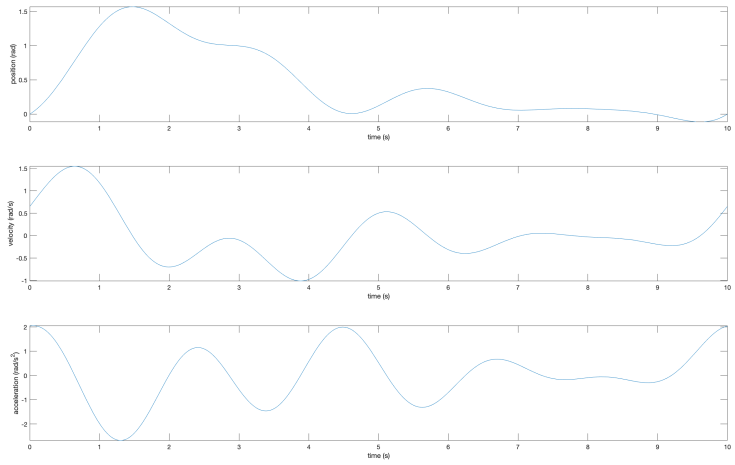

Figure 8: Plot of position, velocity and acceleration; the last two are obtained by symbolic differentiation

The predicted torques are:

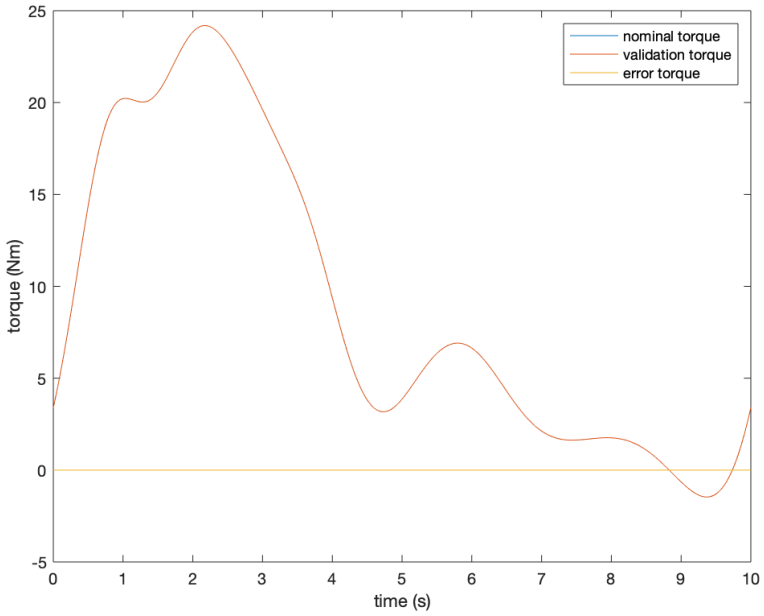

Figure 9: Plot of the nominal torque vs validation torque; the error is computed as nominal torque - validation torque

3 3R spatial robot

After the simple case, we tried to extend this procedure to a more complex robot: a 3R spatial robot.
So, first of all we needed to determine the structure of the robot by defining the Denavit-Hartenberg parameters and to determine the symbolic dynamic model of the robot.

3.1 Robot structure

Denavit-Hartenberg

	α	a	d	θ
link 1	$\pi/2$	0	$L_1 = 0.3$	q_1
link 2	0	$L_2 = 0.3$	$-d_2 = -0.09$	q_2
link 3	0	$L_3 = 0.2$	0	q_3

Table 4: In this table the DH parameters of the robot are reported.

D-H Frames

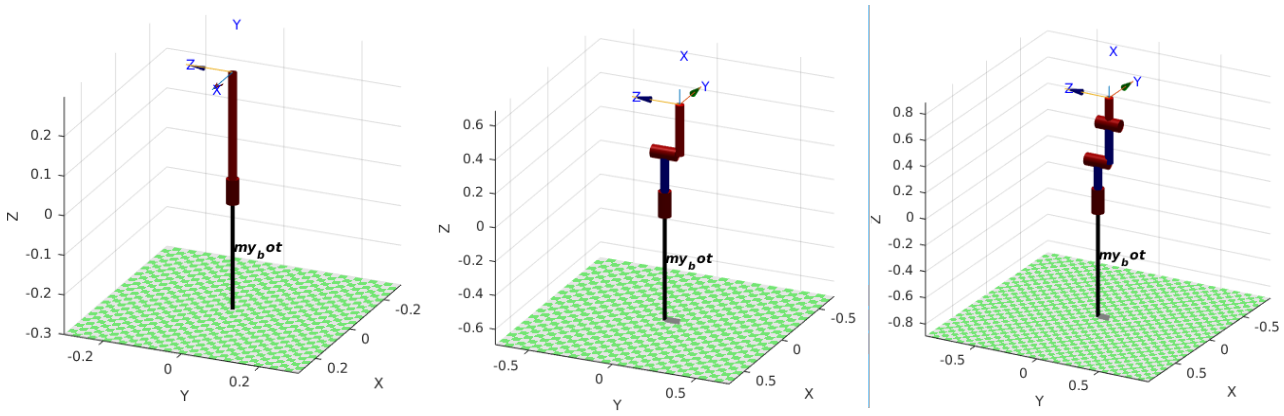

Figure 10: (a) Frame 1 (b) Frame 2 (c) Frame 3

Centers of Mass

We computed the position of the centers of mass in their respective frames, according to the DH frames we just defined.

$$r_{c1} = \begin{pmatrix} 0 \\ -a \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ -0.15 \\ 0 \end{pmatrix}$$

$$r_{c2} = \begin{pmatrix} -b \\ 0 \\ -c \end{pmatrix} = \begin{pmatrix} -0.15 \\ 0 \\ -0.06 \end{pmatrix}$$

$$r_{c3} = \begin{pmatrix} -d \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} -0.10 \\ 0 \\ 0 \end{pmatrix}$$

Dynamic model

To compute the dynamic model of the robot we followed these steps:

- We implemented the moving frame algorithm in order to obtain the linear and angular velocities of the frames' origins;

- We computed the kinetic energy expressing the inertial tensors with respect to the centers of mass;
- From the kinetic energy, we computed the inertia matrix and the Coriolis and centrifugal terms of the model;
- After that we computed the potential energy, using the centers of mass expressed with respect to the base frame and we computed the gravity vector.

Once we obtained the dynamic model of the robot, we found a minimal representation of the model. The dynamic coefficients we found are given by:

$$\pi_1 = I_{1,yy} + m_2(c + d_2)^2 + m_3 d_2^2$$
$$\pi_2 = m_2(L_2 - b)^2 + m_3 L_2^2 + I_{2,yy}$$
$$\pi_3 = I_{2,xx}$$
$$\pi_4 = I_{3,yy} + m_3(L_3 - d)^2$$
$$\pi_5 = I_{3,xx}$$
$$\pi_6 = m_3(L_3 - d)L_2$$
$$\pi_7 = I_{2,zz} + m_2(L_2 - b)^2 + I_{3,zz} + m_3(L_2^2 + (L_3 - d)^2)$$
$$\pi_8 = m_3(L_3 - d)^2 + I_{3,zz}$$
$$\pi_9 = m_2(L_2 - b)(c + d_2) + m_3 L_2 d_2$$
$$\pi_{10} = m_3(L_3 - d)d_2$$
$$\pi_{11} = g_0 m_2(L_2 - b)$$
$$\pi_{12} = g_0 m_3(L_3 - d)$$
$$\pi_{13} = g_0 m_3 L_2$$

3.2 Simulation

We created a new V-REP scene from scratch, and we represented the robot arm we just discussed.

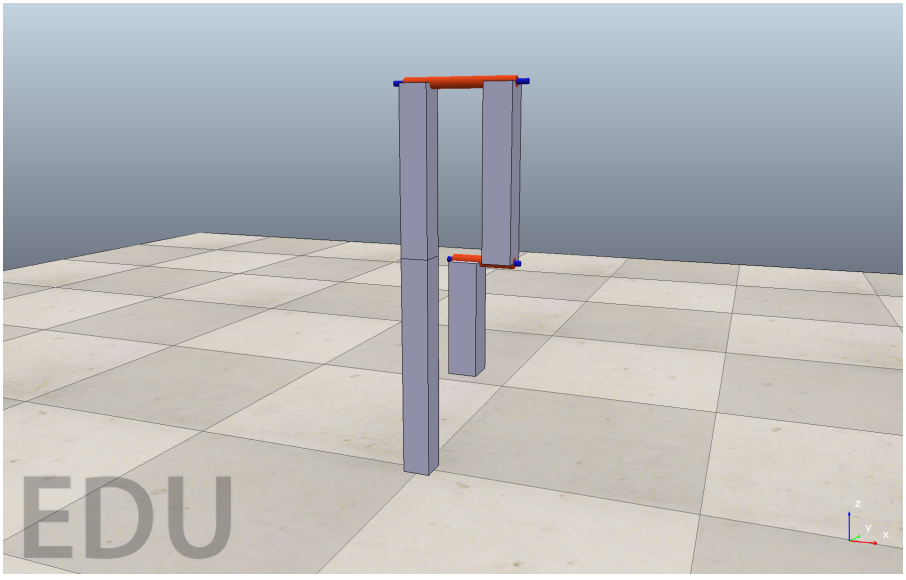

Figure 11: 3R Robot-arm V-REP scene

For the simulation we used the following values for masses and inertia tensors.

$$m_1 = 10\ kg, \quad I_{xx,1} = 7.708 \cdot 10^{-2}\ kg \cdot m^2, \quad I_{yy,1} = 4.167 \cdot 10^{-3}\ kg \cdot m^2, \quad I_{zz,1} = 7.708 \cdot 10^{-2}\ kg \cdot m^2$$
$$m_2 = 1.125\ kg, \quad I_{xx,2} = 4.6879 \cdot 10^{-4}\ kg \cdot m^2, \quad I_{yy,2} = 8.6715 \cdot 10^{-3}\ kg \cdot m^2, \quad I_{zz,2} = 8.6715 \cdot 10^{-3}\ kg \cdot m^2$$
$$m_3 = 0.75\ kg, \quad I_{xx,3} = 3.1252 \cdot 10^{-4}\ kg \cdot m^2, \quad I_{yy,3} = 2.6565 \cdot 10^{-3}\ kg \cdot m^2, \quad I_{zz,3} = 2.6565 \cdot 10^{-3}\ kg \cdot m^2$$

The first and the second links are two parallelepipeds which length is 0.3 m and their transversal section is a square with side equal to 0.05 m, while the third link is 0.2 m long and has a transversal section that is a square with side equal to 0.05 m.

As regards the simulation, we used the same approach of the 1R robot, but of course this time we had to deal with 3 links so we had 3 joint positions and 3 joint torques at each robot update.

We used Bullet2.78 as physics engine as we are interested in torque applied to the joint motor, and using this engine is the only way to get directly this information from the V-REP remote API.

3.3 Experiments

We remember that the starting configuration of the robot is $q_1 = 0$, $q_2 = -\pi/2$ and $q_3 = 0$. So, we have generated some trajectories starting from 0 (that we will use for the first and third link) and others starting from $-\pi/2$ (that we will use for the second link) with the generate_exciting_traj() function. Then, we have done 4 simulations by combining them in different ways.
Again, all these trajectories are repeated 5 consecutive times but we take only the 3 intermediate periods.
One note about the filtering process: we used low-pass filters with different cutoff frequencies in order to accomplish different tasks. In particular we filtered velocities, obtained from positions differentiation, with a cutoff frequency of $0.5 \cdot \pi$ Hz. Then, since the acceleration needed two consecutive numerical differentiations, it results to be very noisy, so we used a filter with a lower cutoff frequency ($0.3 \cdot \pi$ Hz). The same filter is used also to filter the second and third joint torques, while for the first joint torque we used an even more aggressive filter with a cutoff frequency of $0.25 \cdot \pi$ Hz.

Experiment 1 We report the results of the simulation with the first trajectory generated:

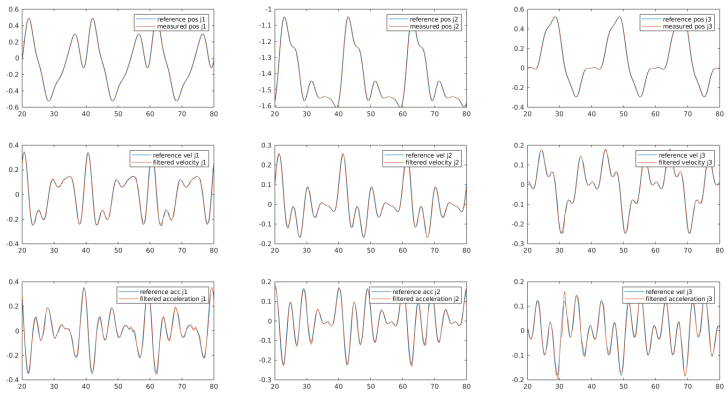

Figure 12: Plot of the reference position, velocity and acceleration vs measured position, velocity and (filtered) acceleration

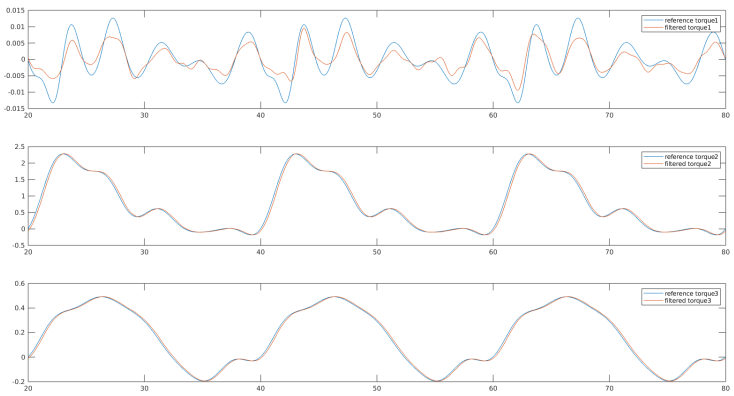

Figure 13: Plot of the reference torque computed using the dynamic model and the reference position, velocity and acceleration vs (filtered) measured torque

Experiment 3 We report the results of the simulation with the second trajectory generated:

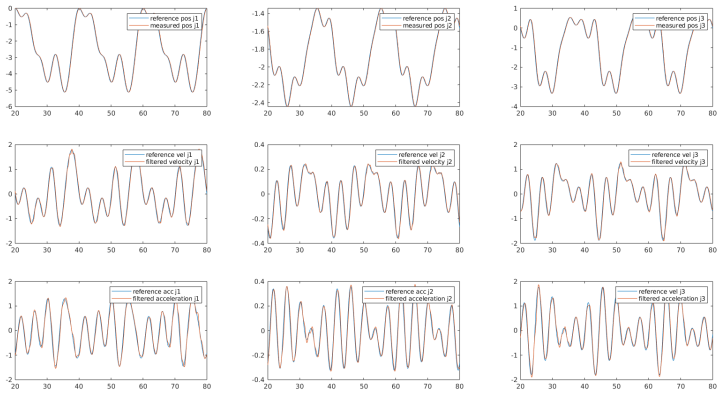

Figure 14: Plot of the reference position, velocity and acceleration vs measured position, velocity and (filtered) acceleration

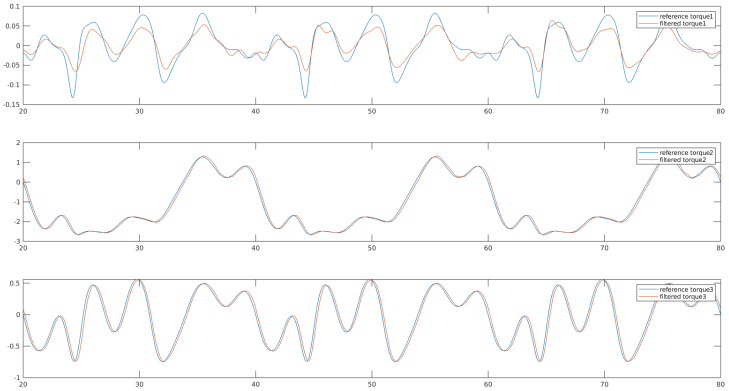

Figure 15: Plot of the reference torque computed using the dynamic model and the reference position, velocity and acceleration vs (filtered) measured torque

Experiment 3 We report the results of the simulation with the third trajectory generated:

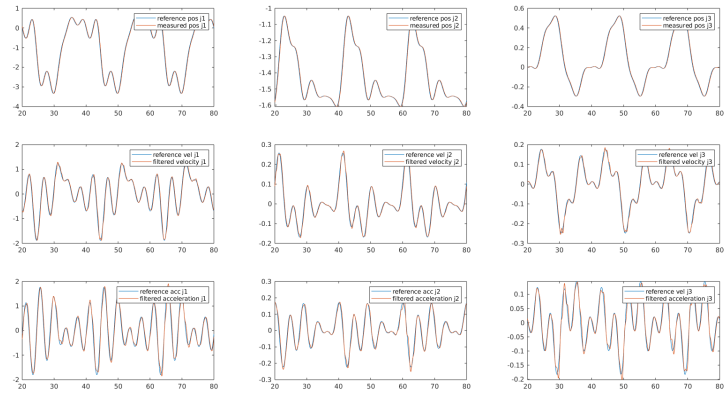

Figure 16: Plot of the reference position, velocity and acceleration vs measured position, velocity and (filtered) acceleration

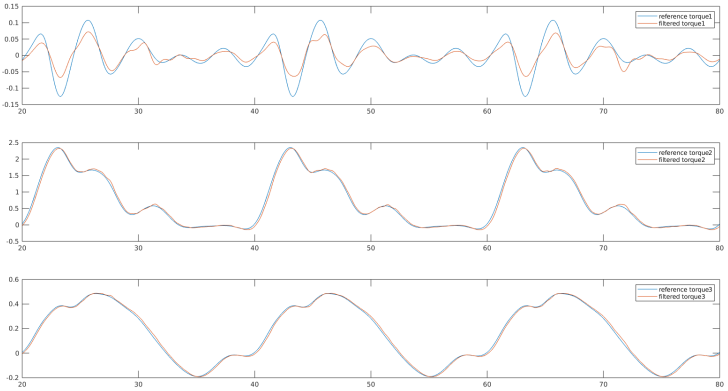

Figure 17: Plot of the reference torque computed using the dynamic model and the reference position, velocity and acceleration vs (filtered) measured torque

Experiment 4 We report the results of the simulation with the fourth trajectory generated:

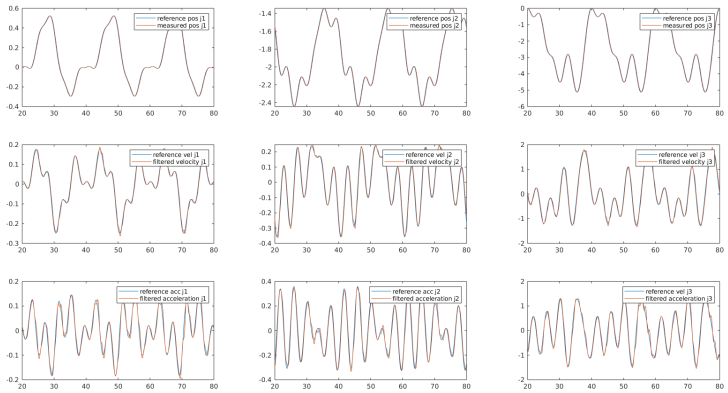

Figure 18: Plot of the reference position, velocity and acceleration vs measured position, velocity and (filtered) acceleration

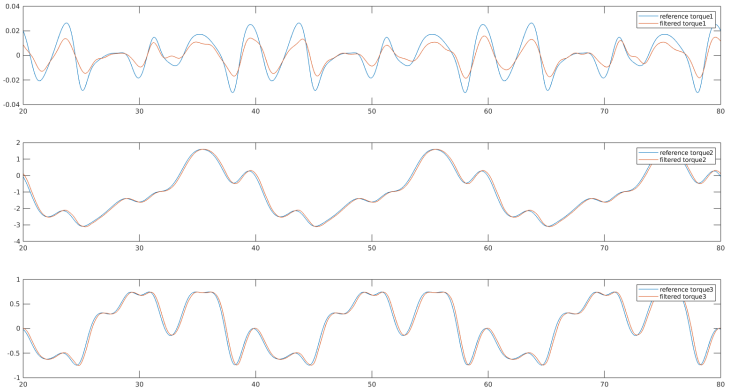

Figure 19: Plot of the reference torque computed using the dynamic model and the reference position, velocity and acceleration vs (filtered) measured torque

3.4 PBRP algorithm

We have extended the procedure for the 1R robot to the 3R case. Since we have more than one joint, we need to add some constraints; this is the complete algorithm:

Algorithm 1: Parameters retrieval

$\boldsymbol{p}_0 \leftarrow LB + (UB - LB)\boldsymbol{u}, \quad \text{with} \quad \boldsymbol{u} \sim \mathcal{U}(0,1)$;
$\xi_1 \leftarrow 0$;
for $k = 1, \ldots, \kappa$ **do**
  // Start the optimization from the previous step solution
  $p_{k,\text{init}} \leftarrow p_{k-1}$;
  // Solve the following optimization problem

$$\begin{cases} \min\limits_{\boldsymbol{p}_k} f(\boldsymbol{p}_k) &= \phi(\boldsymbol{p}_k) + \xi_k \gamma(\boldsymbol{p}_k) \\ &= \|\boldsymbol{\pi}(\boldsymbol{p}_k) - \hat{\boldsymbol{\pi}}\|^2 + \xi_k \sum_\iota g(h_\iota(\boldsymbol{p}_k)) \\ \text{s.t.} \quad & LB \leq \boldsymbol{p}_k \leq UB \end{cases}$$

  $\xi_{k+1} \leftarrow 10k$
end

Figure 20: PBRP algorithm

In particular, in addition to the lower and upper bounds, for each link l_i the following triangle inequalities regarding the inertias must be satisfied:

$$\begin{cases} \overline{I}_{i,x} + \overline{I}_{i,y} > \overline{I}_{i,z} \\ \overline{I}_{i,z} + \overline{I}_{i,y} > \overline{I}_{i,x} \\ \overline{I}_{i,x} + \overline{I}_{i,z} > \overline{I}_{i,y} \end{cases}$$

These inequalities can be rewritten as:

$$\frac{tr(I_{l_i})}{2} - \lambda_{max}(I_{l_i}) > 0$$

Moreover, the total sum of the link masses must be in a given range, that is:

$$m_{rob,min} \leq \sum_i m_i \leq m_{rob,max}$$

The search algorithm is simulated annealing as before and we run the algorithm 3 times.

3.5 Tree of solutions

The tree algorithm is a simple extension of the 1R case; we highlight the fact that segments of different links are put and examined together in random order, so that we do not give priority to any link or segment. The algorithm used to solve the optimization problems of the tree is pattern search like in the previous case.

After the tree has been built, we have the sequence of most likely signs that we apply to the absolute torques. Finally, we run the PBRP algorithm to the torques with the estimated signs for 3 times by using simulated annealing as search algorithm.

3.6 Results

Also in this case we are going to show the results obtained by applying the PBRP algorithm to the torques with the estimated signs.

	Ground truth	Experiment 1	Experiment 2	Experiment 3	Experiment 4
m_1	10.0000	8.7750	12.0069	8.2248	8.2510
m_2	1.1250	2.8457	1.6704	1.5018	3.0000
m_3	0.7500	0.3780	0.7781	0.3902	0.5550
r_{c1x}	0	0	0	0	0
r_{c1y}	-0.1500	-0.1775	-0.2717	-0.0743	-0.1428
r_{c1z}	0	0	0	0	0
r_{c2x}	-0.1500	-0.2019	-0.2042	-0.1174	-0.2241
r_{c2y}	0	0	0	0	0
r_{c2z}	-0.0600	0	0	0	0
r_{c3x}	-0.1000	$-2.6625 \cdot 10^{-6}$	-0.1028	-0.0053	-0.0653
r_{c3y}	0	0	0	0	0
r_{c3z}	0	0	0	0	0
I_{1yy}	0.0042	0	0	0	0
I_{2xx}	$4.6879 \cdot 10^{-4}$	0	0	0	0
I_{2yy}	0.0087	0	0	0	0
I_{2zz}	0.0087	0	0	0	0
I_{3xx}	$3.1253 \cdot 10^{-4}$	$1.7519 \cdot 10^{-4}$	0	$4.8827 \cdot 10^{-8}$	0
I_{3yy}	0.0025	$1.0697 \cdot 10^{-4}$	0	$1.4168 \cdot 10^{-6}$	0
I_{3zz}	0.0025	$2.5175 \cdot 10^{-4}$	0	$1.4949 \cdot 10^{-6}$	0

Table 5: Optimal solutions

Again, we will report the estimated dynamic coefficients compared with their ground values for each experiment. We provide also a measure of the error.

We remember that:

$$\boldsymbol{\pi} = \begin{bmatrix} \pi_1 \\ \pi_2 \\ \pi_3 \\ \pi_4 \\ \pi_5 \\ \pi_6 \\ \pi_7 \\ \pi_8 \\ \pi_9 \\ \pi_{10} \\ \pi_{11} \\ \pi_{12} \\ \pi_{13} \end{bmatrix} = \begin{bmatrix} I_{1yy} + m_2(c+d_2)^2 + m_3 \cdot d_2^2 \\ m_2(L_2-b)^2 + m_3 \cdot L_2^2 + I_{2yy} \\ I_{2xx} \\ I_{3yy} + m_3(L_3-d)^2 \\ I_{3xx} \\ m_3(L_3-d)L_2 \\ I_{2zz} + m_2(L_2-b)^2 + I_{3zz} + m_3(L_2^2 + (L_3-d)^2) \\ m_3(L_3-d)^2 + I_{3zz} \\ m_2(L_2-b)(c+d_2) + m_3 \cdot L_2 \cdot d_2 \\ m_3(L_3-d)d_2 \\ g_0 \cdot m_2(L_2-b) \\ g_0 \cdot m_3(L_3-d) \\ g_0 \cdot m_3 \cdot L_2 \end{bmatrix}$$

The ground values of the dynamic coefficients are:

$$\begin{bmatrix} \pi_1 \\ \pi_2 \\ \pi_3 \\ \pi_4 \\ \pi_5 \\ \pi_6 \\ \pi_7 \\ \pi_8 \\ \pi_9 \\ \pi_{10} \\ \pi_{11} \\ \pi_{12} \\ \pi_{13} \end{bmatrix} = \begin{bmatrix} 0.0113 \\ 0.3040 \\ 0.0005 \\ 0.0702 \\ 0.0003 \\ 0.0675 \\ 0.3741 \\ 0.0702 \\ 0.0354 \\ 0.203 \\ 4.9663 \\ 2.2073 \\ 2.2072 \end{bmatrix}$$

Experiment 1 The estimated dynamic coefficients without torque signs are:

$$\begin{bmatrix} \pi_1 \\ \pi_2 \\ \pi_3 \\ \pi_4 \\ \pi_5 \\ \pi_6 \\ \pi_7 \\ \pi_8 \\ \pi_9 \\ \pi_{10} \\ \pi_{11} \\ \pi_{12} \\ \pi_{13} \end{bmatrix} = \begin{bmatrix} 0.0261 \\ 0.0614 \\ 0 \\ 0.0152 \\ 0.0002 \\ 0.0227 \\ 0.0768 \\ 0.0154 \\ 0.0353 \\ 0.0068 \\ 2.7377 \\ 0.7416 \\ 1.1124 \end{bmatrix}$$

The norm of the error is 2.9102.

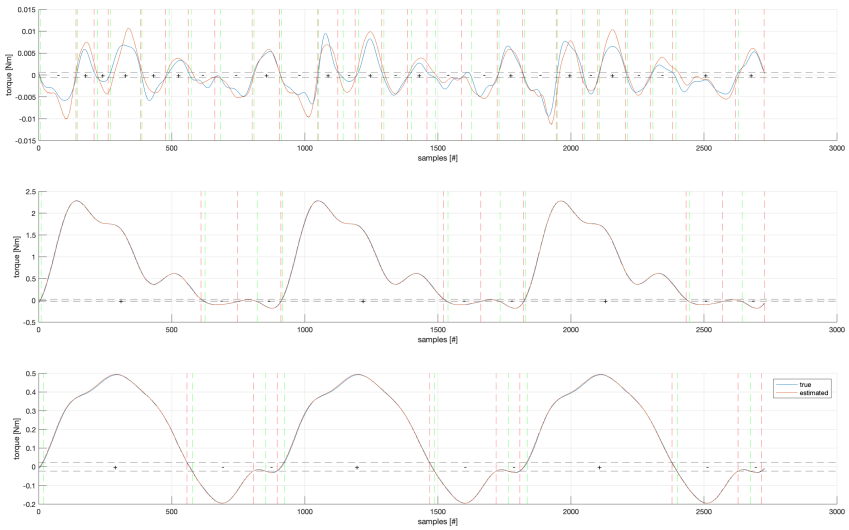

Figure 21: Plot of the reconstructed torque vs nominal torque. The gray dashed lines represent the threshold; the green dashed lines represent the start of the segment, while the red dashed lines represent the end of the segment. Samples under threshold are discarded. The signs estimated by the algorithm are reported for each segment.

Experiment 2 The estimated dynamic coefficients without torque signs are:

$$\begin{bmatrix} \pi_1 \\ \pi_2 \\ \pi_3 \\ \pi_4 \\ \pi_5 \\ \pi_6 \\ \pi_7 \\ \pi_8 \\ \pi_9 \\ \pi_{10} \\ \pi_{11} \\ \pi_{12} \\ \pi_{13} \end{bmatrix} = \begin{bmatrix} 0.0198 \\ 0.0854 \\ 0 \\ 0.074 \\ 0 \\ 0.0227 \\ 0.0927 \\ 0.0074 \\ 0.0354 \\ 0.0068 \\ 1.5701 \\ 0.7423 \\ 2.29 \end{bmatrix}$$

The norm of the error is 3.7181.

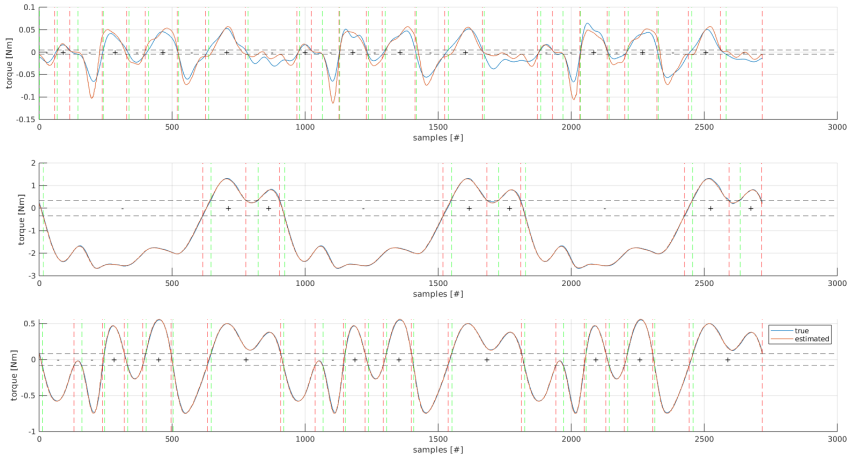

Figure 22: Plot of the reconstructed torque vs nominal torque. The gray dashed lines represent the threshold; the green dashed lines represent the start of the segment, while the red dashed lines represent the end of the segment. Samples under threshold are discarded. The signs estimated by the algorithm are reported for each segment.

Experiment 3 The estimated dynamic coefficients without torque signs are:

$$\begin{bmatrix} \pi_1 \\ \pi_2 \\ \pi_3 \\ \pi_4 \\ \pi_5 \\ \pi_6 \\ \pi_7 \\ \pi_8 \\ \pi_9 \\ \pi_{10} \\ \pi_{11} \\ \pi_{12} \\ \pi_{13} \end{bmatrix} = \begin{bmatrix} 0.0153 \\ 0.0852 \\ 0 \\ 0.0148 \\ 0 \\ 0.0228 \\ 0.1 \\ 0.0148 \\ 0.0352 \\ 0.0068 \\ 2.6904 \\ 0.7454 \\ 1.1483 \end{bmatrix}$$

The norm of the error is 2.9274.

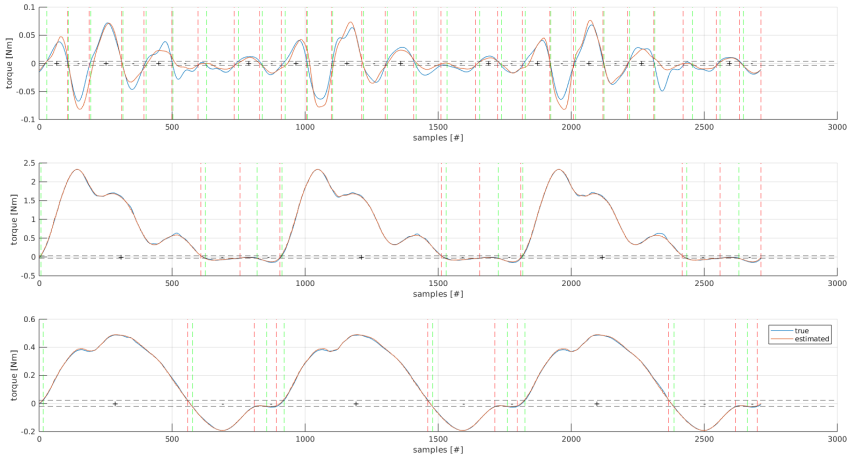

Figure 23: Plot of the reconstructed torque vs nominal torque. The gray dashed lines represent the threshold; the green dashed lines represent the start of the segment, while the red dashed lines represent the end of the segment. Samples under threshold are discarded. The signs estimated by the algorithm are reported for each segment.

Experiment 4 The estimated dynamic coefficients without torque signs are:

$$\begin{bmatrix} \pi_1 \\ \pi_2 \\ \pi_3 \\ \pi_4 \\ \pi_5 \\ \pi_6 \\ \pi_7 \\ \pi_8 \\ \pi_9 \\ \pi_{10} \\ \pi_{11} \\ \pi_{12} \\ \pi_{13} \end{bmatrix} = \begin{bmatrix} 0.0288 \\ 0.0672 \\ 0 \\ 0.0101 \\ 0 \\ 0.0224 \\ 0.0773 \\ 0.0101 \\ 0.0355 \\ 0.0067 \\ 2.2342 \\ 0.7334 \\ 1.6332 \end{bmatrix}$$

The norm of the error is 3.1812.

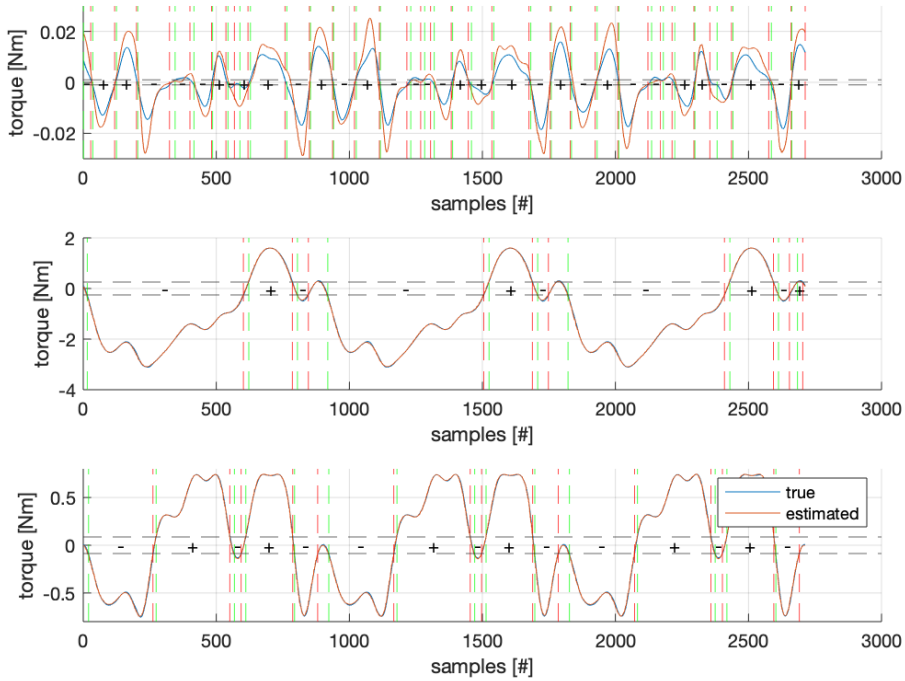

Figure 24: Plot of the reconstructed torque vs nominal torque. The gray dashed lines represent the threshold; the green dashed lines represent the start of the segment, while the red dashed lines represent the end of the segment. Samples under threshold are discarded. The signs estimated by the algorithm are reported for each segment.

4 Simulation

We have implemented a scene in V-REP from scratch, by taking a fixed base, a revolute joint and a link. We control the joint in position with the PID controller built in the joint. The arm is in 0 position at rest. The link is a parallelepiped which length is 1 m and its transversal section is a square with side equal to 0.1 m. The mass of the link is 5 kg and the principal moment of inertia is 0.4208 $kg \cdot m^2$. Since the arm has a uniform density, the center of mass is located at the center of the link, so it is 0.5 m from the top. The physics engine used is Bullet 2.78, because when we will use the API function *simxGetJointForce*, this will return the force or torque applied to the joint motor.

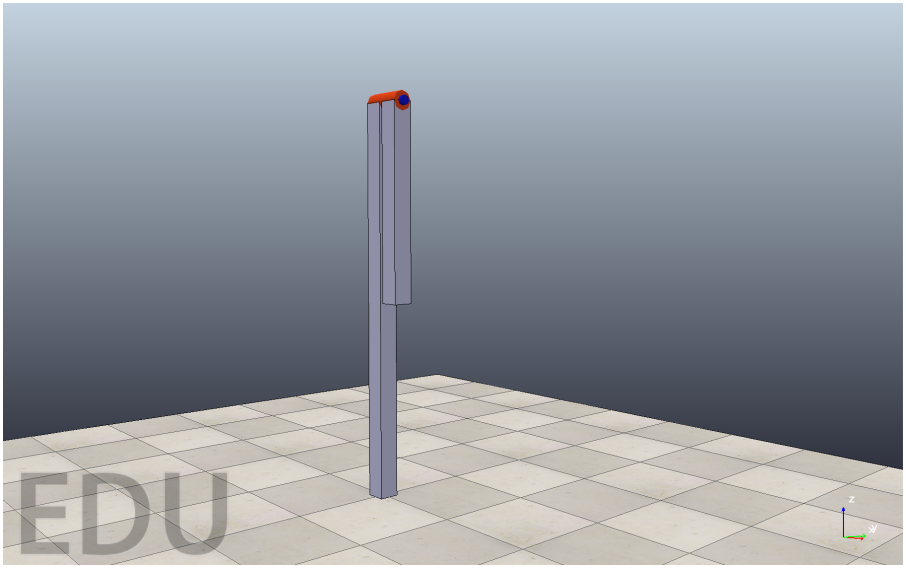

Figure 25: V-REP scene of the 1-dof robot arm

Regarding the simulation, we have written a Matlab script in which we first connect to V-REP and then load the scene. After generated a trajectory, the simulation starts. In the main control loop, we first get the joint position, velocity and torque (with –); then, we set a new target position according to the trajectory and the current instant of time.

We obtain the measured accelerations through numerical differentiation; this leads to somewhat noisy values. In order to compensate for this noise, we apply a low-pass filter of order 16 and cutoff frequency equal to 0.35π Hz. Also the measured torques are noisy, so we apply the same filter.

5 Conclusions

.....

References

[1] C. Gaz, M. Cognetti, A. Oliva, P. R. Giordano, A. De Luca *Dynamic Identification of the Franka Emika Panda Robot With Retrieval of Feasible Parameters Using Penalty-Based Optimization*

[2] C. Gaz, M. Cognetti, A. Oliva, P. R. Giordano, A. De Luca *Dynamic Identification of the Franka Emika Panda Robot With Retrieval of Feasible Parameters Using Penalty-Based Optimization – Supplementary material (revised version: October* 15^{th}*, 2019)*

[3] A. De Luca *Slides of the Robotics I/II courses*