100/100

Robotics
Assignment 1
Group 1_Fri_G

Nicolas Marco Hahn (501703)

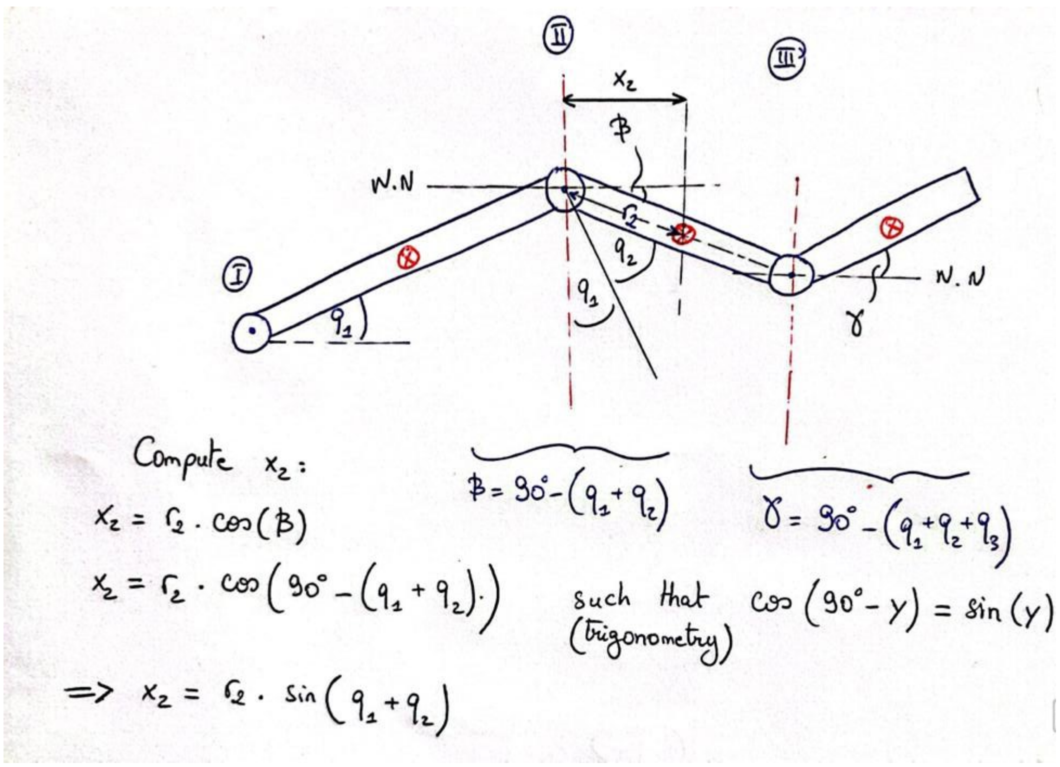
Foued Larbi (412927)

Daniel Papp (401765)

November 5, 2023

1 Calculations

In order to compute the moment arm necessary for the calculation of the torque starting from the center of mass of each joint (red crosses in figure 1), reference axle between the links and trigonometric laws are used. By way of explanation the reference angle q_2 is separated from its predecessor q_1 by an orthogonal axle to the first link. Therefore, the length x_2 can be calculated using the transition from cosine to sine function. This computation process applies also for the third joint and is illustrated in figure 1.



II
III
x_2
β
W.N
r_2
q_2
q_1
I
q_1
W.N
γ
Compute x_2:
$x_2 = r_2 \cdot \cos(\beta)$
$\beta = 90° - (q_1 + q_2)$
$\gamma = 90° - (q_1 + q_2 + q_3)$
$x_2 = r_2 \cdot \cos(90° - (q_1 + q_2).)$
such that (trigonometry) $\cos(90° - y) = \sin(y)$
$\Rightarrow x_2 = r_2 \cdot \sin(q_1 + q_2)$

Figure 1: Sketch showing angle computation

The following abbreviations are used in this documentation:
$c_1 = cos(q_1)$, $s_{12} = sin(q_1 + q_2)$, $s_{123} = sin(q_1 + q_2 + q_3)$

The torques τ_i caused by gravity on joints $i = 1, 2, 3$ are the following:

$$\tau_1 = m_1 \begin{pmatrix} r_1 c_1 \\ 0 \\ 0 \end{pmatrix} g, \quad \tau_2 = m_2 \begin{pmatrix} L_1 c_1 + s_{12} r_2 \\ r_2 s_{12} \\ 0 \end{pmatrix} g$$

and $\tau_3 = m_3 \begin{pmatrix} L_1 c_1 + L_2 s_{12} + s_{123} r_3 \\ s_{12} L_2 + r_3 s_{123} \\ s_{123} r_3 \end{pmatrix} g$

The nature constant g was defined as $-9.81m/s^2$ in the assignment, which is why there is no minus sign in the above equations like in the lecture.

The final gravity vector $G(q_1, q_2, q_3)$ is computed by adding the three torque vectors τ_1, τ_2, τ_3 and it estimates the torque caused by gravity at each joint.

$$G(q_1, q_2, q_3) = \begin{pmatrix} (L_1 c_1 + L_2 s_{12} + r_3 s_{123}) m_3 + (L_1 c_1 + r_2 s_{12}) m_2 + r_1 c_1 m_1 \\ (L_2 s_{12} + r_3 s_{123}) m_3 + r_2 s_{12} m_2 \\ r_3 s_{123} m_3 \end{pmatrix} g$$

2 Implementations

2.1 njmoveControl()

For the implementation of the P controller in $njmoveControl()$ the following control law is applied where it consists of the desired and the current angle positions (qd and q) :

$$\tau = k_p(qd – q)$$

(a) "What kind of behaviors do you observe with different gains?"
Three patterns of behaviour are observed during the simulation. These are depending on the k_p-values:

- For the well tuned k_p values : Since the exerted torque is optimized, a fast and free of overshoot/oscillations response of the robot is observed. However, it is important to acknowledge that generally a slight overshoot and specifically for Joint 1 and 2 an inability to attain the desired angle of 10° are noticed, as this controller relies solely on proportional joint gains and neglect the derivative part (k_v).

- For lower k_p values : A struggling and slow response of the robot is observed. In fact, the exerted torque is insufficient to attain the desired angle position $q_{des} = 10°$.

- For higher k_p values : The robot moves to the desired position quicker, but if k_p is set too high, it will overshoot and a strong oscillating motion will occur. Furthermore, the exerted torques in the joints exceed the limits.

(b): "Why are well tuned gains different for each joint?"
$k_{p1} > k_{p2} > k_{p3}$. The motion structure of the PUMA Robot can be presented as a human arm. By way of explanation the shoulder and the upper arm (joint 1) should be able to move the other parts (the joints 2 and 3) and the lower arm (joint 2) must in turn move the hand (joint 3) in order to reach an object in the space, which means that joint 1 needs to move a larger mass than joint 2 and joint 2 in turn needs to move a larger mass than joint 3. Thus, different magnitudes of forces need to be applied to move each of them.

The robot's behaviour for a step from $q_{init} = [0°\, 0°\, 0°]^T$ to $q_{des} = [10°\, 10°\, 10°]^T$ is documented by plotting the desired angles qd and the actual angles q in fig. 2 and the applied torques τ in fig. 3. The proportional gains k_p were tuned such that:

1. q_{des} is reached as fast as possible
2. there is no oscillation/overshoot in any of the values of q

3. the robot's torque limits are not exceeded

With these goals in mind, the following gains were set for joints $1, 2, 3$: $k_{p1} = 345$, $k_{p2} = 135$ and $k_{p3} = 28$.

Note : The desired angle position $q_{des} = 0.175\text{rad} = 10^\circ$ is mentioned as dq_i for $i = 1, 2, 3$ in all the plots. Please do not mistake it for the angular velocity dq.

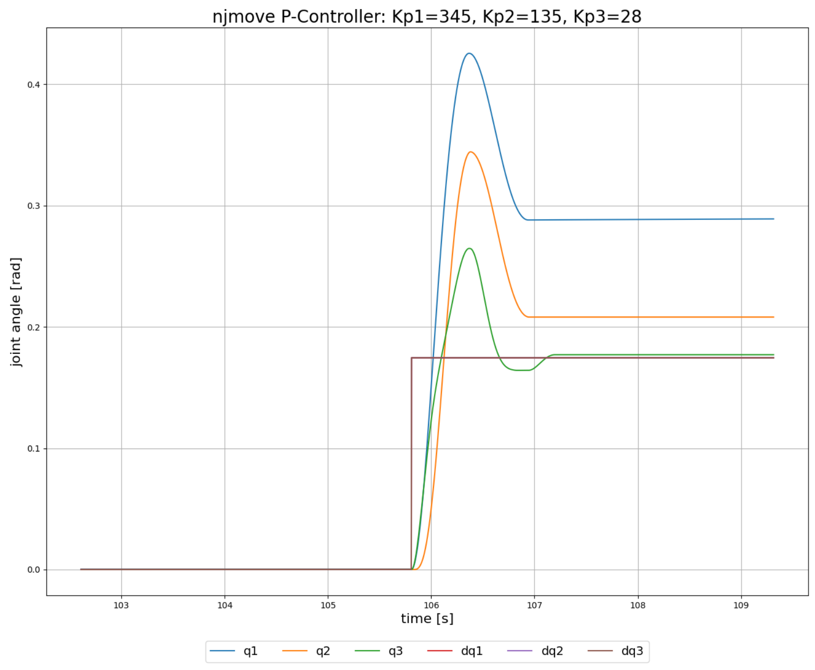


Figure 2: njmove: Joint positions

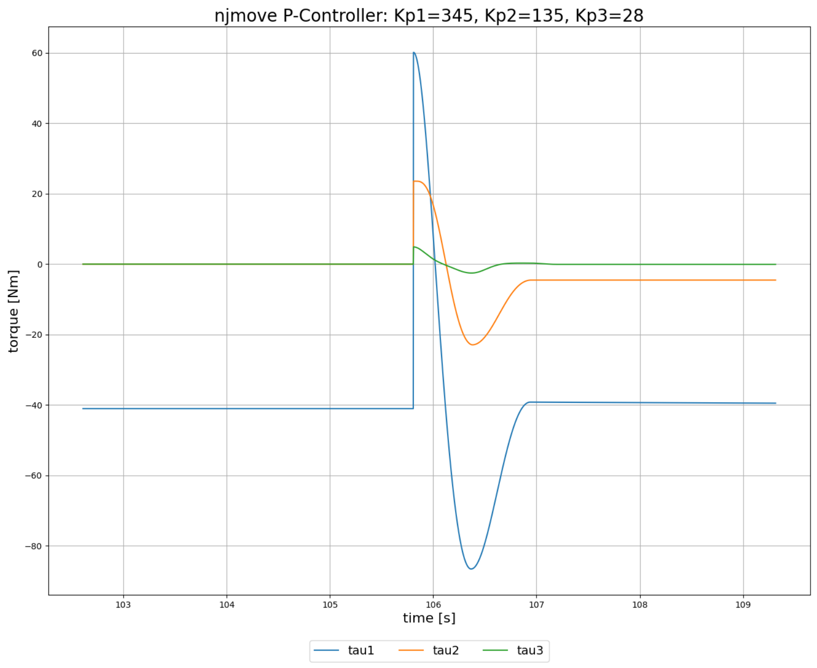


Figure 3: njmove: Joint torques

As one could expect, it is impossible to reach all three goals perfectly. The main focus is not to exceed the torque limits and keep oscillation/overshoot of q to a minimum. For these specific values of k_p q_{des} is only reached in joint 3 in contrast to joint 2 and specifically to joint 1.

2.2 PreprocessControl()

The function derived in section $Calculations$ is stored in the vector G, using the following parameters for the actual geometry of the PUMA robot, which are defined in $param.h$:

- $r_1 = R2,\ r_2 = 0.189783,\ r_3 = R6$
- $L_1 = L2,\ L_2 = L3,\ L_3 = L6$
- $m_1 = M2,\ m_2 = M3 + M4 + M5,\ m_3 = M6$
- $g = -9.81$

2.3 floatControl()

This function is used such that the robot keeps its current pose and only moves when an external force is applied. This is achieved by compensating the effect of gravity on the links, meaning $\tau = G$.

2.4 njgotoControl()

For the implementation of the P-controller with gravity compensation in $njgotoControl()$ the gravity vector G is added up to the proportional control law, resulting in:

$$\tau = G + k_p(qd - q)$$

The response of the PUMA Robot to a $10°$ step in desired angles is visualized in fig. 4 and 5. The gains were set to: $k_{p1} = 330$, $k_{p2} = 120$ and $k_{p3} = 21$.
Note : The desired angle position $q_{des} = 0.175\text{rad} = 10°$ is mentioned as dq_i for $i = 1, 2, 3$ in all the plots. Please do not mistake it for the angular velocity dq.

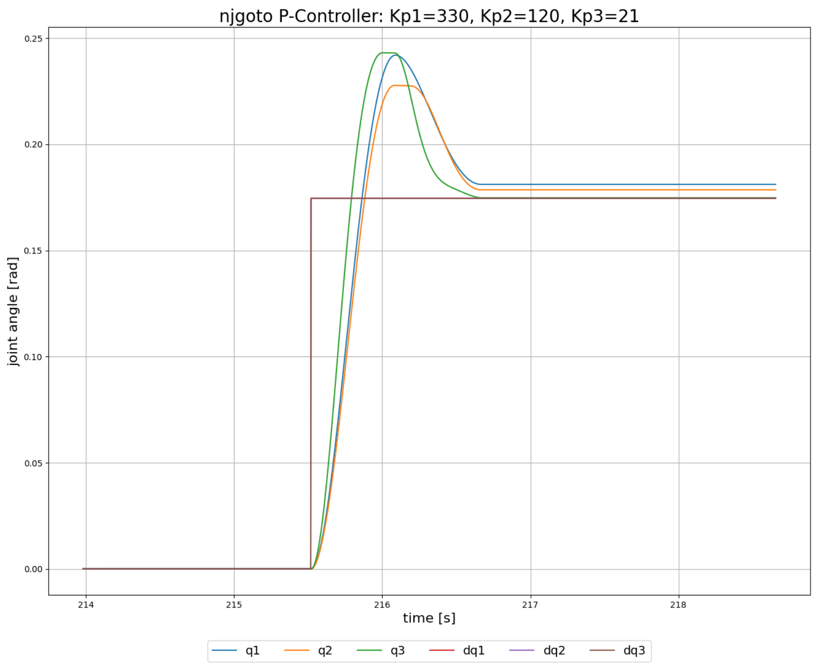


Figure 4: njgoto: Joint positions

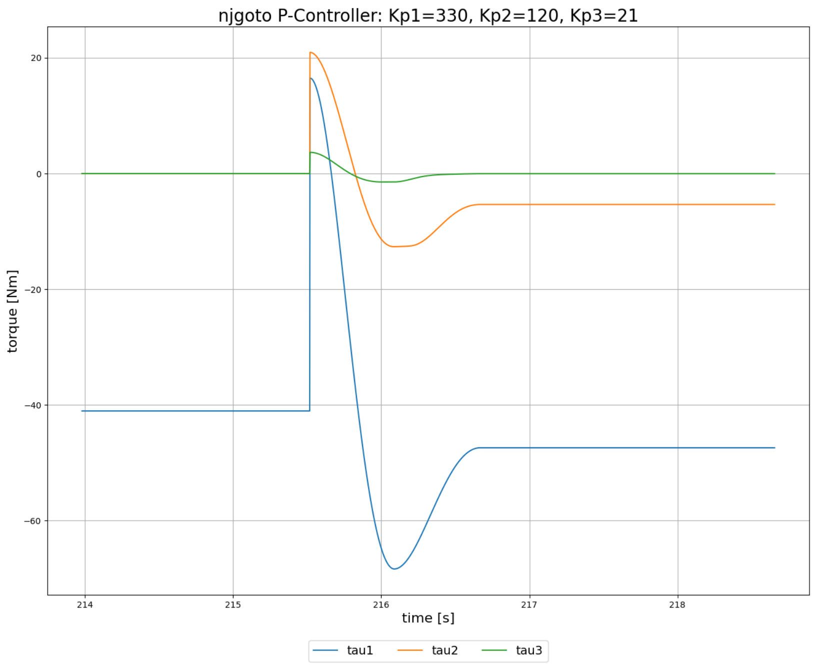


Figure 5: njgoto: Joint torques

In comparison to the P-controller without gravity compensation used in $njmoveControl()$, the joint positions q do not overshoot the target as much and the desired joint positions q_{des} are reached almost perfectly for all joints (fig. 4). Another change can be observed in the torques exerted on all the joints (fig. 5), which are now smaller than without gravity compensation.

2.5 jgotoControl()

For the implementation of the PD-controller with gravity compensation in $jgotoControl()$ a derivative part - which consists of a derivative coefficient k_v and the angular velocity dq - is brought in to the control law stated in section 2.4, resulting in:

$$\tau = G + k_p(qd - q) - k_v dq$$

The robot's response to a 10° step in desired angles using the aforementioned PD-controller is shown in fig. 6 and 7. The proportional and derivative gains were set to the following values: $k_{p1} = 995$, $k_{p2} = 475$, $k_{p3} = 115$, $k_{v1} = 130$, $k_{v2} = 63$ and $k_{v3} = 21$.

Note : The desired angle position $q_{des} = 0.175\text{rad} = 10^\circ$ is mentioned as dq_i for $i = 1, 2, 3$ in all the plots. Please do not mistake it for the angular velocity dq.

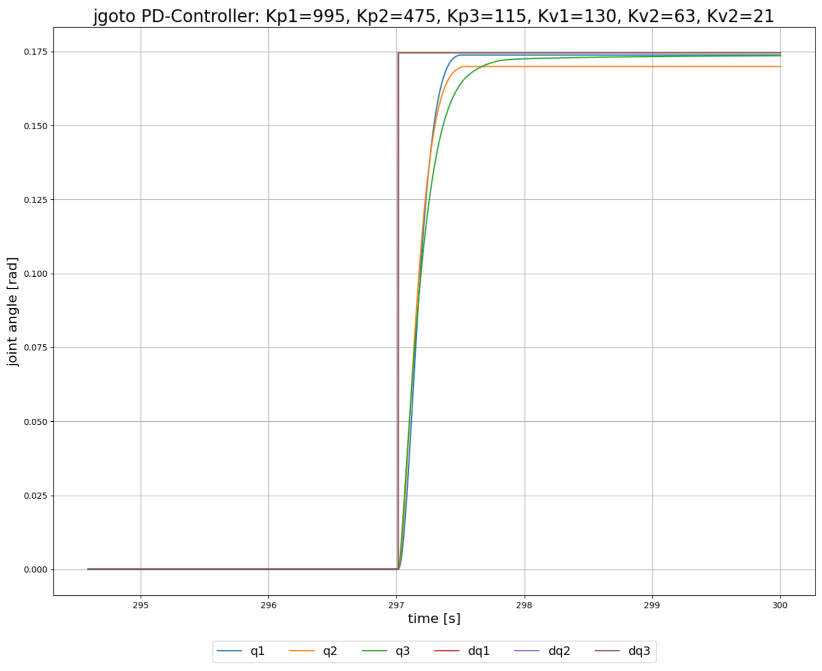


Figure 6: jgoto: Joint positions

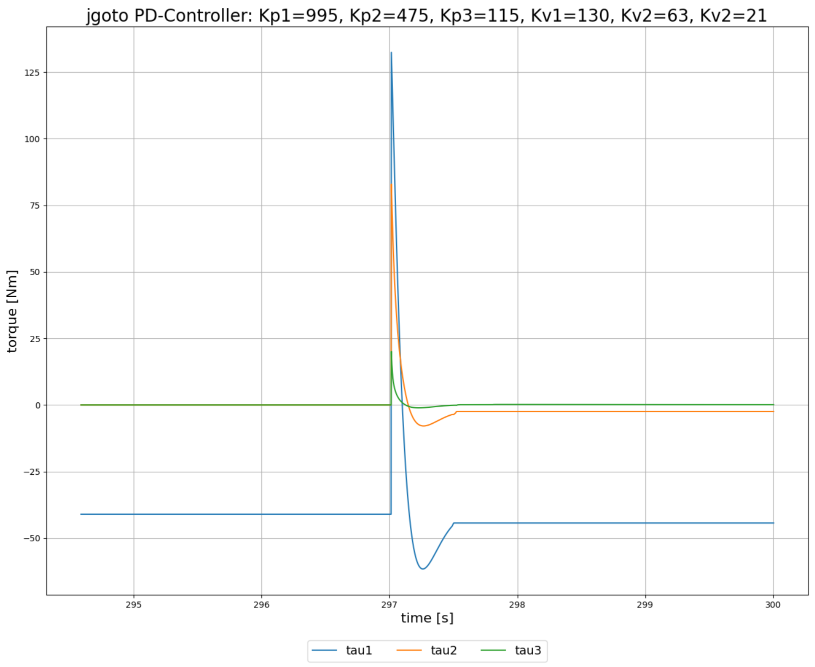


Figure 7: jgoto: Joint torques

As expected, the joint positions q reach the desired angle q_{des} without any oscillation/overshoot. However, the end position of joint 2 is not exactly q_{des}, but a little smaller (fig. 6). The torques exerted on each joint (fig. 7) are higher than those using the P-controller with gravity compensation, because the proportional gains were set to distinctly higher values.

Why can the gains k_p now be higher compared to the P-controller?
The d-part of the controller plays a pivotal role in damping overshoot and oscillations. In fact, it functions as a stabilizer and paves the way for higher k_p values compared to those in the sections 2.1 and 2.4, offering the PUMA robot enhanced responsiveness and accuracy. As a result, the time needed to reach q_{des} could be almost halved e.g the first joint in figures 4 and 6.

Table 1: Contribution

Student Name	B1	B4	B5	B6	B7
Nicolas Marco Hahn		x		x	x
Foued Larbi	x			x	x
Daniel Papp	x	x	x		