Università degli Studi di Genova

CORSO DI LAUREA MAGISTRALE

Robotics Engineering

ROBOT DYNAMICS AND CONTROL

Assignment Two Report

Presented By:

Youssef Mohsen Mahmoud Attia

5171925

Supervised By:

Prof. Giorgio Cannata

DECLARATION

I assure that this material, I now submit for the assessment on the Robot Dynamics And Control course first assignment is with no hesitation my own work, and that I have implemented reasonable care to guarantee the originality of the work presented. This material does not to the best of understanding violate any law of copyright and has not been stolen or copied from the work of others. The information and evidence shown has been cited, analysed and acknowledged within the text

Signed: *Youssef Attia*

Registration No.: 5171925

Date: 7th of June 2022.

ABSTRACT

This Report shows all the results for the exercise proposed as Robot Dynamics and Control Second assignment, all the results are motivated with an explanation. Furthermore, the algorithm used which is Newton-Euler algorithm for computing the torques on each robot joint is illustrated with the mathematics used along with the MATLAB script.

Table of Contents

I. Introduction

The assignment is composed of three robots. Firstly, a revolut-revolut (RR) robot in 2D and another revolut-prismatic (RP) and lastly, RRR robot. All three robots have been taken into consideration and applied the **recursive Newton-Euler algorithm** in which the torque needed on each joint to do a specific motion is computed knowing the robot geometric design (mass, moment of inertia, ect.) The algorithm is applied using MATLAB and is further explained in detail in the report.

Recursive Newton-Euler algorithm

Newton–Euler equations describe the combined translational and rotational dynamics of a rigid body. The Newton–Euler equations is the grouping together of Euler's two laws of motion for a rigid body into a single equation with 6 components, using column vectors and matrices. These laws relate the motion of the center of gravity of a rigid body with the sum of forces and torques (or synonymously moments) acting on the rigid body.

This algorithm can be summarized as follows as it is divided into two separate recursive equations (Forward and Backward recursion). All the mathematical equations are presented along with the MATLAB script responsible for computing this section.

A. Forward recursions:

1. Position calculation:

$$r_{i/i-1} = \begin{cases} P_i - P_{i-1}\ ; & i \in RJ \\ (P_i - P_{i-1}) + k_i q_i\ \ ; & i \in TJ \end{cases}$$

```
    %Step 1 (postion):
    if robotlink(:,i-1)==0
        r(:,i-1)=pi(:,i)-pi(:,i-1);
    else
        r(:,i-1)=pi(:,i)-pi(:,i-1)+k*q(i-1);
    end
```

2. Velocity calculation:

2.1) Angular velocity:

$$\omega_{i/0} = \begin{cases} \omega_{i-1/0} + k_i \dot{q}_i\ \ ; i \in RJ \\ \omega_{i-1/0}\ \ ; \ i \in TJ \end{cases}$$

```
%Step 2 (velocity):
    %Angular:
    if robotlink(:,i-1)==0
        w(:,i)=w(:,i-1)+k*qdot(i-1);
    else
        w(:,i)=w(:,i-1);
    end
```

2.2) Linear velocity:

$$v_{i/0} = \begin{cases} v_{i-1/0} + \omega_{i-1/0} \times r_{i/i-1} \ ; i \in RJ \\ (v_{i-1/0} + \omega_{i-1/0} \times r_{i/i-1})\, k_i \dot{q}_i \ ; \ i \in TJ \end{cases}$$

```
%liner:
    if robotlink(:,i-1)==0
        v(:,i)=v(:,i-1)+cross(w(:,i-1),r(:,i-1));
    else
        v(:,i)=v(:,i-1)+cross(w(:,i-1),r(:,i-1))+k*qdot(i-1);
    end
```

2.3) Velocity at center of mass:

$$v_{c_i/0} = v_{i/0} + \omega_{i/0} \times r_{c_i/i}$$

```
%Linear wrt com:
    rcom(:,i-1)=picom(:,i)-picom(:,i-1);
    vcom(:,i)=v(:,i)+cross(w(:,i), rcom(:,i-1));
```

3. Acceleration calculation:

3.1) Angular acceleration:

$$\dot{\omega}_{i/0} = \begin{cases} \dot{\omega}_{i-1/0} + (\dot{\omega}_{i-1/0} \times k_i) + k_i \ddot{q}_i \ ; i \in RJ \\ \dot{\omega}_{i-1/0} \ ; \ i \in TJ \end{cases}$$

```
%Angular:
    if robotlink(:,i-1)==0
        wdot(:,i)=wdot(:,i-1)+cross(w(:,i-1),k)+k*qdotdot(i-1);
    else
        wdot(:,i)=wdot(:,i-1);
    end
```

3.2) Linear acceleration:

$$\dot{v}_{i/0} = \begin{cases} \dot{v}_{i-1/0} + (\dot{\omega}_{i-1/0} \times r_{i/i-1}) + \omega_{i-1/0} \times (\omega_{i-1/0} \times r_{i/i-1}) \ ; i \in RJ \\ \dot{v}_{i-1/0} + (\dot{\omega}_{i-1/0} \times r_{i/i-1}) + \omega_{i-1/0} \times (\omega_{i-1/0} \times r_{i/i-1}) + 2(\omega_{i-1/0} \times k_i)\dot{q}_i + k_i \ddot{q}_i \ ; \ i \in TJ \end{cases}$$

```
%Linear:
    if robotlink(:,i-1)==0
        vdot(:,i)=vdot(:,i-1)+cross(wdot(:,i-1),r(:,i-1))+cross(w(:,i-1),cross(w(:,i-
1),r(:,i-1)));
    else
        vdot(:,i)=vdot(:,i-1)+cross(wdot(:,i-1),r(:,i-1))+cross(w(:,i-1),cross(w(:,i-
1),r(:,i-1)))+2*cross(w(:,i-1),k)*qdot(i-1)+k*qdotdot(i-1);
    end
```

3.3) Acceleration at center of mass:

$$\dot{v}_{c_i} = \dot{v}_{i/0} + \dot{\omega}_{i/0} \times r_{c_i/i} + \omega_{i/0} \times (\omega_{i/0} \times r_{c_i/i})$$

```
%Linear wrt com:
    vdotcom(:,i)=vdot(:,i)+cross( wdot(:,i), rcom(:,i-1))
                 +cross(w(:,i),cross(w(:,i),rcom(:,i-1)));
```

B. Backward recursions:

1. Forces and Moments calculation:

$$D_i = m_i \dot{v}_{c_i/0}$$

$$\Delta_i = I_{c_i} \dot{\omega}_{i/0} + (\omega_{i/0} \times I_{c_i} \omega_{i/0})$$

$$F_{i/i-1} = F_{i+1/i} - F_{c_i}^{ext} + D_i - m_i g_i$$

$$M_{i/i-1} = M_{i+1/i} - M_{c_i}^{ext} - (r_{i/c_i} \times F_{i/i-1}) + (r_{i-1/c_i} \times F_{i+1/i}) + \Delta_i$$

```
D=m(i)*vdotcom(:,i+1);
    deltai=I(:,(i*3)-2:i*3)*wdot(:,i+1)+cross(w(:,i+1),I(:,(i*3)-2:i*3)*w(:,i+1));
    F(:,i)=F(:,i+1)-Fext+D-m(i)*g;
    M(:,i)=M(:,i+1)-Mext-cross(-rcomnew(:,i+1),F(:,i))
           +cross(-rcomnew(:,i),F(:,i+1))+deltai;
```

2. Torque calculation:

$$\tau = \begin{cases} M_{i/i-1} * k_i \ ; \ i \in RJ \\ F_{i/i-1} * k_i \ ; \ i \in TJ \end{cases}$$

```
    if robotlink(i)==0
        Torque=M(:,i).*k
    else
        Torque=F(:,i).*k
    end
```

Please note that for the previous recursion the forward recursion starts from link 0 to link n and the backward recursion starts from link n descending to link 0 where n is the number of degrees of freedom of the robot.

II. Generic function

For this algorithm summarized before a general function is implemented to calculate torque on any robot joints regarding it geometry.

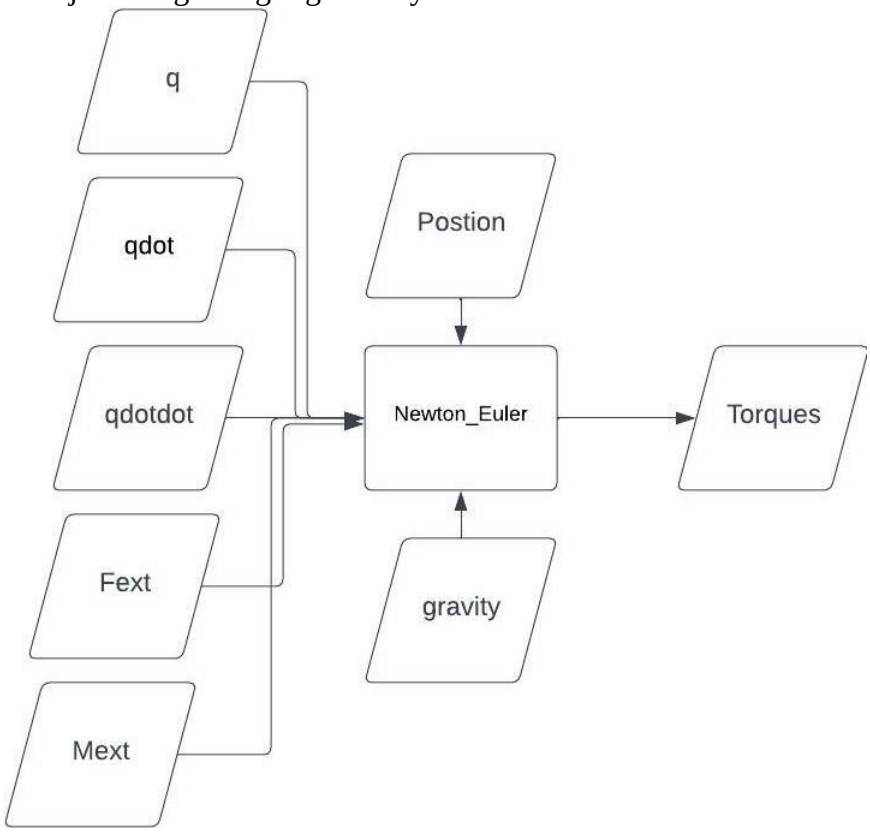
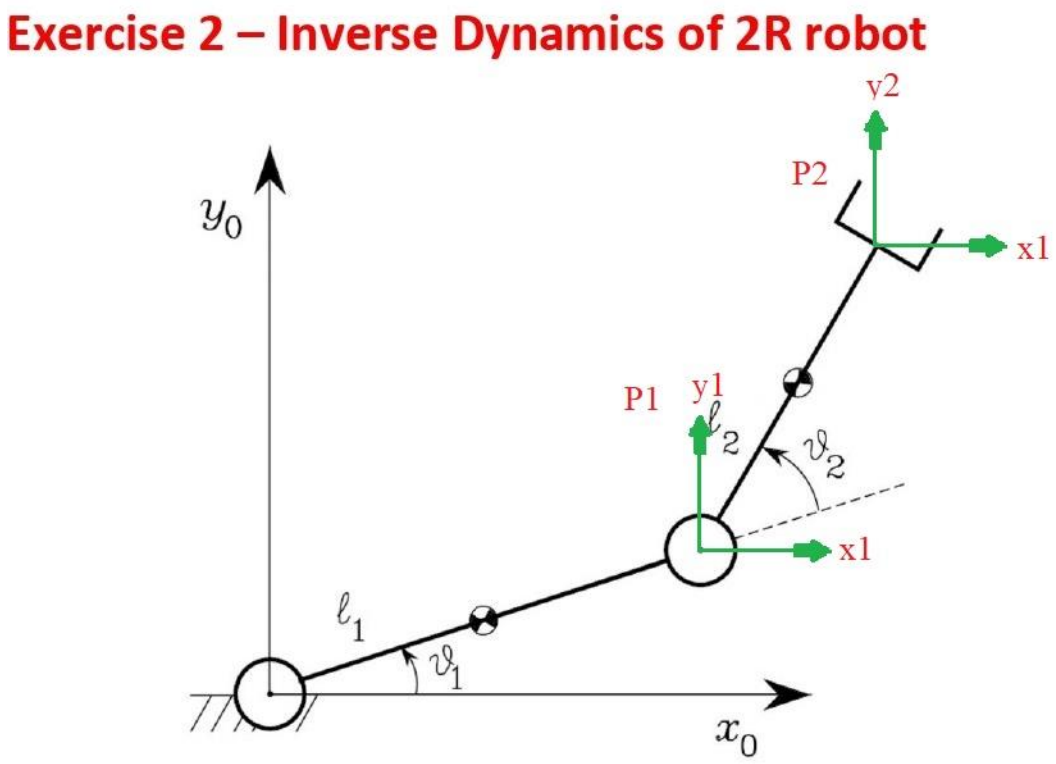


Figure 1: Newton Eular function flowchart

Figure 1 shows the input parameters of the generic function and its expected output, as for the implementation done for this assignment some other parameters were passed for the function like the number of degrees of freedoms, also the positions of center of mass of each link just for simplicity but for further enhancements these parameters can be computed from the other passed parameters. Like for example, the number of degrees of freedom can be found from the size of q. Also, the position of the center of masses and the links frames can be understood and computed from q.

III. Case study & Results

Let's take each robot geometric design and analyze it finding all the points needed for calculating: $r_{i/i-1}$ and $r_{c_i/i}$ and further find the results of the torques for each exercise using the algorithm illustrated above.



For the following figure (x_1, y_1, z_1) and (x_2, y_2, z_2) can be computed using simple trigonometric rules:

$$P_1 = \begin{matrix} l_1 cos\theta_1 \\ l_1 sin\theta_1 \\ 0 \end{matrix}$$

$$P_2 = \begin{matrix} l_1 cos\theta_1 + l_2 cos(\theta_1 + \theta_2) \\ l_1 sin\theta_1 + l_2 sin(\theta_1 + \theta_2) \\ 0 \end{matrix}$$

Using the points of each frame with reference to the origin and the known parameters of the robot it was found the following torques:

Table 1: RR Robot results

	Without gravity	With gravity
Exercise 2.1	$\tau_2 = 1.2259\ N.m$ $\tau_1 = 2.442\ N.m$	$\tau_2 = 38.5039\ N.m$ $\tau_1 = 228.6969\ N.m$
Exercise 2.2	$\tau_2 = -3.4340\ N.m$ $\tau_1 = -12.6172\ N.m$	$\tau_2 = 33.844\ N.m$ $\tau_1 = 213.6377\ N.m$

Exercise 3 – Inverse Dynamics of RP robot

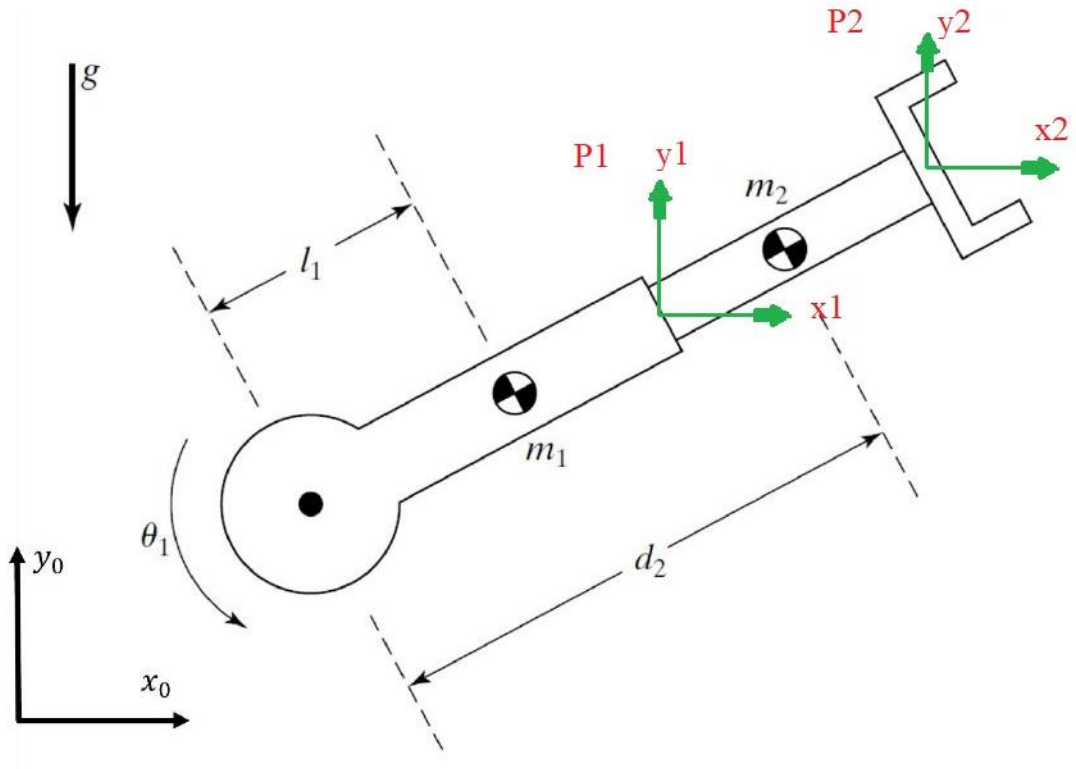


Figure 2: RP Planar manipulator

$$P_1 = \begin{matrix} l_1 cos\theta_1 \\ l_1 sin\theta_1 \\ 0 \end{matrix}$$

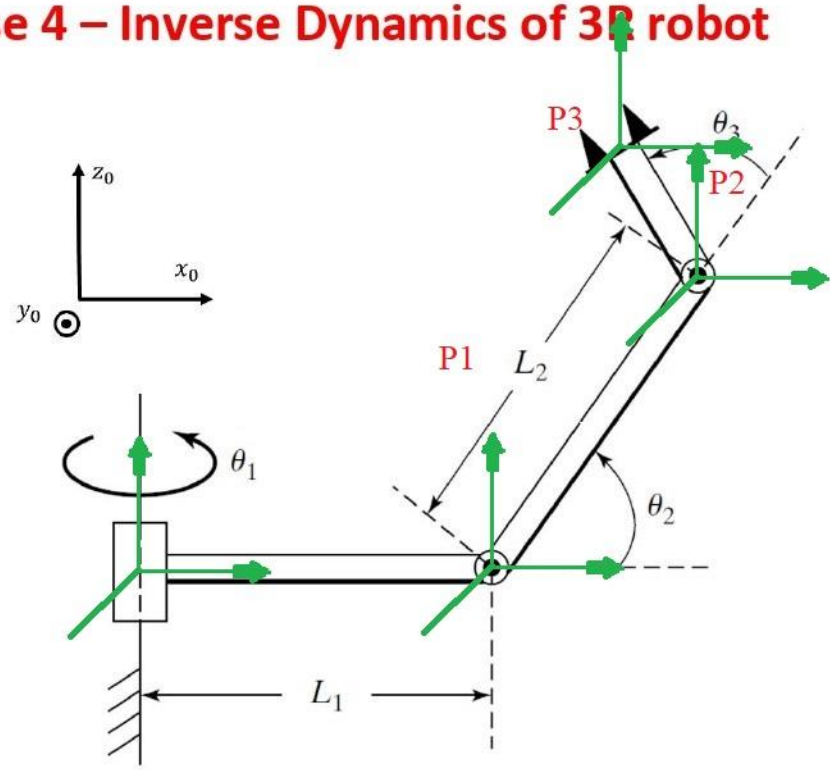
$$P_2 = \begin{matrix} (l_1 + d2) cos\theta_1 \\ (l_1 + d2)\ sin\theta_1 \\ 0 \end{matrix}$$

Using the points of each frame with reference to the origin and the known parameters of the robot it was found the following torques:

Table 2: RP Robot results

	Without gravity	With gravity
Exercise 3.1	$\tau_2 = 0.0516\ N.m$ $\tau_1 = 0.4346\ N.m$	$\tau_2 = -0.0516\ N.m$ $\tau_1 = -0.4346\ N.m$
Exercise 3.2	$\tau_2 = -3.4340\ N.m$ $\tau_1 = -12.6172\ N.m$	$\tau_2 = 5.4794\ N.m$ $\tau_1 = 78.8435\ N.m$

Exercise 4 – Inverse Dynamics of 3R robot



For this complex RRR robot finding its points using basic trigonometric rules is hard so Denavit–Hartenberg "DH" method was used in which frames were chosen according to the direction of motion of the joint (Z-axis) and then the DH table 3 was found and using these parameters the transformation function was computed using MATLAB Robotics Toolbox by Peter Corke and therefore the points at the end effector was found and further used in the Newton algorithm.

Table 3: RRR Robot DH table

	θ	d	a	α
Link 1	θ_1	0	l_1	90
Link 2	θ_2	0	l_2	0
Link 3	θ_3	0	l_3	0

Where:

θ_i: angle between X_i and X_{i-1}about Z_{i-1}

d_i: distance between X_i and X_{i-1}along Z_{i-1}

a_i: distance between Z_i and Z_{i-1}along X_i

$\propto_i$: angle between Z_i and Z_{i-1}about X_i

$$P_1 = \begin{matrix} l_1 cos\theta_1 \\ l_1 sin\theta_1 \\ 0 \end{matrix}$$

$$P_2 = \begin{matrix} l_1 cos\theta_1 + l_2 cos(\theta_1 + \theta_2) \\ l_1 sin\theta_1 + l_2 sin(\theta_1 + \theta_2) \\ 0 \end{matrix}$$

$$P_3 = \begin{matrix} l_1 cos\theta_1 + l_2 cos(\theta_1)\cos(\theta_2) + l_3 cos(\theta_1)\cos(\theta_2)\cos(\theta_3) - l_3 cos(\theta_1)\sin(\theta_2)\sin(\theta_3) \\ l_1 sin\theta_1 + l_2 \cos(\theta_2)\sin(\theta_1) + l_3 \cos(\theta_2)\cos(\theta_3) sin\theta_1 - l_3 sin\theta_1 \sin(\theta_2)\sin(\theta_3) \\ l_2 \sin(\theta_2) + l_3 \cos(\theta_2)\sin(\theta_3) + l_3 \cos(\theta_3)\sin(\theta_2) \end{matrix}$$

Using the points of each frame with reference to the origin and the known parameters of the robot it was found the following torques:

Table 4: RRR Robot results

	Without gravity	With gravity
Exercise 4	$\tau_3 = 0.639\ N.m$ $\tau_2 = 4.8593\ N.m$ $\tau_1 = 7.7092\ N.m$	$\tau_3 = 28.9849\ N.m$ $\tau_2 = 176.401\ N.m$ $\tau_1 = 391.2737\ N.m$

IV. Conclusion

In concurrent the algorithm successfully computed the torques of each joint for all the three robots proposed in the assignment and with some fine adjustments to the function it can be more generic and used in other applications. Moreover, if a closer look is given to the output torques it can be found that its directly proportional with the gravity and this make sense for sure “regarding the direction”. Also, the torques are directly proportional to the velocity and acceleration of the link and this can be noticed from Table 1 and Table 2.