Robot Dynamics and Control Assignment 2.

📍 Luca Predieri, mat. 4667708

The second assignment of Robot Dynamics and Control class in Robotics Engineering Course, regards the study of four exercises. The assignment regards the development of the Inverse Newton-Euler Algorithm in Matlab and its application in three different configurations.

1. Mathematical Approach.

The algorithm is said inverse because it is given desired velocities and accelerations of the joints and I had to find the τ acting on each joint (a force in prismatic and a torque in revolutes).

In the Inverse Newton-Euler Algorithm we have two main sections:

1. Forward Recursion
2. Backward Recursion

1.1 Forward Recursion.

Regarding the first part of the algorithm we calculate (given a desired velocity and acceleration) the distances between the frames, the velocities (angular and linear) and the

accelerations (angular and linear) for each joint. I decided to compute all these values with respect to the base frame, in order to understand better the iteration of the algorithm.
The forward recursion ends with the computation of the linear acceleration of each joint.
It is said to be “forward” because the algorithm starts to compute the values from the first joint, ending with the last one.

1.2 Backward Recursion

The backward Recursion involves the computation of the dynamic forces, and the computation of the momentums and forces acting on a joint because of the previous one (that is why it’s backward). This means that I have to compute the values starting from the most far joint!

1.3 Conclusion of the algorithm.

Later, by projecting the component of the force/momentum on the k_i axis I obtain the useful data. What I obtained is the force/momentum I have to actuate on a joint in order to get an instantaneous velocity and acceleration on the joint that I gave as Input.

---

2. Matlab structure of the code.

I developed a function called `NewtEuler.m` The function contains all the steps described in the previous section. As input the function receives:

- `robot` a struct which contains all the geometrical characteristics and the current configuration of the robot, such as q, $\dot{q}$, $\ddot{q}$.
- `F_ext` the external forces acting on the robot.
- `M_ext` the external momentums acting on the robot.
- `gravity` which is the value of the desired gravity (as the exercises asked with both no gravity and with gravity.

The struct `robot` has many fields:

- `robot.ROT` rotation matrix of the frames of the joints, prof. Cannata calls it R_i.
- `robot.R` structure containing a rigidbodytree object, it contains many fields, such as mass, type of joint, number of bodies.
- `robot.C` structure containing the configuration in which is calculated the algorithm, contains position, acceleration and velocity of the joints.
- `robot.r` contains length of the links when there is no rotation/translation applied with respect to i-th frame.
- `robot.Inertia` contains Inertia matrices of the links, I considered it with respect to the frame of the joint.

- `r_com` distance between the i-th frame and the Com of the i-th link, computed easily by dividing the length of `robot.r`.

---

3. Answers to the exercises.

Now I'm going to explain all the computational reason behind each point, explaining the contributes and showing the results I obtained from the *matlab* script. The matlab script is highly commented so it's easier to understand, most of the computation is repetitive so I decided to automatize it a little bit.

3.1 Exercise 1. Developing the algorithm

As far as the first exercises asks to create the algorithm itself, I decided to highly comment the code. The matlab file is `NewtEuler.m`. The steps I created in the *matlab* function tried to be closest as possible to the pseudocode prof. Cannata gave us. I think it can be improved but it is a good base to start.

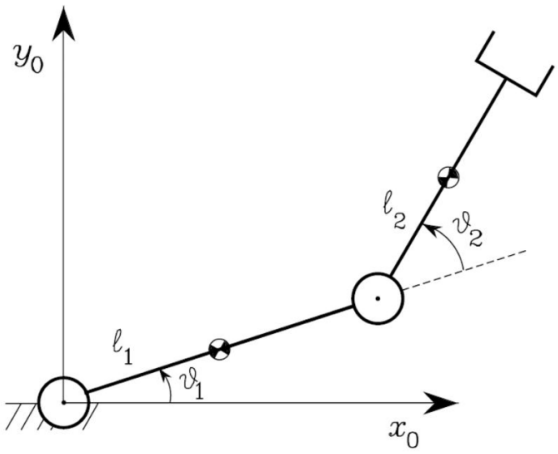
3.1 Exercise 2. First Robot, RR chain

The first robot is a manipulator with two revolute joints with same axis. The axis given were easy because as we can see the axis z is already parallel to the axis of rotation of the joints. The values that I will pass to the struct will depend on the configuration that I will have to manage. The part of the struct in matlab that will change is the `robot.C`.

This simplified a lot of work and helped me understanding the difference of the variables I'm using.

The robot characteristics:

- $m_1 = 22\ Kg$
- $m_2 = 19\ Kg$
- $l_1 = 1\ m$
- $l_2 = 0.8\ m$
- $I_{zz_1} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0.4 \end{bmatrix}$
- $I_{zz_2} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0.3 \end{bmatrix}$



3.1.1 Exercise 2 instance 1.

These are the values for the first configuration:

- $\theta_1 = \pi/9$
- $\theta_2 = 2 \cdot \pi/9$
- $\dot{\theta_1} = 0.2\ rad/s$
- $\dot{\theta_2} = 0.15\ rad/s$
- $\ddot{\theta_1} = 0.1\ rad/s^2$
- $\ddot{\theta_2} = 0.085\ rad/s^2$

The result of the τ is the following (first with gravity, second without):

$$\tau_g = \begin{bmatrix} 318.1937 \\ 38.6735 \end{bmatrix}$$

$$\tau_{ng} = \begin{bmatrix} 4.3641 \\ 1.3955 \end{bmatrix}$$

3.1.2 Exercise 2 instance 2.

These are the values for the first configuration:

- $\theta_1 = \pi/2$
- $\theta_2 = \pi/4$
- $\dot{\theta_1} = -0.8\ rad/s$
- $\dot{\theta_2} = 0.35\ rad/s$
- $\ddot{\theta_1} = -0.4\ rad/s^2$
- $\ddot{\theta_2} = 0.1\ rad/s^2$

The result of the τ is the following (first with gravity, second without):

$$\tau_g = \begin{bmatrix} -65.0917 \\ -52.4313 \end{bmatrix}$$

$$\tau_{ng} = \begin{bmatrix} -12.3727 \\ 0.2878 \end{bmatrix}$$

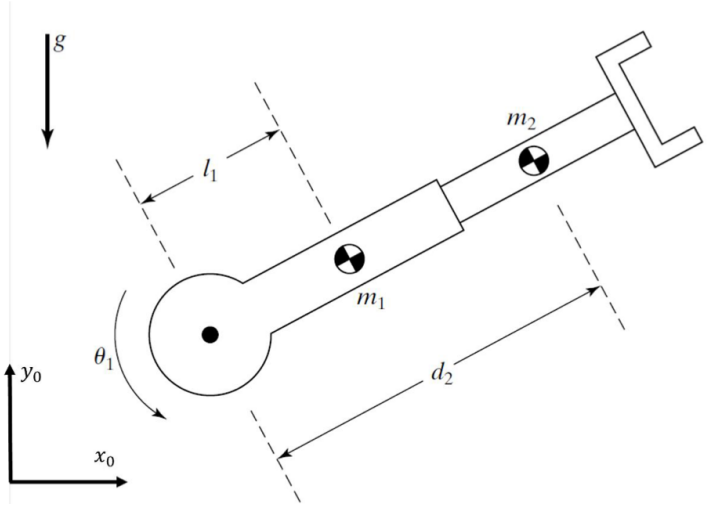
3.2 Exercise 3. Second Robot, RP chain

The second robot is a manipulator with one revolute joint and a prismatic joint. The frame of the first one has the rotational axis parallel to the z axis of the frame given. The second frame is a little bit tricky, so I had to compute two rotations, the first one with axis z by $\pi/2$, the second one with axis x by again $\pi/2$. The values that I will pass to the struct will depend on

the configuration that I will have to manage. The part of the struct in matlab that will change is the `robot.C`.

The robot characteristics:

- $m_1 = 10\ Kg$
- $m_2 = 6\ Kg$
- $l_1 = 1\ m$
- $I_{zz_1} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0.4 \end{bmatrix}$
- $I_{zz_2} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0.3 \end{bmatrix}$



3.2.3 Exercise 3 instance 1.

These are the values for the first configuration:

- $\theta_1 = \pi/9$
- $d_2 = 0.2\ m$
- $\dot{\theta_1} = 0.08\ rad/s$
- $v_2 = 0.03\ m/s$
- $\ddot{\theta_1} = 0.1\ rad/s^2$
- $a_2 = 0.01\ m/s^2$

The result of the τ is the following (first with gravity, second without):

$$\tau_g = \begin{bmatrix} 113.6529 \\ 20.1452 \end{bmatrix}$$

$$\tau_{ng} = \begin{bmatrix} 1.1886 \\ 0.0139 \end{bmatrix}$$

3.2.3 Exercise 3 instance 2.

These are the values for the first configuration:

- $\theta_1 = 2/3 \cdot \pi$
- $d_2 = 0.6\ m$
- $\dot{\theta_1} = -0.4\ rad/s$

- $v_2 = -0.08\ m/s$
- $\ddot{\theta_1} = -0.1\ rad/s^2$
- $a_2 = -0.01\ m/s^2$

The result of the τ is the following (first with gravity, second without):

$$\tau_g = \begin{bmatrix} -72.8246 \\ 49.3783 \end{bmatrix}$$

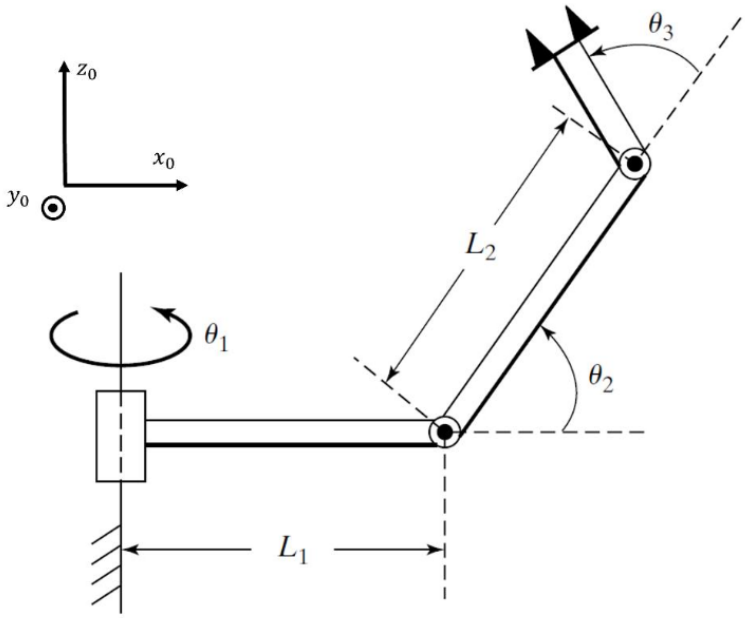
$$\tau_{ng} = \begin{bmatrix} -1.2116 \\ -1.5960 \end{bmatrix}$$

3.2 Exercise 3. Third Robot, RP chain

The third robot is a manipulator with three revolute joints. The frame of the first one has the rotational axis parallel to the z axis of the frame given. The second frame has a rotation around x of $\pi/2$. The difference between the other robots is the gravity, which is along z axis, while in the other ones it was along y. The third one is the same of the second. The values that I will pass to the struct will depend on the configuration that I will have to manage. The part of the struct in matlab that will change is the `robot.C`.

The robot characteristics:

- $m_1 = 20\ Kg$
- $m_2 = 20\ Kg$
- $m_3 = 6\ Kg$
- $l_1 = 1\ m$
- $l_2 = 0.8\ m$
- $l_3 = 0.35\ m$



$$I_{zz_1}\begin{bmatrix}0.2 & 0 & 0\\ 0 & 0.2 & 0\\ 0 & 0 & 0.4\end{bmatrix}\ kgm^2\ \ I_{zz_2}\begin{bmatrix}0.2 & 0 & 0\\ 0 & 0.2 & 0\\ 0 & 0 & 0.4\end{bmatrix}\ kgm^2 I_{zz_3}\begin{bmatrix}0.08 & 0 & 0\\ 0 & 0.08 & 0\\ 0 & 0 & 0.1\end{bmatrix}\ k$$

3.2.3 Exercise 3 unique instance.

These are the values for the first configuration:

- $\theta_1 = \pi/9$
- $\theta_2 = 2 \cdot \pi/9$
- $\theta_2 = \pi/18$
- $\dot{\theta_1} = 0.2\ rad/s$
- $\dot{\theta_2} = 0.15\ rad/s$
- $\dot{\theta_3} = -0.2\ rad/s$
- $\ddot{\theta_1} = 0.1\ rad/s^2$
- $\ddot{\theta_2} = 0.085\ rad/s^2$
- $\ddot{\theta_3} = 0\ rad/s^2$

The result of the τ is the following (first with gravity, second without):

$$\tau_g = \begin{bmatrix}5.1128\\ 104.1829\\ 6.7743\end{bmatrix}$$

$$\tau_{ng} = \begin{bmatrix}5.1128\\ 1.3712\\ 0.1532\end{bmatrix}$$

---

4. Conclusion.

The function on matlab is full of comments which should make it readable. It was pretty tough having every module of each vector with respect to base frame but with a little bit of patience I finally got a code that I think is clean and understandable. I will show again for the last time the data in order to complete the report.

Exercise 1 Instance 1, gravity and not gravity.

tau{1, 1}

1	2
318.1937	38.6735

tau{1, 2}

1	2
4.3641	1.3955

Exercise 1 Instance 2, gravity and not gravity.

tau{1, 1}

1	2
-65.0917	-52.4313

tau{1, 2}

1	2
-12.3727	0.2878

Exercise 2 Instance 1, gravity and not gravity.

tau{1, 1}

1	2
113.6529	20.1452

tau{1, 2}

1	2
1.1886	0.0139

Exercise 2 Instance 2, gravity and not gravity.

tau{1, 1}

1	2
-72.8246	49.3783

tau{1, 2}

1	2
-1.2116	-1.5960

Exercise 3 Instance 1, gravity and not gravity.

tau{1, 1}

1	2	3
5.1128	104.1829	6.7743

tau{1, 2}

1	2	3
5.1128	1.3712	0.1532