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 achiving many tasks.

In particular we have dealed 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 developed an algorithm able to estimate torque signs so that we could procede 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.

2 1R arm under gravity

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

2.1 Simulation and dynamic model

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 $5 \cdot 1.667 \cdot 10^{-3}$ $kg \cdot m^2$. 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.

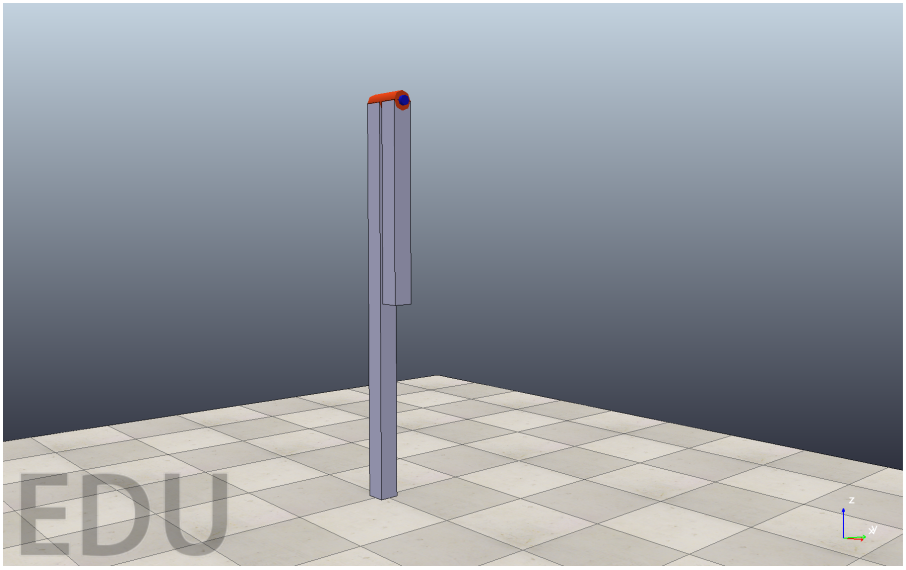

Figure 1: V-REP scene of the 1-dof robot arm

The dynamic model of this kind of robot is:

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

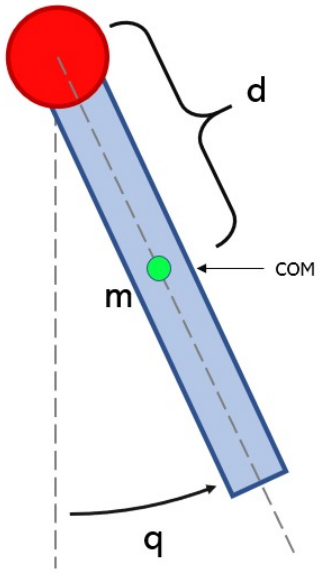

Figure 2: Robot arm model: m is the mass of the link, d is the position of the center of mass, I is the moment of inertia when rotating around the center of mass

Regarding the simulation, we have written a Matlab script in which we first connect to V-REP and then load the scene.
We generate exciting trajectories according to the following 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 $\omega_f = 0.1\pi$.
However, we have started with simpler trajectories generated through the function looping_splines(),which makes an interpolation of some given positions and instants of time with a cubic splines.

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 $0.35 * \pi$. Also the measured torques are noisy, so we apply the same filter.

After that, we have all the ingredients to generate the regressor matrix with all the computed values by staking all the Y in a unique matrix. We do the same for the measured torques obtaining a vector $\overline{\tau}$.

2.2 Experiments

Experiment 1 We have started from a very simple trajectory according to which the robot has to move from the 0 position to $\frac{\pi}{4}$ and then go back to the 0 position in 20 seconds. We interpolated this trajectory using cubic splines.
With this trajectory we obtain a torque which is always positive:

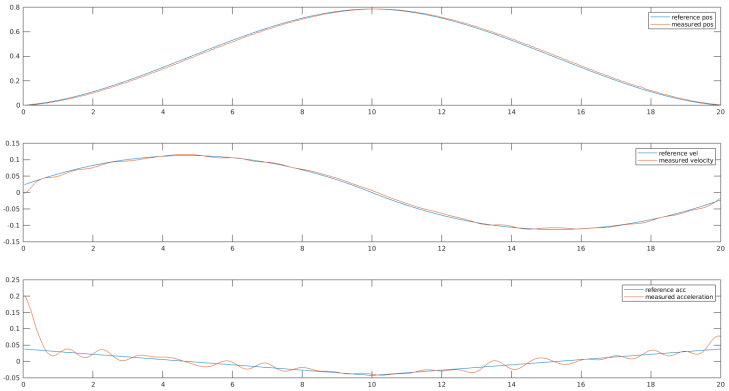

Figure 3: Plot of the reference position, velocity and acceleration vs measured position, velocity and (filtered) acceleration

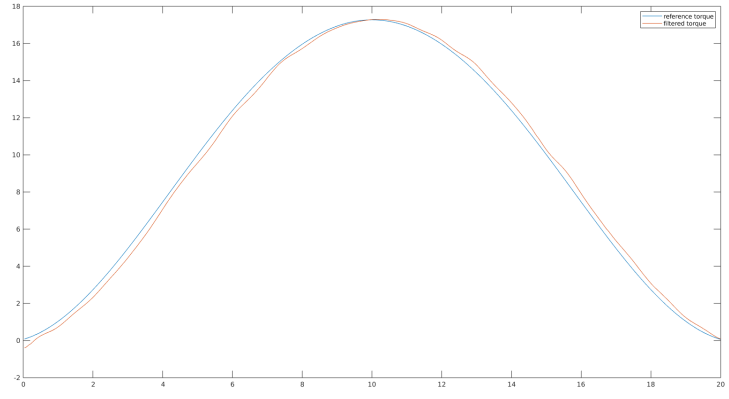

Figure 4: Plot of the reference torque computed using the dynamic model and the reference position, velocity and acceleration vs (filtered) measured torque

Experiment 2 Then, we moved to a more complex trajectory according to which the robot has to move from the 0 position to $\frac{\pi}{4}$, then go to $-\frac{\pi}{4}$ and finally return in the 0 position in 20 seconds. As before, we interpolated this trajectory using cubic splines.

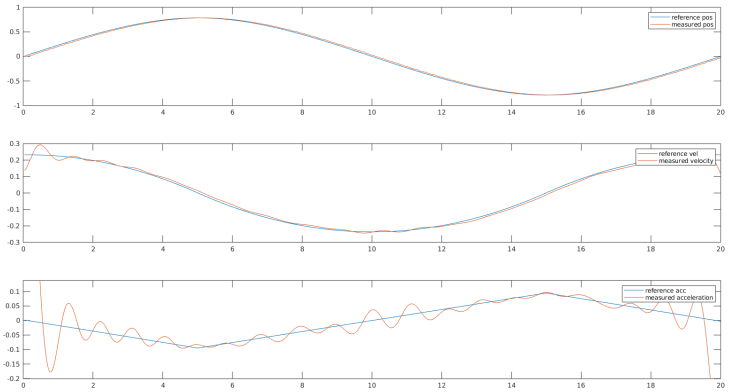

Figure 5: Plot of the reference position, velocity and acceleration vs measured position, velocity and (filtered) acceleration

With this trajectory we obtain a torque which is positive in the first half wave and negative in the second half wave:

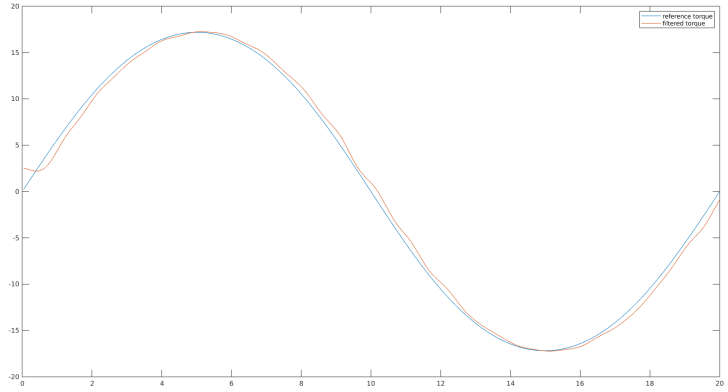

Figure 6: Plot of the reference torque computed using the dynamic model and the reference position, velocity and acceleration vs (filtered) measured torque

Experiment 3 and 4 Finally, we have generated two trajectories made by the combination of sines and cosines through the generate_exciting_traj() function. Both these trajectories are repeated 3 consecutive times. The results are reported below.

Trajectory one:

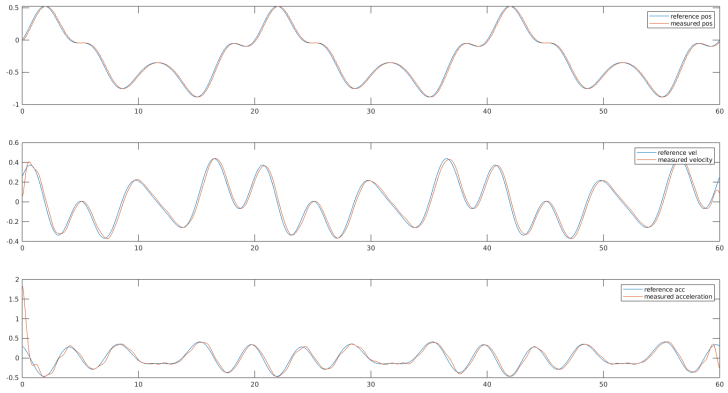

Figure 7: Plot of the reference position, velocity and acceleration vs measured position, velocity and (filtered) acceleration

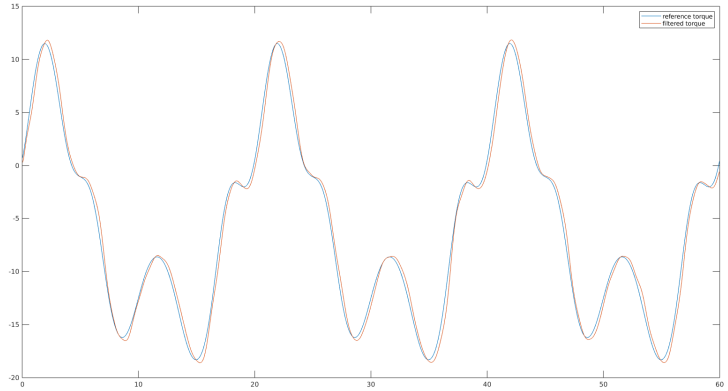

Figure 8: Plot of the reference torque computed using the dynamic model and the reference position, velocity and acceleration vs (filtered) measured torque

Trajectory two:

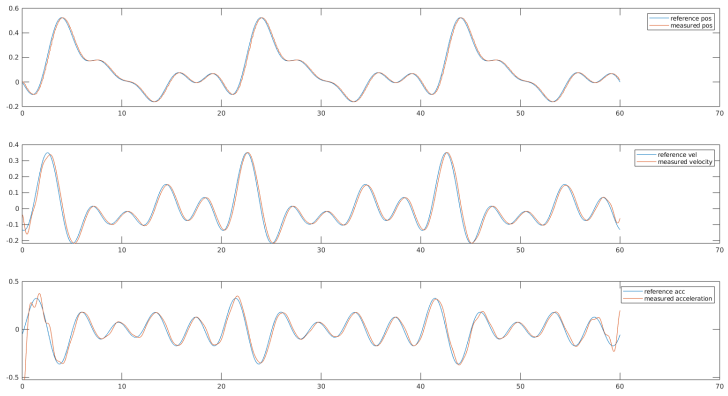

Figure 9: Plot of the reference position, velocity and acceleration vs measured position, velocity and (filtered) acceleration

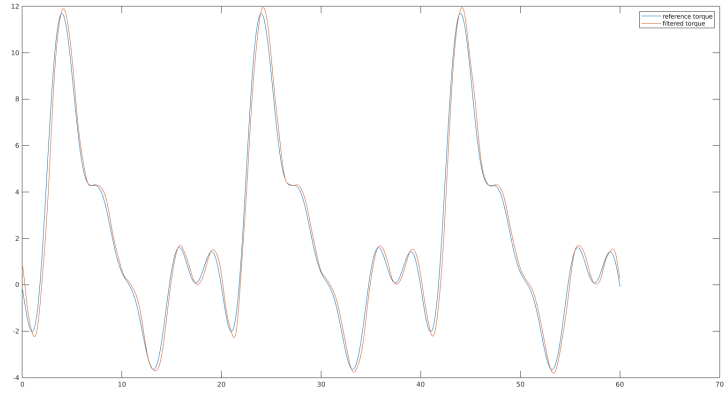

Figure 10: Plot of the reference torque computed using the dynamic model and the reference position, velocity and acceleration vs (filtered) measured torque

2.3 PBRP algorithm

If we had torque signs, we would apply directly the Penalty-Based Parameters Retrivial (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.

2.4 Tree of solutions

First of all, we take the absolute values of the filtered measured torques. Then, 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.

After that, we have to predict the torque sign for each segment. The idea is: we first take a random segment which could be either positive or negative, so we have two possible solutions; then, we take another segment 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.

2.5 Results

In this section we are going to show the results obtained by applying the PBRP algorithm to the torques with the estimated signs.

	m	d	I
Ground truth	5	0.5	0.4208
Experiment 1	9.7053	0.2577	0.4116
Experiment 2	9.9157	0.2519	0
Experiment 3	6.9846	0.3577	0.0825
Experiment 4	6.3661	0.3915	0

Table 1: Optimal solutions

As we can see from the table, the optimal solutions that we get are quite different from the nominal values of the dynamic parameters. However, we are interested in the dynamic coefficients, so for each experiment, we will report the estimated dynamic coefficients compared with their ground values. In order to get a measure of the error, we decide to take the norm of the difference between these two vectors.

We remember that:

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

The ground values of the dynamic coefficients are:

$$\begin{bmatrix} \pi_1 \\ \pi_2 \end{bmatrix} = \begin{bmatrix} 24.5250 \\ 1.6708 \end{bmatrix}$$

Experiment 1 The estimated dynamic coefficients without torque signs are:

$$\begin{bmatrix} \pi_1 \\ \pi_2 \end{bmatrix} = \begin{bmatrix} 24.5312 \\ 1.0559 \end{bmatrix}$$

The norm of the error is 0.6149.

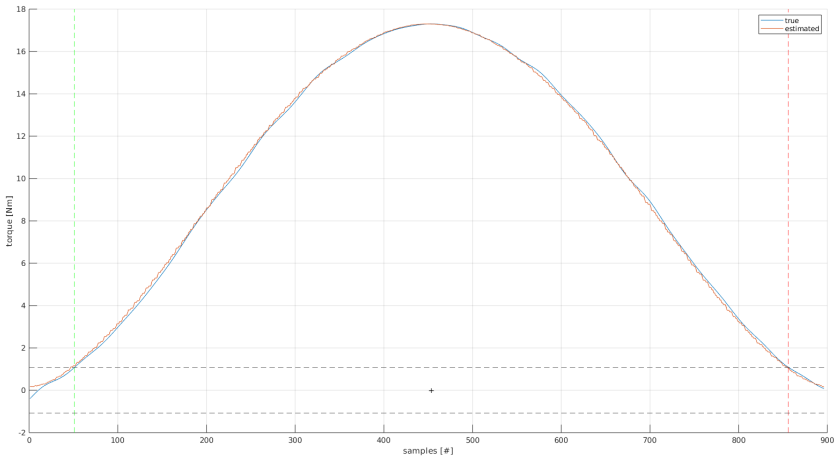

Figure 11: Plot of the reconstructed torque vs nominal torque. The gray dashed lines represent the threshold; the green dashed lines represent the start of the segment, while the red dashed lines represent the end of the segment. Samples under threshold are discarded. The signs estimated by the algorithm are reported for each segment.

Experiment 2 The estimated dynamic coefficients without torque signs are:

$$\begin{bmatrix} \pi_1 \\ \pi_2 \end{bmatrix} = \begin{bmatrix} 24.4990 \\ 0.6290 \end{bmatrix}$$

The norm of the error is 1.0422.

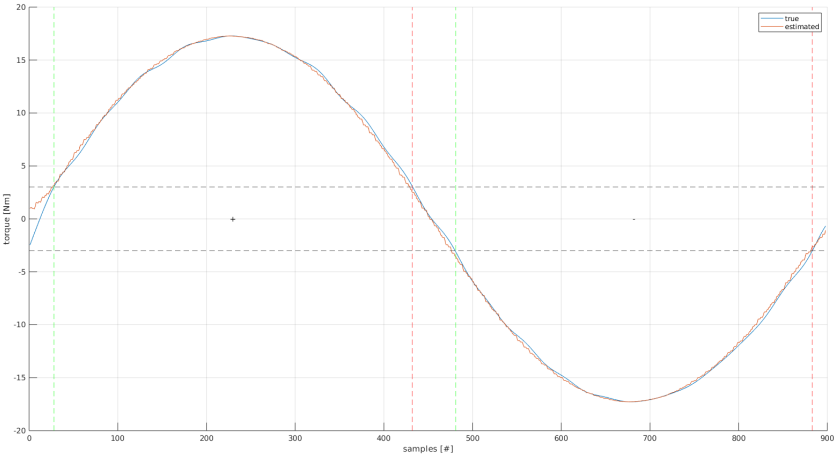

Figure 12: Plot of the reconstructed torque vs nominal torque. The gray dashed lines represent the threshold; the green dashed lines represent the start of the segment, while the red dashed lines represent the end of the segment. Samples under threshold are discarded. The signs estimated by the algorithm are reported for each segment.

Experiment 3 The estimated dynamic coefficients without torque signs are:

$$\begin{bmatrix} \pi_1 \\ \pi_2 \end{bmatrix} = \begin{bmatrix} 24.5069 \\ 0.9760 \end{bmatrix}$$

The norm of the error is 0.6950.

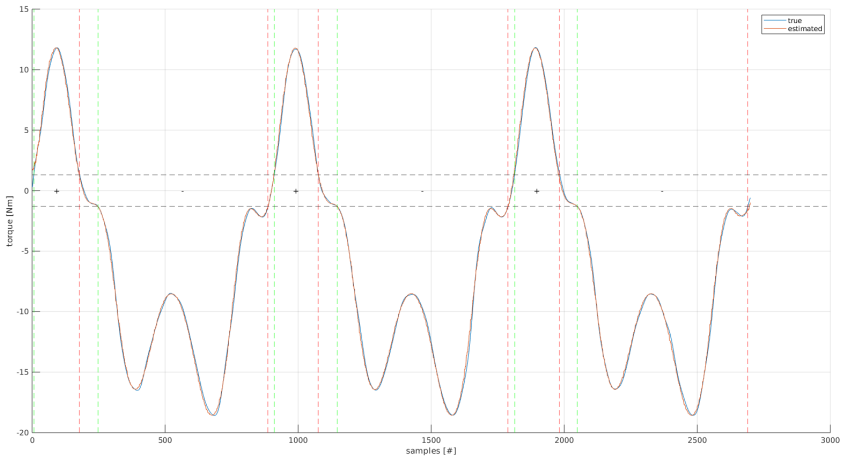

Figure 13: Plot of the reconstructed torque vs nominal torque. The gray dashed lines represent the threshold; the green dashed lines represent the start of the segment, while the red dashed lines represent the end of the segment. Samples under threshold are discarded. The signs estimated by the algorithm are reported for each segment.

Experiment 4 The estimated dynamic coefficients without torque signs are:

$$\begin{bmatrix} \pi_1 \\ \pi_2 \end{bmatrix} = \begin{bmatrix} 24.4473 \\ 0.9756 \end{bmatrix}$$

The norm of the error is 0.6996.

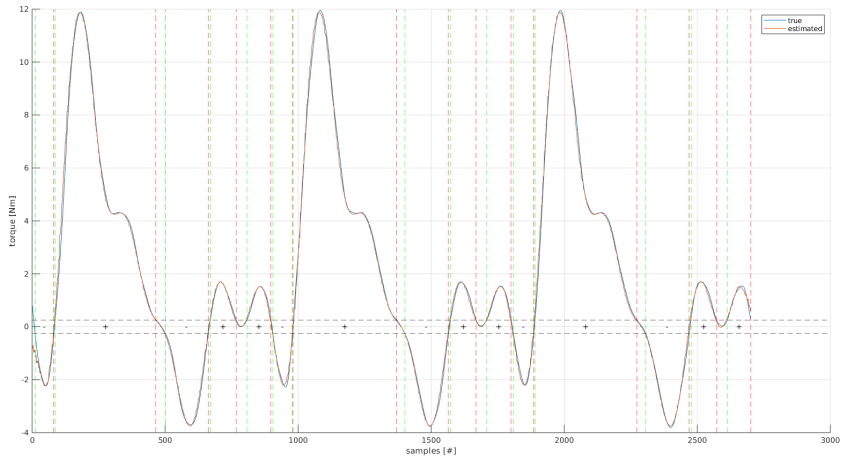

Figure 14: Plot of the reconstructed torque vs nominal torque. The gray dashed lines represent the threshold; the green dashed lines represent the start of the segment, while the red dashed lines represent the end of the segment. Samples under threshold are discarded. The signs estimated by the algorithm are reported for each segment.

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

D-H Frames

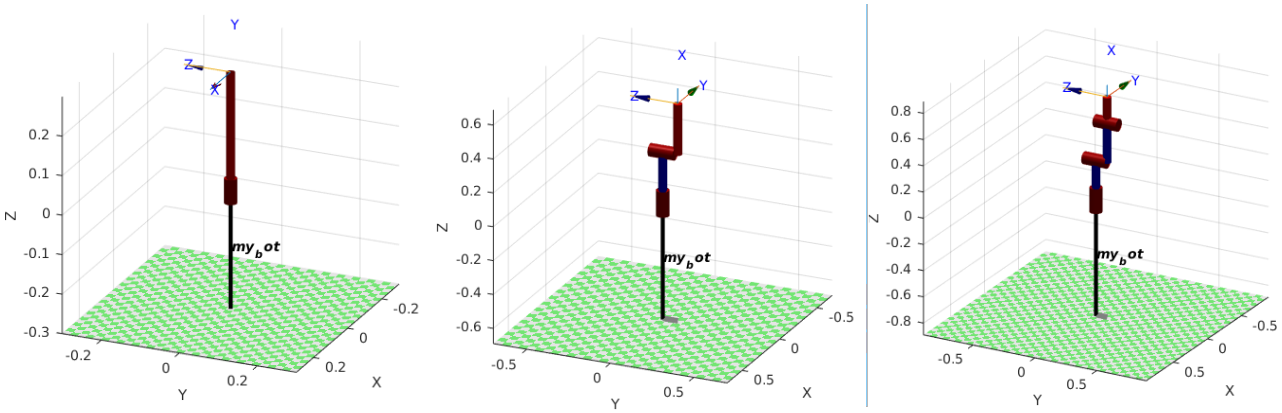

Figure 15: (a) Frame 1 (b) Frame 2 (c) Frame 3

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 origin;
- 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.

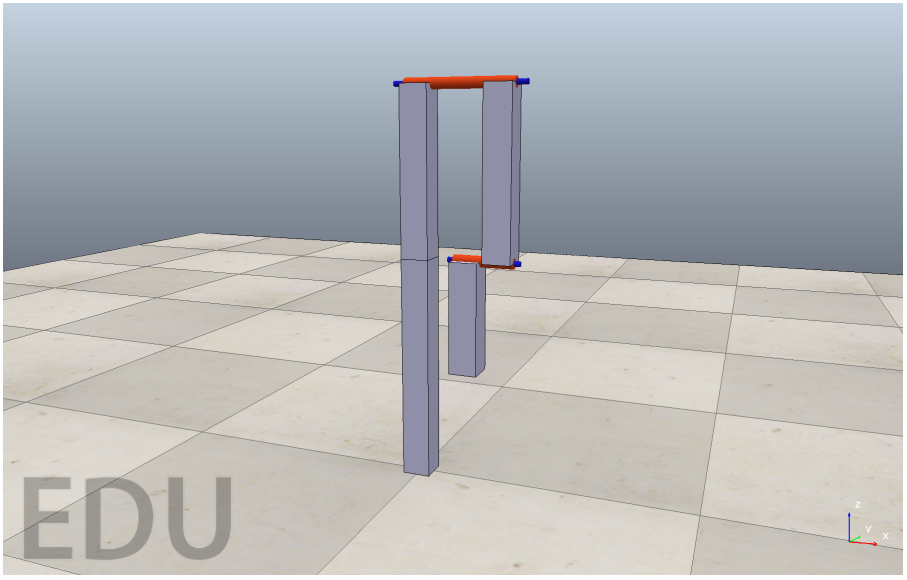

Figure 16: 3R Robot-arm V-REP scene

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

$$m_1 = 10\ kg, \quad I_{xx,1} = 7.708 \cdot 10^{-2}\ kg \cdot m^2, \quad I_{yy,1} = 4.167 \cdot 10^{-3}\ kg \cdot m^2, \quad I_{zz,1} = 7.708 \cdot 10^{-2}\ kg \cdot m^2$$
$$m_2 = 1.125\ kg, \quad I_{xx,2} = 4.167{\cdot}1.125{\cdot}10^{-4}\ kg{\cdot}m^2, \quad I_{yy,2} = 7.708{\cdot}1.125{\cdot}10^{-3}\ kg{\cdot}m^2, \quad I_{zz,2} = 7.708{\cdot}1.125{\cdot}10^{-3}\ kg{\cdot}m^2$$
$$m_3 = 0.75\ kg, \quad I_{xx,3} = 4.167{\cdot}0.75{\cdot}10^{-4}\ kg{\cdot}m^2, \quad I_{yy,3} = 3.542{\cdot}0.75{\cdot}10^{-3}\ kg{\cdot}m^2, \quad I_{zz,3} = 3.542{\cdot}0.75{\cdot}10^{-3}\ kg{\cdot}m^2$$

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 reguards 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.

(vedi velocità)

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

3.4 PBRP algorithm

aggiunti vincoli (es. traccia, ...)

3.5 Tree of solutions

vedi numeri iterazioni

3.6 Results

4 Conclusions