# Robot Dynamics and Control Assignment.

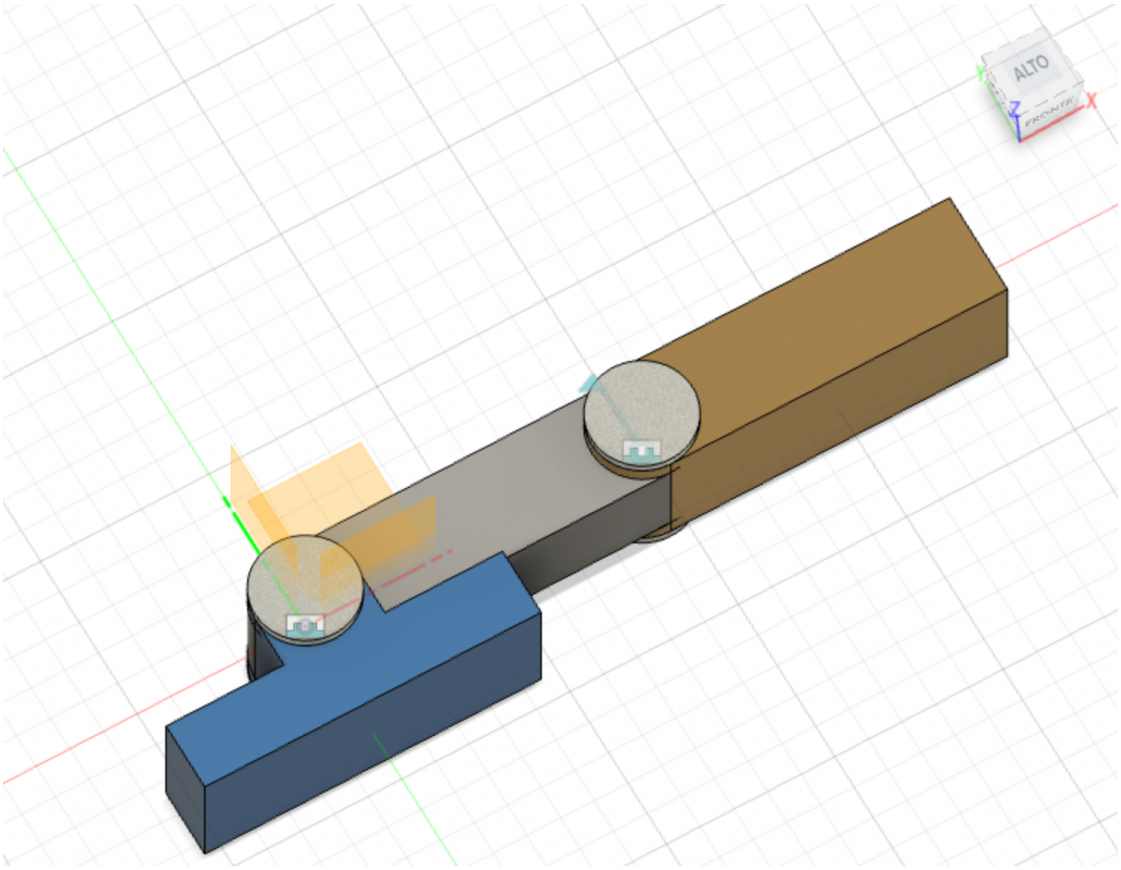
📍 Luca Predieri, mat. 4667708

The first assignment of Robot Dynamics and Control class in Robotics Engineering Course, regards the study of two exercises. These two exercises face the problem of the equilibrium of two manipulators. The manipulators have two degrees of freedom. The first one presents two revolute joints, the second one prismatic and one revolute.

## 0. Table of contents.

# 1. CAD Model.

In order to have a good model of the manipulator what I had to do was to model the robot in Fusion. To have exact parameters when transferring the data to matlab, I had to be sure that the frames of the links and motors of the manipulators where set according to *Denavit - Hartenberg* convention.




In the first exercise I will have different frames than in the second, because in the second case, for the prismatic joint, I'll have a different configuration for the frame 1.

Then, I exported the Assembly with the extensions Ms. Baldini told us. I opened the model in Inventor and, using the plug-in *SMIdata* I exported the file .XML needed to use the data we needed from the Assembly.

These process was done twice because I created two different CAD models.

# 2. Transfer of the data to Matlab.

## 2.1 Importing the data.

To treat the data I had to install the Simscape Multibody add-in for Matlab, after modifying some issues for the .XML file, I imported the data using:

```
smimport('Assembly.xml')
```

Then I obtained a SMIdata file, containing different data. Some of it it's useful some other it's useless. The assembly data files are two, because of the two configurations of the manipulator.

The files are divided in two structures, the first one is `smiData.RigidTransform(i)` which gives me data about the position of the parts in the assembly with $i = 1, 2, 3, 4, 5$ (numbers of the parts in the assembly) respecting the frame of the link and the base frame. I used the attribute `smiData.RigidTransform(i).translation` which tells me the distance between the base frame and the frame referred to the object. The second structure is the `smiData.Solid(i)` which gives us data about the object itself with $i = 1, 2, 3, 4$ (numbers of the unique parts in the assembly). I used `smiData.Solid(i).mass` to get the bodies' masses and `smiData.Solid(i).CoM` which tells me the center of mass of the bodies with respect to their frames.

## 2.2 Run the matlab.

To run the matlab script, open the `PredieriLuca_Assignment_1_solution` folder in matlab and run `PredieriLuca_assignment_1_solution.m`. This should call all the function files created for making the code lighter and show the results in the workspace tab of matlab.

# 3. Mathematical approach.

As far as this exercise is about the equilibrium of the manipulator and finding I'll have to study all the wrenches related to the body and finding $\tau_{eq}$. In the final computation we will have to face different contributes. The possible contributes are the following:

- External Forces
- External Torques
- Gravity Forces.

To solve all these contributes and finding the final torques we need to deepen the jacobian matrices and all their construction.

## 3.1 Rotational and Transformation matrices.

In order to treat all the distances with respect to different frames, I need to compute many rotational matrices to obtain the transformation matrices.

The matrices we will need for transformation will be the following:

$$R_z = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad R_y = \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix}$$

The first one (the rotation around z axis) is needed for computing the transformation matrices of all the joints except the first joint of the second configuration (the prismatic one) because the *Denavit - Hartenberg* convention wants the z-axis along the translation in case of prismatic joint.

As long as in ours configuration I only have one frame distant from the base frame in *matlab* script we only computed one distance, which is the one between the base frame and the second frame. This distance will be computed for the transformation matrix from base frame to frame 2. To be sure of the future computations, I decided to compute both the **transformation matrix** and the **homogeneous transformation matrix**.

In general the the transformation matrices will be the following (homogeneous):

$$T_{hom_1} = \begin{bmatrix} R_1 & zero(3,1) \\ zero(1,3) & 1 \end{bmatrix} \qquad T_{hom2} = \begin{bmatrix} R_2 & dist_{b-f} \\ zero(1,3) & 1 \end{bmatrix}$$

Where $zero(i,j)$ is a matrix with $i$ rows and $j$ columns. The first matrix is referred to the first frame and the second to the second frame. **As far as the first frame is at the base, we have a zero distance instead of the second one which has a distance.**

⚠️ The distance between the frame 1 and frame 2 in the first exercise is obtained by always premultiplying the rotational matrix with the `.translation(i)` given by the data file, because the frame 2 is changing its position in the space.

## 3.2 Jacobian matrix.

In order to finish the computation of the Jacobian matrix I need to know the contributes. In general, as far as I have 2 joints, I know that the Jacobian will have 6 rows and 2 columns.

So the structure will be like this:

$$J_{ex,joint} = \begin{bmatrix} J_{a1} & J_{a2} \\ J_{l1} & J_{l2} \end{bmatrix}$$

Where for $ex$ I mean the different configuration for exercise, for $joint$ I mean the joint the jacobian is referred to. At the end we will have 20 jacobians (2 exercises, 5 points for each and 2 joints).

The two kind of matrices $J_a$ and $J_b$ are by definition this:

**For prismatic joints.**

$$J_{a-prism} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \qquad J_{l-prism} \begin{bmatrix} k_x \\ k_y \\ k_z \end{bmatrix}$$

**For revolute joints.**

$$J_{a-rev} = \begin{bmatrix} k_x \\ k_y \\ k_z \end{bmatrix} \qquad J_{l-rev} \left[ \vec{k} \times \vec{r} \right]$$

Where $\vec{k}$ is the axis around which the frame rotates and $\vec{r}$ is the distance between the frame of the link and the center of mass with respect to the frame.

## 3.3 Rigid Body Jacobian.

In the case we had some forces acting on the link not applied on the center of mass of the link, I had to refer it to the point of application. So, we want a matrix S such that:

$$\begin{bmatrix} M_{c_i}^{ext} \\ F_{c_i}^{ext} \end{bmatrix} = S_{p_i/c_i}^{t} \begin{bmatrix} M_{i}^{ext} \\ F_{i}^{ext} \end{bmatrix}$$

The matrix that helps doing it is the rigid jacobian matrix, which structure is like this:

$$S_{p_i/c_i} = \begin{bmatrix} \mathbb{I}_{3,3} & zeros(3,3) \\ -[r_{c_i/p_i}] & \mathbb{I}_{3,3} \end{bmatrix}$$

$r_{c_i/p_i}$ has the following structure:

$$r_{c_i/p_i} = \begin{bmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{bmatrix}$$

$-[r_{c_i/p_i}]$ is the skew matrix and $\mathbb{I}_{3,3}$ is an identical matrix 3x3. So the rigid body jacobian has 6 rows and 6 columns.

If you multiply the skew matrix and a vector in the space you obtain the vectorial product between these two, this is the core of the S matrix, which helps us to obtain the wrench we want to multiply with the jacobian matrix calculated on the center of mass.

## 3.4 Computing the equilibrium torques of the manipulator.

Before showing the formula prof. Cannata explained us, I want to remember the structure of $\tau_{eq}$ which is:

$$\tau_{eq} = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix}$$

Where $\tau_1$ is the one related to the first joint while $\tau_2$ is related to the second joint.

In general, what I will use is the following relation:

$$\tau_{eq} = -J^T W_{ext} \tag{1}$$

Where $J^T$ is the transpose of the jacobian matrix of a certain part of an exercise and $W_{ext}$ is the wrench acting on the center of mass. As I said at the beginning of this section, we have three contributes, one for the **external forces**, one for **external torques** and one for **gravity forces**.

In general, depending by one of these contributes, the torque changes. I will develop the contributes in every part of each exercise, trying to highlight the differences and the way the wrenches change. It though important to always remember the main relation which is the one explained before.

# 4. Answers to the exercises.

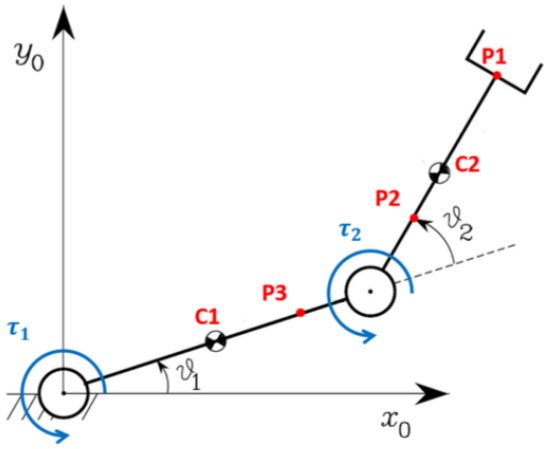
Now I'm going to explain all the computational reason behind each point, explaining the contributes and showing the results I obtained from the *matlab* script. The matlab script is highly commented so it's easier to understand, most of the computation is repetitive so I decided to automatize it a little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

I remember that every time there is a jacobian matrix of a joint, it is referred to that particular configuration. So, even if they have the same name in each configuration, they are different. It is just because of the several subscripts and superscripts, it would've become unreadable.

## 4.1 Exercise 1.

Compute the robot generalized joint forces required to balance external forces acting on the robot under the following conditions. There are 3 different points:

- P1 is located at the end of the second link.
- P2 is located -20 cm from C2 .
- P3 at 40 cm from C1.



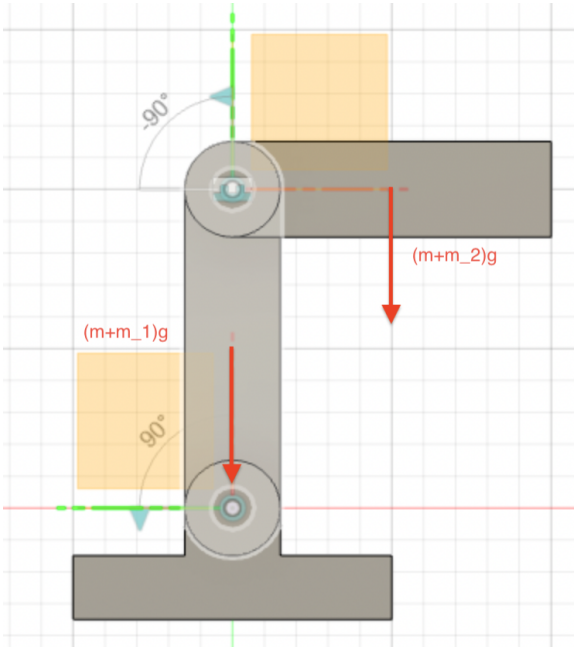
### 4.1.1 First configuration.

Angles:

- $\theta_1 = \frac{\pi}{2}$
- $\theta_2 = -\frac{\pi}{2}$

Forces:

- $(m_{motor} + m_{link1})\vec{g}$ on CoM of link 1.
- $(m_{motor} + m_{link2})\vec{g}$ on CoM of link 2.




We have here two contributes, the forces of gravity of both the links, they're applied on the center of mass of the two links. The equation (1) will be like this:

$$\tau_{eq} = -J^T_{link1} W_1 - J^T_{link2} W_2$$

where

$$W_1 = \begin{bmatrix} zeros(3,1) \\ (m_1 + m_{motor}) * g \end{bmatrix}$$

$$W_2 = \begin{bmatrix} zeros(3,1) \\ (m_2 + m_{motor}) * g \end{bmatrix}$$

and $-J^T_{link-i}$ is the jacobian matrix of the following configuration of the i-th link, it can be found on the cell J in the workspace of *matlab*. The solution is:

$$\tau_{eq} = \begin{bmatrix} 5.7997 \\ 5.7997 \end{bmatrix} 10^5 Nmm$$

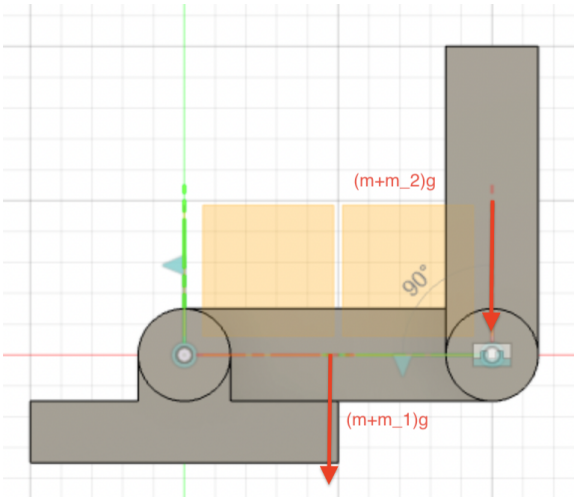
### 4.1.2 Second configuration.

Angles:

- $\theta_1 = 0$
- $\theta_2 = \frac{\pi}{2}$

Forces:

- $(m_{motor} + m_{link1})\vec{g}$ on CoM of link 1.
- $(m_{motor} + m_{link2})\vec{g}$ on CoM of link 2.




We have the same contributes as before but in a different configuration. So the equation (1) will be like this:

$$\tau_{eq} = -J^T_{link1} W_1 - J^T_{link2} W_2$$

where

$$W_1 = \begin{bmatrix} zeros(3,1) \\ (m_1 + m_{motor}) * g \end{bmatrix} \qquad W_1 = \begin{bmatrix} zeros(3,1) \\ (m_1 + m_{motor}) * g \end{bmatrix}$$

and $-J^T_{link-i}$ is the jacobian matrix of the following configuration of the i-th link, it can be found on the cell J in the workspace of *matlab*. The solution is:

$$\tau_{eq} = \begin{bmatrix} 1694925.7016590 \\ 0 \end{bmatrix} Nmm$$

$\tau_2$ is equal to 0 because the force is parallel to the distance vector.

---

### 4.1.3 Third configuration.

Angles:

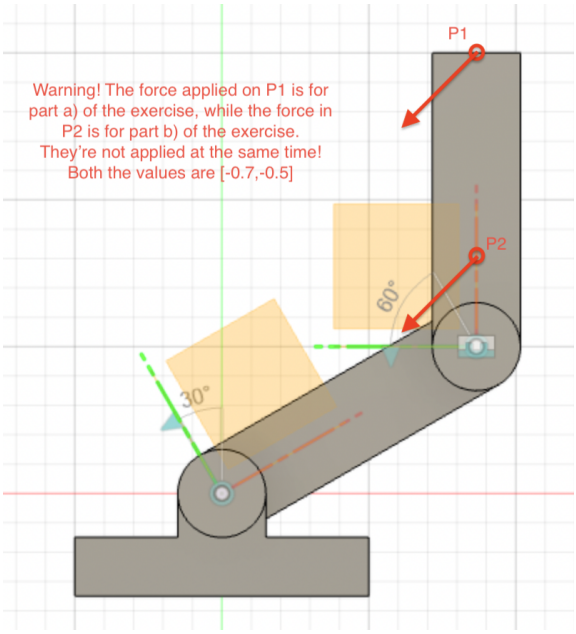
- $\theta_1 = \frac{\pi}{6}$
- $\theta_2 = \frac{\pi}{3}$

Forces:

- $[-0.7, 0.5]N$ applied on P1 for part a)
- $[-0.7, 0.5]N$ applied on P2 for part b)

Here the manipulator has an external force not applied on the center of mass, we have to use the rigid jacobian matrix. In both the part of the problem 1.3 we'll have:

$$\tau_{eq} = -J^T_{link2} S^T_{p_i/c_2} W_2$$




where $i = 1, 2$. For part a) $i = 1$, for part b) $i = 2$. The wrench $W_2$ is the following:

$$W_2 = \begin{bmatrix} zeros(3,1) \\ \begin{bmatrix} -0.7 \\ 0.5 \\ 0 \end{bmatrix} \end{bmatrix}$$

I'll have as equilibrium torques:

$$\tau_{eq_a} = \begin{bmatrix} -616.9872981077 \\ -700 \end{bmatrix} Nmm \qquad \tau_{eq_a} = \begin{bmatrix} -126.9872981077 \\ -210 \end{bmatrix} Nmm$$

As we can conclude, the second torque is easier to balance than the first one, because the distance between the frame of the joint and the point of the application is much smaller.

---

### 4.1.4 Fourth configuration.

Angles:

- $\theta_1 = \frac{\pi}{6}$
- $\theta_2 = \frac{\pi}{3}\theta_1 = \frac{\pi}{3}$

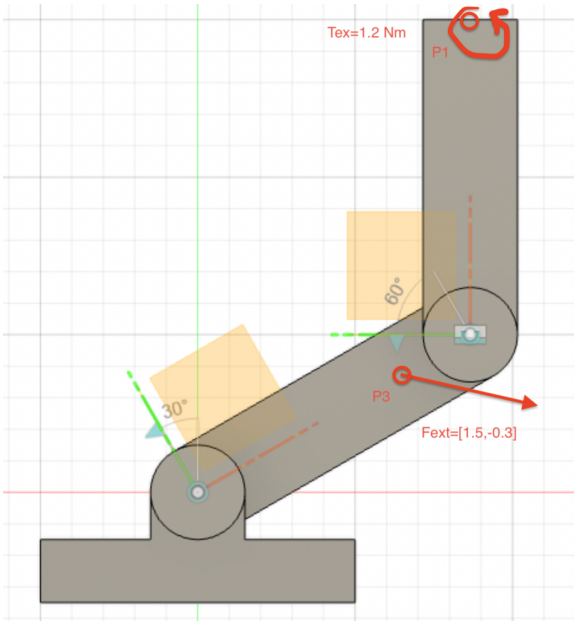
Forces:

- $[1.5, -0.3] N$ applied on P3.

Torques:

- $\tau_{ext} = 1.2Nm$ on link 2.

Now the manipulator has an external force on link 1 not applied on the center of mass and an external torque on link 2. We have to use the rigid jacobian matrix for the external force and pay attention to the wrenches in both the contributes. We'll have:




$$\tau_{eq} = -J_{link1}^T S_{p_3/c_1}^T W_1 - J_{link2}^T W_2$$

where

$$W_1 = \begin{bmatrix} zeros(3,1) \\ \begin{bmatrix} 1.5 \\ -0.3 \\ 0 \end{bmatrix} \end{bmatrix} \qquad W_2 = \begin{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1200 \end{bmatrix} \\ zeros(3,1) \end{bmatrix}$$

The result of the torque to permit the equilibrium is:

$$\tau_{eq} = \begin{bmatrix} -291.173140978202 \\ -1200 \end{bmatrix} Nmm$$

---

### 4.1.5 Fifth configuration.

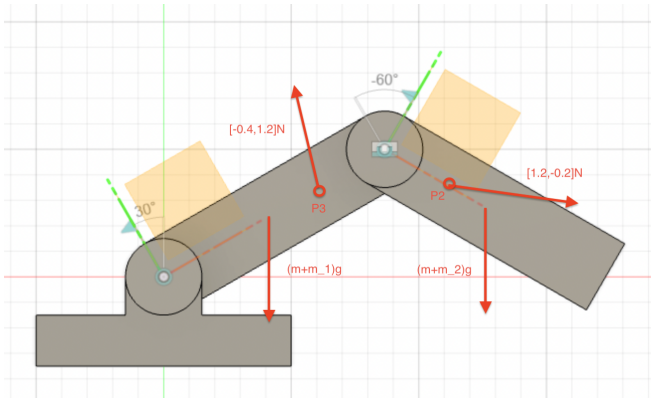
Angles:

- $\theta_1 = \frac{\pi}{6}$
- $\theta_2 = -\frac{\pi}{3}$

Forces:

- $[-0.4, 1.2]N$ applied on P3.
- $[1.2, -0.2]N$ applied on P2.

- $(m + m_1)\vec{g}$ applied on CoM1
- $(m + m_2)\vec{g}$ applied on CoM2




In this configuration we have four contributes, two external forces and two forces due to gravity. The idea is still similar to the other cases:

$$\tau_{eq} = -J^T_{link1} S^T_{p_3/c_1} W_{ext,1} - J^T_{link2} S^T_{p_2/c_2} W_{ext,2} - J^T_{link1} W_{grav,1} - J^T_{link2} W_{grav,2}$$

where

$$W_{ext,1} = \begin{bmatrix} zeros(3,1) \\ \begin{bmatrix} -0.4 \\ 1.2 \\ 0 \end{bmatrix} \end{bmatrix} \qquad W_{ext_2} = \begin{bmatrix} zeros(3,1) \\ \begin{bmatrix} 1.2 \\ -2 \\ 0 \end{bmatrix} \end{bmatrix}$$

$$W_{grav,1} = \begin{bmatrix} zeros(3,1) \\ (m_1 + m_{motor}) * g \end{bmatrix} \qquad W_{grav,2} = \begin{bmatrix} zeros(3,1) \\ (m_2 + m_{motor}) * g \end{bmatrix}$$

The resulting equilibrium torque is:

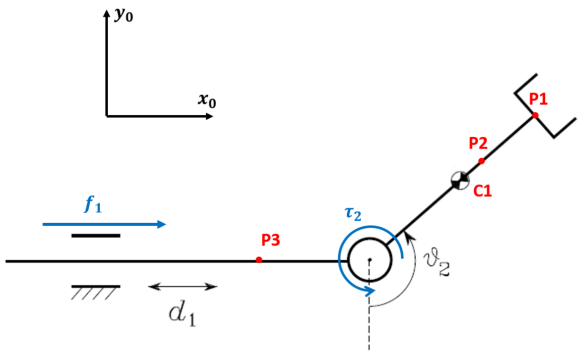
$$\tau_{eq} = \begin{bmatrix} 1969474.57914318 \\ -501967.966334646 \end{bmatrix} Nmm$$

---

## 4.2 Exercise 2.

Compute the robot generalized joint forces required to balance external forces acting on the robot under the following conditions. There are 3 different points:

- P1 is located at the end of the second link.
- P2 is located 15 cm from C2 .
- P3 at -20 cm from the axis of the revolut joint.




Warning! $\theta_2$ is calculated between the link 2 and the line passing through the prismatic joint. In the picture the angle is not corrected.

---

### 4.2.1 First configuration.

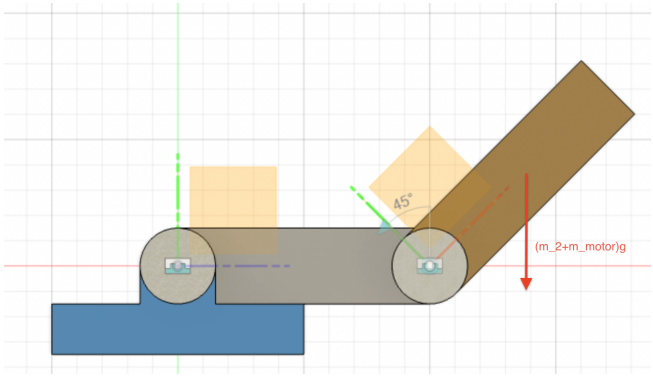
Angles:

- $\theta_2 = \frac{\pi}{4}$

Forces:

- $(m + m_2)\vec{g}$ applied on CoM2

This is a simple configuration where we only have the contribute of gravity of the second link,




The contribute of gravity of the first link is not considered because of the reaction force the ground creates. Indeed the equation (1) will be:

$$\tau_{eq} = -J_{link2}^T W_2 \qquad W_2 = \begin{bmatrix} zeros(3,1) \\ (m_2 + m_{motor}) * g \end{bmatrix}$$

Which gives me the solution:

$$\tau_{eq} = \begin{bmatrix} 0 \\ 409959.671225178 \end{bmatrix} Nmm$$

---

### 4.2.2 Second configuration.

Angles:

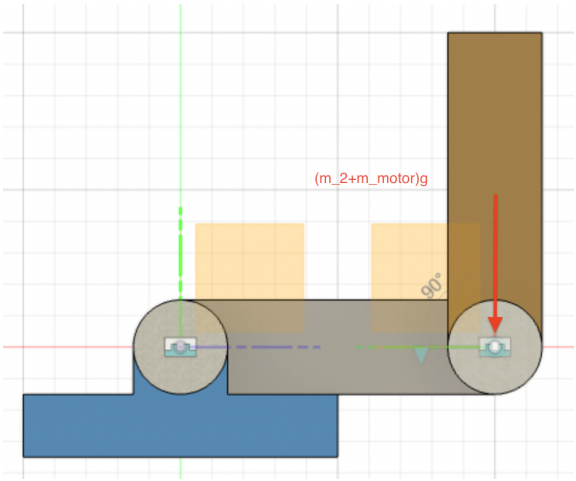
- $\theta_2 = \frac{\pi}{2}$

Forces:

- $(m + m_2)\vec{g}$ applied on CoM2

Again we only have the contribute of gravity of the second link,

The  contribute of gravity of the first link is not considered because of the reaction force the ground creates. Indeed the equation (1) will be:




$$\tau_{eq} = -J_{link2}^T W_2$$

$$W_2 = \begin{bmatrix} zeros(3,1) \\ (m_2 + m_{motor}) * g \end{bmatrix}$$

Which gives me the solution:

$$\tau_{eq} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} Nmm$$

As long as the gravity force has no contribution on the x-axis, the force cannot generate any torque on the joints. In mathematical terms, the vectorial product $\vec{F} \times \vec{r}$ is equal to 0 (they're parallel). The robot is in an equilibrium position itself.

---

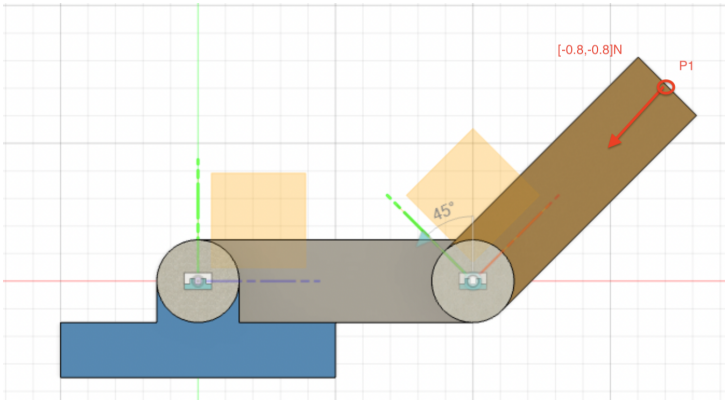
### 4.2.3 Third configuration.

- $\theta_2 = \frac{\pi}{4}$

Forces:

- $[-0.8, -0.8]N$ applied on P1.

Now we have only one external force acting on the top of the second link. Now we have to use the rigid jacobian in order to find the equilibrium torques.




$$\tau_{eq} = -J_{link2}^T S_{p_i/c_2}^T W_{ext,1}$$

With:

$$W_{ext,1} = \begin{bmatrix} zeros(3,1) \\ \begin{bmatrix} -0.8 \\ -0.8 \\ 0 \end{bmatrix} \end{bmatrix}$$

Which gives me the solution:

$$\tau_{eq} = \begin{bmatrix} 0.8000 \\ 0 \end{bmatrix} Nmm$$

---

### 4.2.4 Fourth Configuration.

Angles:

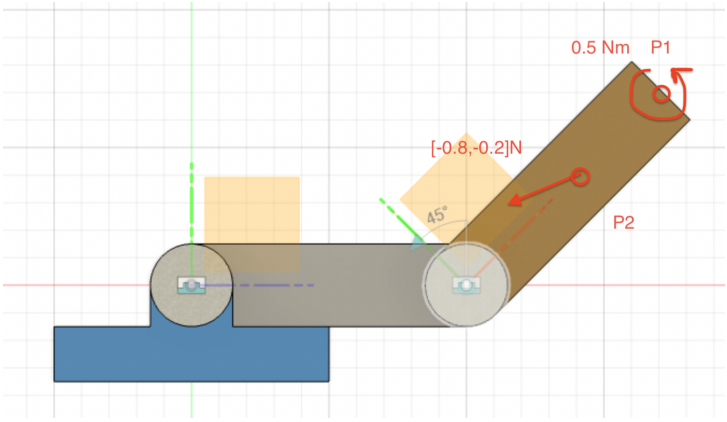
- $\theta_2 = \frac{\pi}{4}$

Forces:

- $[-0.8, -0.2]N$ applied on P2.

Torques:

- $\tau_{ext} = 0.5Nm$ on link 2.




Now we have only one external force acting on P2, while we have a torque acting on the second link, this means we'll have two contributes.

$$\tau_{eq} = -J_{link1}^T S_{p_2/c_2}^T W_1 - J_{link2}^T W_2$$

Where:

$$W_1 = \begin{bmatrix} zeros(3,1) \\ \begin{bmatrix} -0.8 \\ -0.2 \\ 0 \end{bmatrix} \end{bmatrix} \qquad W_2 = \begin{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 500 \end{bmatrix} \\ zeros(3,1) \end{bmatrix}$$

The final $\tau_{eq}$ will be:

$$\tau_{eq} = \begin{bmatrix} 0.8000 \\ -775.771644662754 \end{bmatrix} Nmm$$

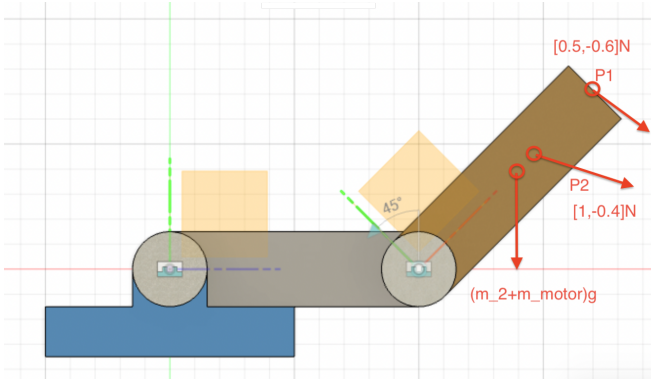
### 4.2.5 Fifth configuration.

Angles:

- $\theta_2 = \frac{\pi}{4}$

Forces:

- $[1, -0.4]N$ applied on P2.
- $[0.5, -0.6]N$ applied on P1.
- $(m_2 + m_{motor})\vec{g}$ applied on C1.




Now I have two external forces plus the gravity force acting on the center of mass of the second joint. All the forces are making the revolute joint turning clockwise, so I am expecting a $\tau_{eq_2} > 0$.

The contributions are the following:

$$\tau_{eq} = -J_{link1}^T S_{p_1/c_2}^T W_{ext,1} - J_{link2}^T S_{p_2/c_2}^T W_{ext,2} - J_{link2}^T W_{grav,2}$$

with:

$$W_{ext,1} = \begin{bmatrix} zeros(3,1) \\ \begin{bmatrix} 0.5 \\ -0.6 \\ 0 \end{bmatrix} \end{bmatrix} \qquad W_{ext_2} = \begin{bmatrix} zeros(3,1) \\ \begin{bmatrix} 1 \\ -0.4 \\ 0 \end{bmatrix} \end{bmatrix}$$

$$W_{grav,2} = \begin{bmatrix} zeros(3,1) \\ (m_2 + m_{motor}) * g \end{bmatrix}$$

The solution is:

$$\tau_{eq} = \begin{bmatrix} -1.5000 \\ 411380.955855363 \end{bmatrix} Nmm$$

# 4. Conclusion.

At the end, the $\tau_{eq}$ I found were what I was expecting. It was a little bit hard to create a matlab script light and tidy, but I think it is pretty easy to read, as it is covered with a lot of comments. The core variables, for example the transformation matrices and the jacobian matrices, are grouped in cells. This was for a better computation of them. The $\tau_{eq}$ are not grouped in cells but are expressed as it shows the following picture: