96/100

Robotics
Assignment 2
Group 1_Fri_G

Nicolas Marco Hahn (501703)

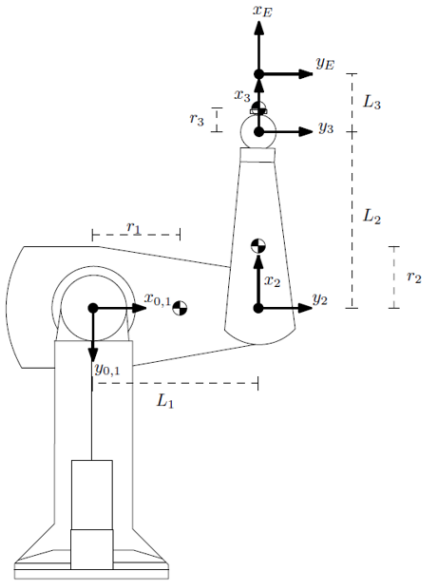
Foued Larbi (412927)

Daniel Papp (401765)

November 26, 2023

1 A Forward Kinematics

In this section we derive the forward kinematics of the 3-DOF RRR Puma depicted in fig. 1.



In the implementation, you should have

Figure 1: RRR Puma in zero configuration

1.1 Find the DH parameters for the 3-DOF RRR Puma

The following table 1 contains the derived DH parameters.

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	q_1
2	0	L_1	0	$q_2 - 90°$
3	0	L_2	0	q_3
4(E)	0	L_3	0	0

Table 1: DH Parameters for the 3-DoF RRR Puma

1.2 Transformation between frames

In this section the forward kinematics ${}^{0}_{E}T(q)$ for the end effector and the homogenous transformations between adjacent links ${}^{i-1}_{i}T(q)$ for this manipulator using the DH-parameter stated in table 1 are computed.
To get the homogeneous transformation between link 0 and the end-effector (${}^{0}_{E}T(q)$), we simply multiply the transformations from link 0 to 1, from 1 to 2 and so on. This leads to the following equation:

$$ {}^{0}_{E}T(q) = {}^{0}_{1}T(q) \cdot {}^{1}_{2}T(q) \cdot {}^{2}_{3}T(q) \cdot {}^{3}_{E}T(q) \tag{1} $$

As seen in the tutorial the DH parameters can be used to compute the the transition matrices with the following formula presented in figure 2.

$$ \begin{aligned} {}^{i-1}_{i}T &= R_X(\alpha_{i-1})\ D_X(a_{i-1})\ R_Z(\theta_i)\ D_Z(d_i) \\ &= \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1} d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} $$

Figure 2: Homogeneous transform matrix with DH parameters

The transformation ${}^{0}_{1}T(q)$ uses the DH parameters with $i = 1$, which leads to the following matrix:

$$ {}^{0}_{1}T(q) = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

The translation's vector in that matrix equals 0 in all entries, as the origin of the coordinate system is placed exactly in link 1, thus the two coordinate systems only differ in their rotation around the z-axis when $\theta_1 > 0$, but never in their translation's part.

The transformation between joint 1 and 2 uses the DH parameters with $i = 2$, which leads

to the following matrix:

$$
{}^{1}_{2}T(q) = \begin{bmatrix} s_2 & c_2 & 0 & L_1 \\ -c_2 & s_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

The transformation between joint 2 and 3 uses the DH parameters with $i = 3$, which leads to the following matrix:

$$
{}^{2}_{3}T(q) = \begin{bmatrix} c_3 & -s_3 & 0 & L_2 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

As expected, the translational part only differs in the x-value by L_2 and the rotation around the z-axis depends on θ_3.

The transformation between joint 3 and the end effector is built as follows using the DH parameters with $i = 4$:

$$
{}^{3}_{E}T(q) = \begin{bmatrix} 1 & 0 & 0 & L_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

The forward kinematics ${}^{0}_{E}T(q)$ for the end effector is computed using the equation 1.

$${}^{0}_{E}T(q) = \begin{bmatrix} c_3s_{12} + s_3c_{12} & -s_3s_{12} + c_3c_{12} & 0 & L_3(c_3s_{12} + s_3c_{12}) + L_2(c_1s_2 + s_1c_2) + L_1c_1 \\ -c_3c_{12} + s_3s_{12} & s_3c_{12} + c_3s_{12} & 0 & L_3(s_3s_{12} - c_3c_{12}) - L_2c_{12} + L_1s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Using trigonometric identities $s_{12} = s_1c_2 + c_1s_2$ and $c_{12} = c_1c_2 - s_1s_2$ the forward kinematics ${}^{0}_{E}T(q)$ can be simplified to:

$${}^{0}_{E}T(q) = \begin{bmatrix} s_{123} & c_{123} & 0 & L_3s_{123} + L_2s_{12} + L_1c_1 \\ -c_{123} & s_{123} & 0 & -L_3c_{123} - L_2c_{12} + L_1s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

1.3 End effector position in Operational space

As mentioned in the lecture, the homogeneous transform matrix ${}^{0}_{E}T(q)$ is formed by translational vector $t = (t_x, t_y, t_z)^T$ and rotational matrix $R(\theta)$:

$${}^{0}_{E}T(q) = \begin{bmatrix} & & & t_x \\ & R(\theta) & & t_y \\ & & & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Since the angle α is computed by adding up all the joint angles depicted in table 1 and the z-component is neglected, the coordinates of the end-effector in the base frame can be described as follows:

$$F(q) = \begin{pmatrix} t_x \\ t_y \\ \alpha \end{pmatrix} = \begin{pmatrix} L_3 s_{123} + L_2 s_{12} + L_1 c_1 \\ -L_3 c_{123} - L_2 c_{12} + L_1 s_1 \\ q_1 + q_2 - \frac{\pi}{2} + q_3 \end{pmatrix}$$

1.4 Compute the End Effector Jacobian

The Jacobian for the end effector is computed using the equation $J(q) = \frac{\partial F(q)}{\partial q}$. As we have 3 rows in $F(q)$ and 3 joint angles, the Jacobian is a 3x3 matrix of the following form:

$$J(q) = \begin{bmatrix} \frac{\partial F_1}{\partial q_1} & \frac{\partial F_1}{\partial q_2} & \frac{\partial F_1}{\partial q_3} \\ \frac{\partial F_2}{\partial q_1} & \frac{\partial F_2}{\partial q_2} & \frac{\partial F_2}{\partial q_3} \\ \frac{\partial F_3}{\partial q_1} & \frac{\partial F_3}{\partial q_2} & \frac{\partial F_3}{\partial q_3} \end{bmatrix}$$

Computing all the necessary derivatives yields the following Jacobian for the end effector:

$$J(q) = \begin{bmatrix} L_3 c_{123} + L_2 c_{12} - L_1 s_1 & L_3 c_{123} + L_2 c_{12} & L_3 c_{123} \\ L_3 s_{123} + L_2 s_{12} + L_1 c_1 & L_3 s_{123} + L_2 s_{12} & L_3 s_{123} \\ 1 & 1 & 1 \end{bmatrix}$$

1.5 Understanding the Jacobian Matrix and Pose Singularities

The following figure 3 depicts the sketched translational velocity vectors in three different configurations.

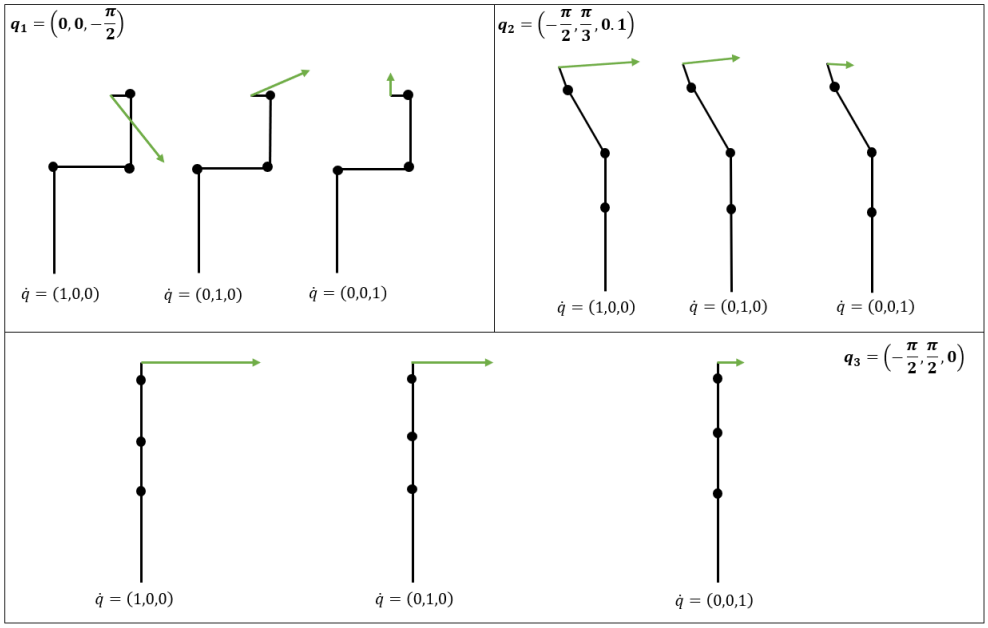


Figure 3: Sketched translational velocity vectors in different configurations

- $q_1 = (0, 0, -\frac{\pi}{2})$: The robot is fully controllable in all directions, as the arrows differ in their x and y direction.
- $q_2 = (-\frac{\pi}{2}, \frac{\pi}{3}, 0.1)$: The robot is close to a singularity, as the arrows almost point into the same direction.
- $q_3 = (-\frac{\pi}{2}, \frac{\pi}{2}, 0)$: The robot is in a singularity, as all arrows point into the x direction (world coordinate system), thus it cannot move in y direction meaning it loses one degree of freedom.

2 B Trajectory Generation in Joint Space

2.1 Generation of smooth trajectories with polynomial splines

a) Compute the cubic spline parameters:

We want to move the robot from zero configuration $q_a = (0, 0, 0)^T$ to the end joint configuration $q_c = (-\frac{\pi}{2}, \frac{\pi}{4}, 0)^T$ within $t_f = 5s$. After half the time ($t_{via} = 2.5\text{s}$), the robot should pass the intermediate point $q_b = q_{via} = (-\frac{\pi}{4}, \frac{\pi}{2}, 0)^T$.

This is done by concatenating the two following cubic splines, which in combination with

initial conditions yield the needed spline parameters:

- Spline from a to via:
  $u_a(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$
  Initial conditions:
  - $u_a(0) = q_a$
  - $\dot{u}_a(0) = \dot{q}_a = (0,0,0)^T$
  - $u_a(t_{via}) = q_{via}$
  - $\dot{u}_a(t_{via}) = \dot{q}_{via}$
- Spline from via to b:
  $u_b(t) = b_0 + b_1(t - t_{via}) + b_2(t - t_{via})^2 + b_3(t - t_{via})^3$
  Initial conditions:
  - $u_b(t_{via}) = q_{via}$
  - $\dot{u}_b(t_{via}) = \dot{q}_{via}$
  - $u_b(t_f) = q_c$
  - $\dot{u}_b(t_f) = \dot{q}_c = (0,0,0)^T$

These splines and their initial conditions lead to the following equations for computing the spline parameters listed in table 2:

- $a_0 = q_a$
- $a_1 = \dot{q}_a$
- $a_2 = \frac{3}{t_{via}^2}(q_{via} - q_a) - \frac{2}{t_{via}}\dot{q}_a - \frac{1}{t_{via}}\dot{q}_{via}$
- $a_3 = -\frac{2}{t_{via}^3}(q_{via} - q_a) + \frac{1}{t_{via}^2}(\dot{q}_a + \dot{q}_{via})$
- $b_0 = q_{via}$
- $b_1 = \dot{q}_{via}$
- $b_2 = \frac{3}{(t_f - t_{via})^2}(q_c - q_{via}) - \frac{2}{t_f - t_{via}}\dot{q}_{via} - \frac{1}{t_f - t_{via}}\dot{q}_c$
- $b_3 = -\frac{2}{(t_f - t_{via})^3}(q_c - q_{via}) + \frac{1}{(t_f - t_{via})^2}(\dot{q}_{via} + \dot{q}_c)$

The end effector velocity at the via point is not defined. Therefore, we apply a heuristic to choose joint velocities: If direction of velocity changes for a joint, its desired velocity at the via point is set to 0. Otherwise we set it to the average velocity of the two adjacent

segments.

As joint 2 changes its direction of velocity, its desired velocity at the via point is set to 0. Joint 3 does not move at all, so its velocity is also 0. The desired velocity at the via point of joint 1 is computed as follows: $\dot{q}_{via,0} = \frac{1}{2} \cdot (\frac{q_{via} - q_a}{t_{via}} + \frac{q_c - q_{via}}{t_{via}})$. This leads to the velocity vector at the via point $\dot{q}_{via} = (-\frac{\pi}{10}, 0, 0)^T$.

	q_1	q_2	q_3
a_0	0	0	0
a_1	0	0	0
a_2	-0.25	0.75	0
a_3	0.05	-0.2	0
b_0	-0.79	1.57	0
b_1	-0.31	0	0
b_2	-0.13	-0.38	0
b_3	0.05	0.1	0

Table 2: Spline parameters for each joint

b) Plot a diagram of the joint angles with respect to time:

The following figures 4-6 illustrate the joint angles with respect to time for each joint.

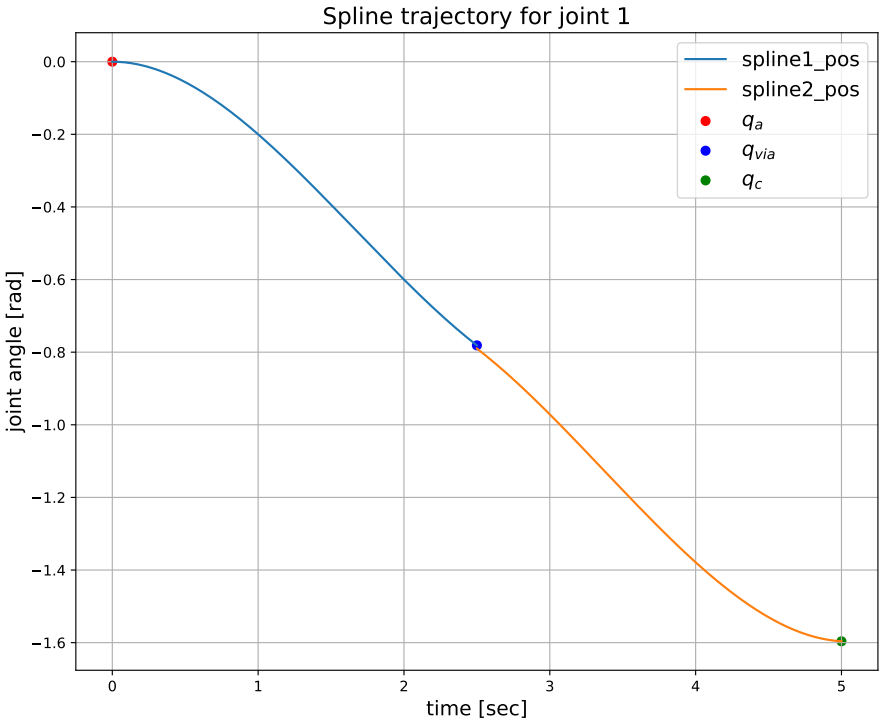


Figure 4: Spline trajectory for joint 1

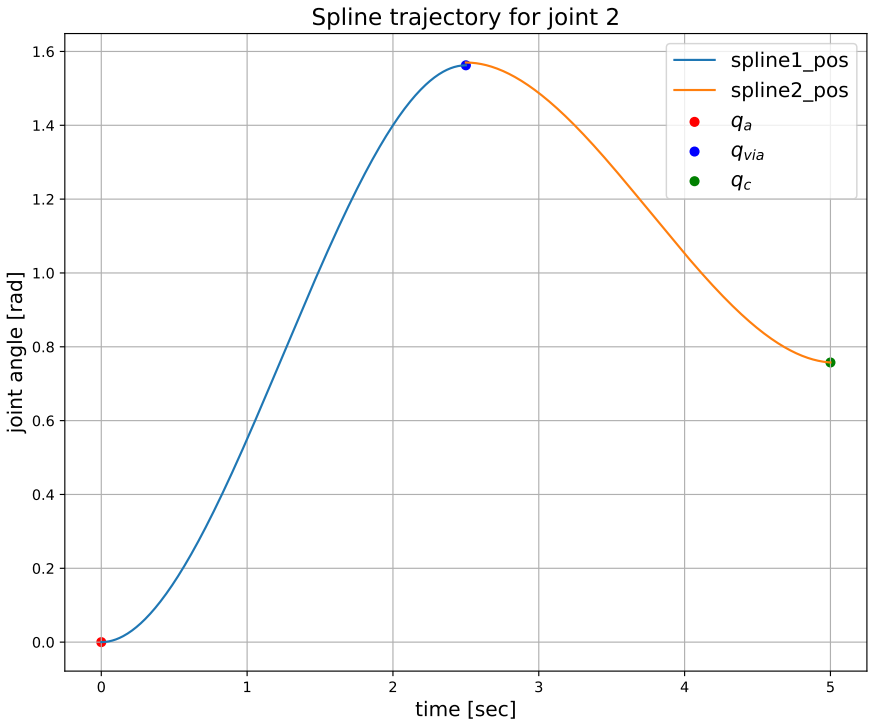


Figure 5: Spline trajectory for joint 2

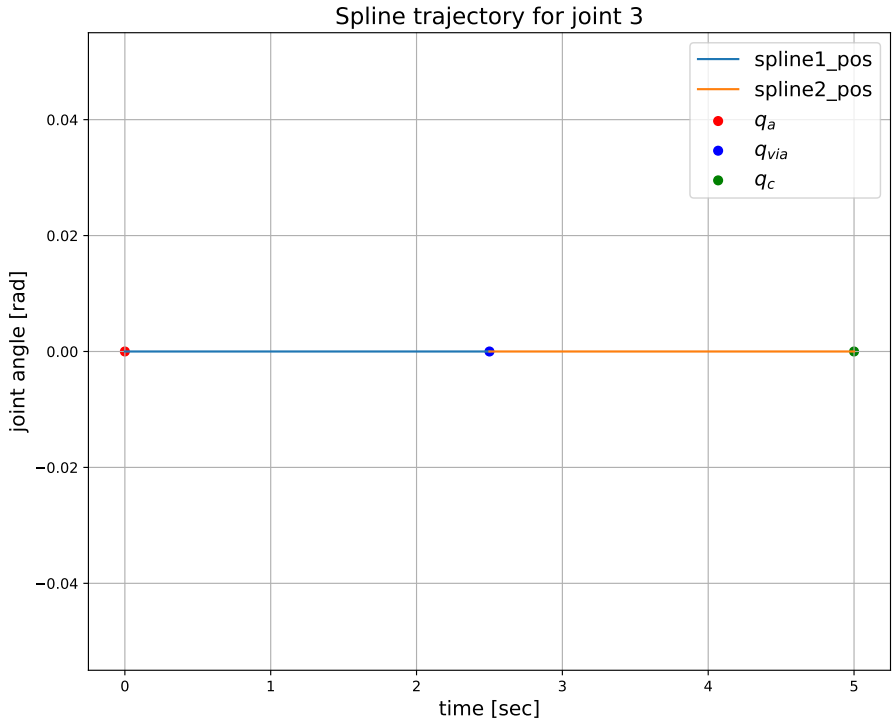


Figure 6: Spline trajectory for joint 3

2.2 Trajectories in joint space

a) Trajectory Generation

As mentioned in the tutorials, the desired joint position, velocity and acceleration cane be expressed as follows:

$q_i(t) = a_{0,i} + a_{1,i}t + a_{2,i}t^2 + a_{3,i}t^3$

$\dot{q}_i(t) = a_{1,i} + 2a_{2,i}t + 3a_{3,i}t^2$

$\ddot{q}_i(t) = 2a_{2,i} + 6a_{3,i}t$

$i = 0, 1, 2$

Given that $\dot{q}_0 = \dot{q}_f = 0$ the spline parameters $a_0 = a_1 = 0$, a_2 and a_3 are computed as follows:

$$a_2 = 3\frac{(q_f - q_0)}{(t_f - t_0)^2} - \frac{(2\dot{q}_0 + \dot{q}_f)}{(t_f - t_0)} = 3\frac{(q_f - q_0)}{(t_f - t_0)^2}$$

$$a_3 = -2\frac{(q_f - q_0)}{(t_f - t_0)^3} + \frac{(\dot{q}_0 + \dot{q}_f)}{(t_f - t_0)^2} = -2\frac{(q_f - q_0)}{(t_f - t_0)^3}$$

The movement of the robot arm is limited by:

Implementation error: proj1 doesn't start.-2 points.

- max. joint velocity: $\dot{q}_{max}$
- max. joint acceleration: $\ddot{q}_{max}$

It is to note that the trajectory reaches its maximum velocity at $t = \frac{t_f}{2}$.

Case 1 (observing velocity)
$\dot{q}_i(\frac{t_f}{2}) = a_{1,i} + 2a_{2,i}(\frac{t_f}{2}) + 3a_{3,i}(\frac{t_f}{2})^2 = \dot{q}_{max}$
resulting in:

$$t_f = \frac{3}{2}\left|\frac{(q_f - q_0)}{\dot{q}_{max}}\right|$$

As a result, the trajectory reaches its maximum acceleration at the beginning $t = 0$, decreases overtime until it turns into a deceleration.

Case 2 (observing acceleration)

$\ddot{q}_i(0) = 2a_{2,i} = \ddot{q}_{max}$
resulting in:

$$t_f = \sqrt{\left|6\frac{(q_f - q_0)}{\ddot{q}_{max}}\right|}$$

In the $computeTf$ function we can now calculate all possible solutions for t_f and choose the highest value, to be then applied for further spline calculations.

b) Control

Afterward we can use our PD-controller to calculate the torque vector taking the gravity vector into account:

$\tau_i = G(q_i) - [k_{p,i}(q_i - q_{des,i}) + k_{v,i}(\dot{q}_i - \dot{q}_{des,i})]$
$i = 0, 1, 2$

c) Trajectory to a specific point

For the simulation the following gain values were used : $k_{p1} = 750$, $k_{v1} = 300$, $k_{p2} = 450$, $k_{v2} = 150$, $k_{p3} = 250$, $k_{p1} = 50$
As specified in the task guidelines, the desired joint angles were established as follows and set in the $njtrackControl$ function:
$q_{des} = (0.096rad, 0.967rad, -1.061rad)^T = (5.5°, 55.4°, -60.8°)^T$

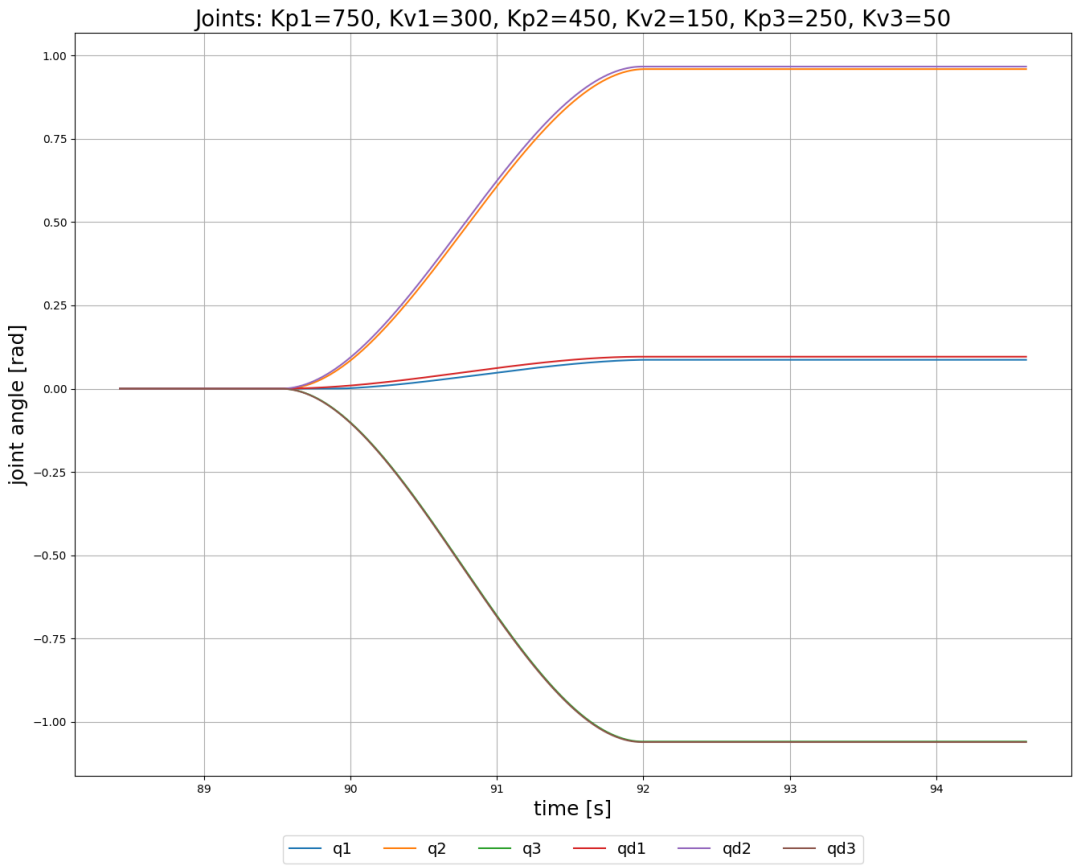


Figure 7: Cubic spline trajectory from zero configuration to q_{des}

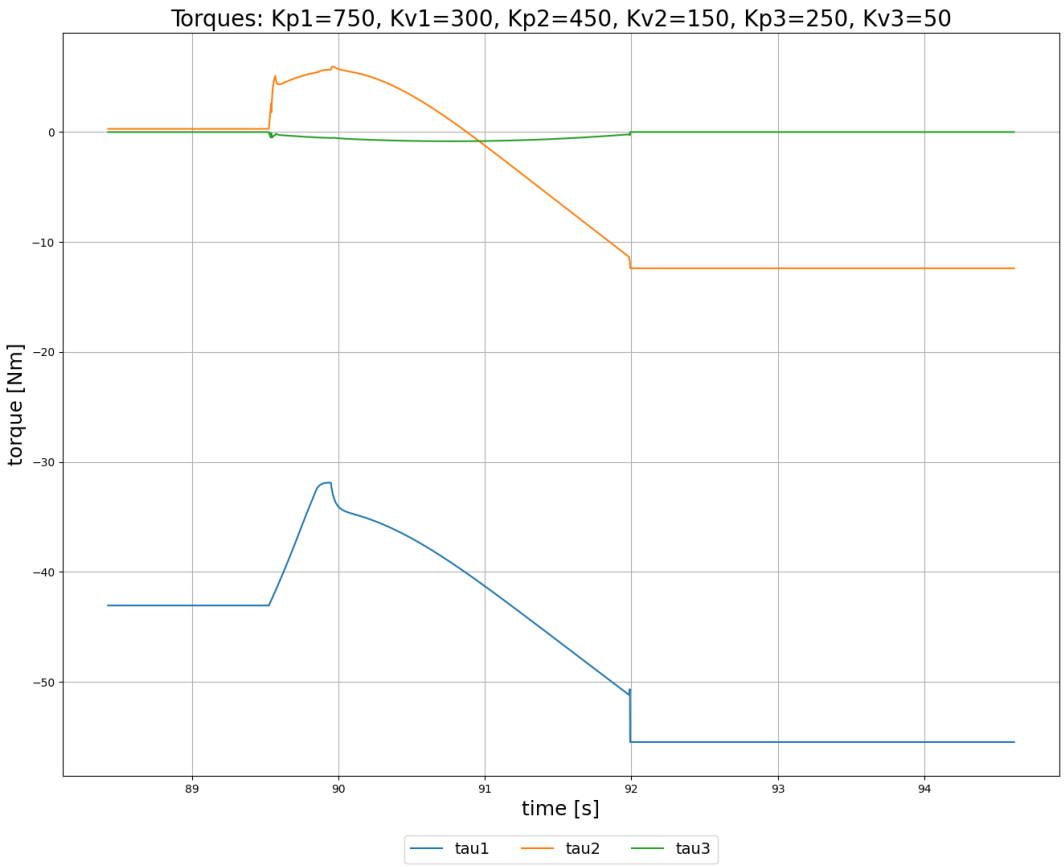


Figure 8: torque angles with controllers

As illustrated in the plots, the exerted torques remain under the limit and the desired joint angles are almost reached with a small error. It is to highlight the accuracy and the responsiveness of the Puma 560.

3 C Operational Space Control

Circle goes in the wrong way! But it is OK.

3.1 Project 2 - Circle

a) Motion Generation

Given the radius r, the angular velocity $\dot{\beta}$, the coordinates of the circle center x_{center} and at

t_0 which are:

x_0	x_{center}	r	$\dot{\beta}$
$(0.8\,m, 0.35\,m)$	$(0.6\,m, 0.35\,m)$	$0.2\,m$	$\frac{2\pi}{5s}$

Table 3: Motion trajectory Parameters

We can generate a circular motion in the operational space by combining the sine and cosine function. The end effector orientation should stay vertical overtime, therefore $x_{d,3} = \dot{x}_{d,3} = 0$

The desired coordinates of the end effector position are computed as follows:

$$x_{des,1}(t) = x_{center} + r\cos\Big(\dot{\beta}t\Big)$$
$$x_{des,2}(t) = y_{center} - r\sin\Big(\dot{\beta}t\Big)$$

By deriving them we can calculate the desired velocity:

$$\dot{x}_{des,1}(t) = -\dot{\beta}r\sin\Big(\dot{\beta}t\Big)$$

$$\dot{x}_{des,2}(t) = -\dot{\beta}r\cos\Big(\dot{\beta}t\Big)$$

b) Control

In order to track the position of the end effector, the control law:

$$F = -k_p(x - x_{des}) - k_v(\dot{x} - \dot{x}_{des})$$

is introduced to the Gravity vector $G(q)$ from Assignment 1 with the help of the transposed Jacobian and Inertia matrix K - which similarly to the lecture was chosen to be $K = 1$ -, the following equation is implemented in the $proj2Control$ function.

$$\tau = G(q) + KJ^TF$$

c) Gain Tuning and Plots

After adjusting k_p and k_v values so that the trajectory is tracked well and the joint torques do not exceed the limits, we set the following values for our PD-Controller: $k_{p1} = 4000, k_{p2} =$

$2000, k_{p3} = 500, k_{v1} = 500, k_{v2} = 250, k_{v3} = 150.$

Figure 9 shows the torques τ that are exerted on each joint during the cycle motion. You can see that the torque limits are not exceeded.

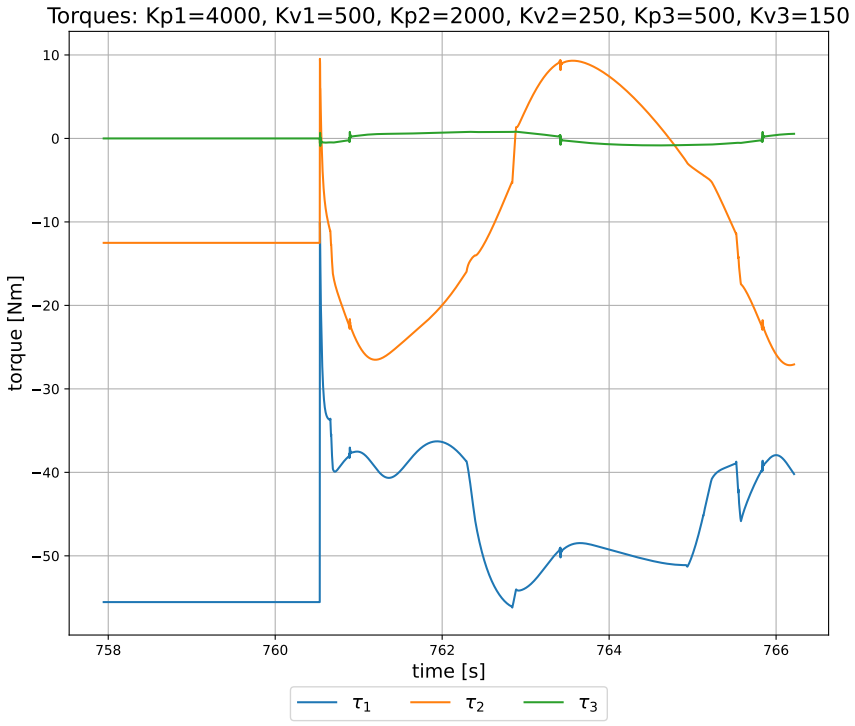


Figure 9: Torques exerted on each joint during one cycle motion

Figure 10 depicts the joint angles q during the cycle motion.

Figure 11 illustrates the desired (x_d) and actual positions (x) during the cycle motion.

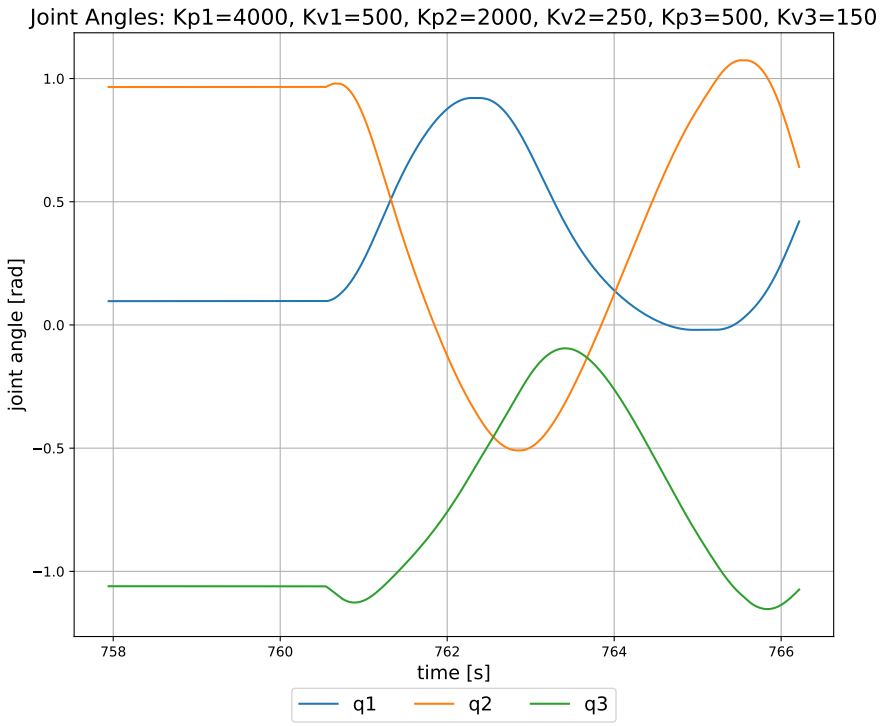


Figure 10: Joint angles during one cycle motion

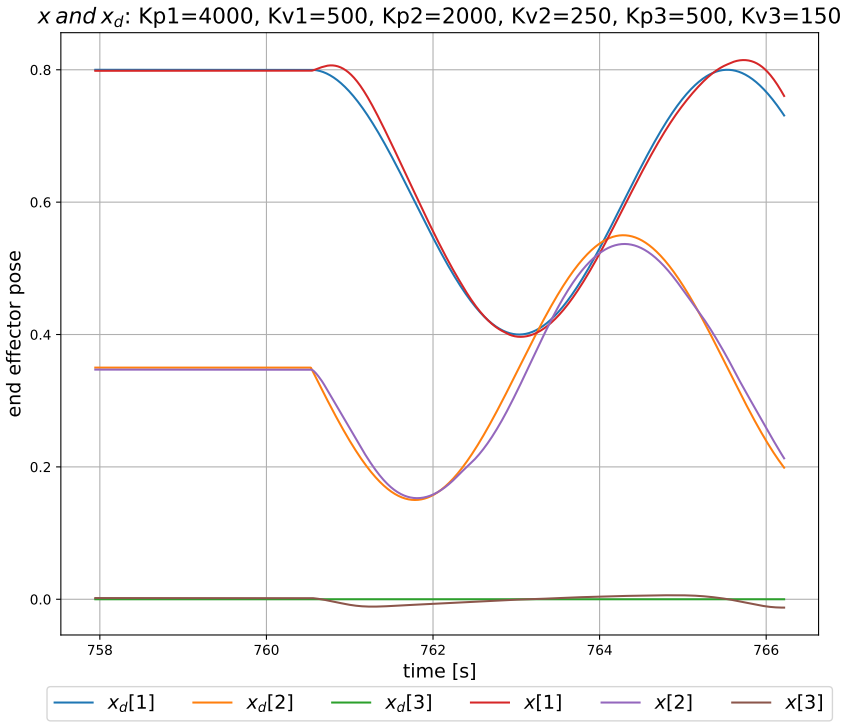


Figure 11: Desired and actual positions [m] during one cycle motion

Figure 12 outlines the position error $e = x - x_{des}$ during the cycle motion.

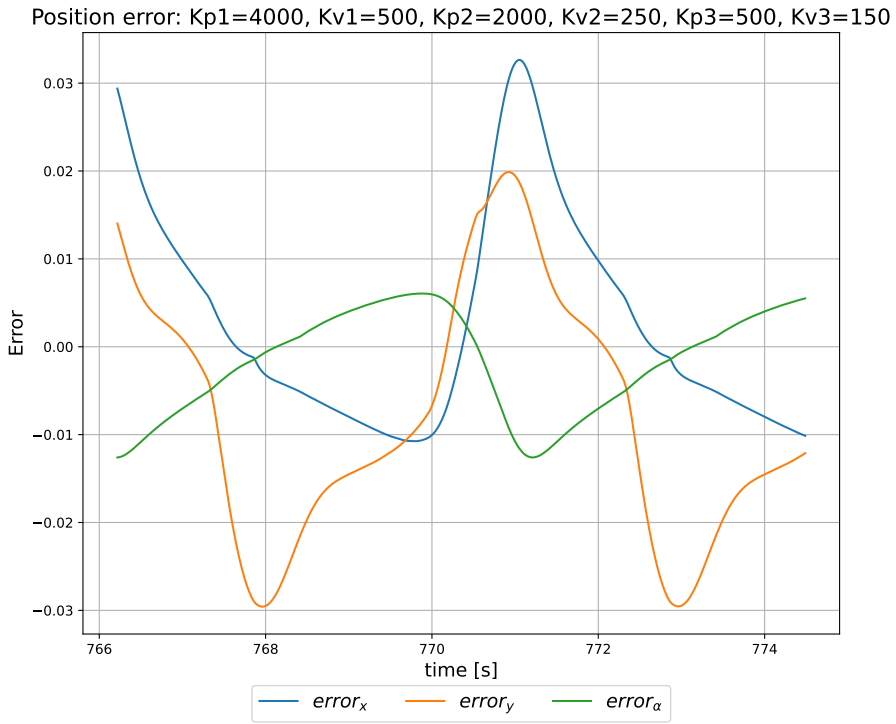


Figure 12: Position errors [m] during one cycle motion

How does the graph for the error change for different values of kp?:
The error gets smaller for high k_p values and higher for low k_p values.
This is due to Newton's first law of motion which states that every object will remain at rest or in uniform motion in a straight line unless compelled to change its state by the action of an external force. This tendency to resist changes in a state of motion is inertia [1]. The external force that makes our end effector perform a circular motion is called centripetal force and it needs to be high enough (by choosing a high k_p) to overcome the inertia and thus make the end effector move in a circle instead of a ragged ellipsis which happens when k_p is set too low (fig.14).

Figure 13 shows the desired (x_d) and actual positions (x) in the x-y-plane.

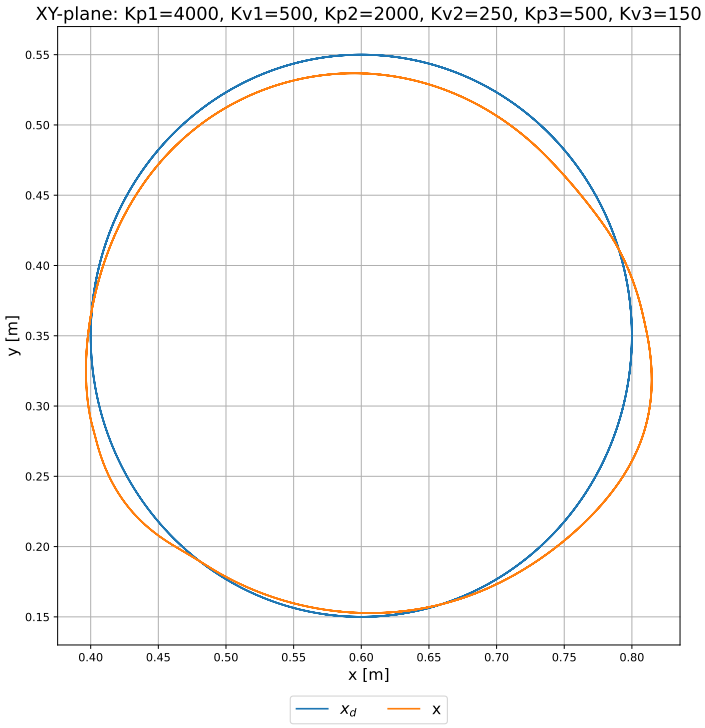


Figure 13: Desired and actual positions in xy-plane

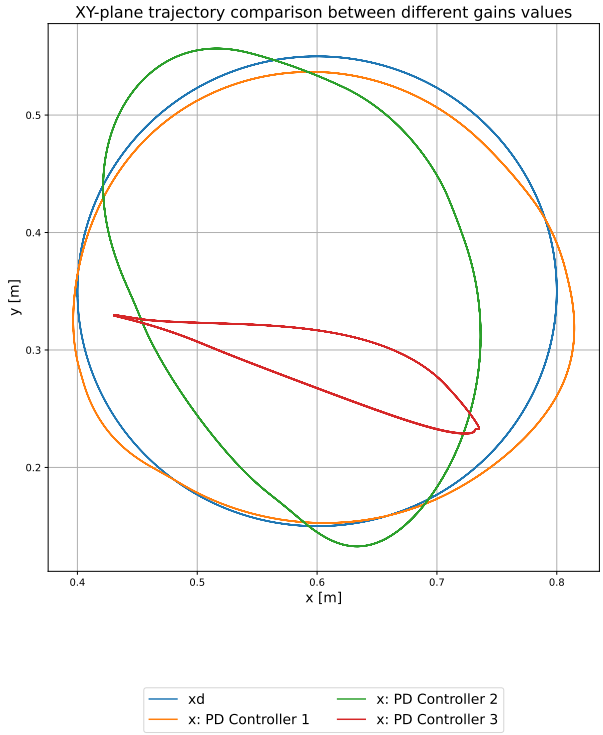


Figure 14: Desired and actual positions in xy-plane using different gains values

The PD Controllers $1, 2, 3$ mentioned in figure 14 have the following gains values:

- PD Controller 1 (our final gains): $k_{p1} = 4000, k_{v1} = 500, k_{p2} = 2000, k_{v2} = 250, k_{p3} = 500, k_{v3} = 150$
- PD Controller 2: $k_{p1} = 750, k_{v1} = 200, k_{p2} = 400, k_{v2} = 100, k_{p3} = 100, k_{v3} = 50$
- PD Controller 3: $k_{p1} = 300, k_{v1} = 50, k_{p2} = 100, k_{v2} = 25, k_{p3} = 50, k_{v3} = 10$

3.2 Project 3 - Parabolic Blends

It is to acknowledge that cubic splines have the negative aspect of discrete velocity jumps when starting and stopping. In this spirit, we resort to parabolic blends for creating the trajectory consisting of a linear segment (t_h) and two blend regions (t_b) with the help of the constraint $|\ddot{\beta}| \leq \frac{2\pi}{25s^2}$. Moreover, the end-effector orientation remains unchanged overtime, therefore $x_{des,3} = \dot{x}_{des,3} = 0$.
As seen in the tutorial, we use the following equations to compute the trajectory:

$$\begin{aligned} \beta &= \frac{1}{2}\ddot{\beta}t^2 + \beta_0 \\ \beta(t_b) &= \frac{1}{2}\dot{\beta}t_b + \beta_0 \quad \text{given that} \\ \dot{\beta} &= t_b \cdot \ddot{\beta}, \quad \beta_0 = 0rad, \quad \beta_f = 6\pi \end{aligned}$$

Constant acceleration:
The blend time t_{b1} is computed as follows:

$$t_{b1} = \frac{\dot{\beta}}{\ddot{\beta}} = \frac{2\pi}{5s} \cdot \frac{25s^2}{2\pi} = 5s$$

The angle β at transition t_{b1} is given by:

$$\beta(t_{b1}) = \frac{1}{2}\left(\frac{2\pi}{25s^2}\right)(t_{b1})^2 = \pi$$

For the following calculations the parameters in table 3 are retained:

$$\mathbf{x}_{des} = \mathbf{x}_{center} + \begin{bmatrix} r \cdot \cos(\beta) \\ -r \cdot \sin(\beta) \end{bmatrix}$$

$$\dot{\mathbf{x}}_{des} = -\frac{2\pi}{25s^2} \cdot \begin{bmatrix} r \cdot \sin(\beta) \cdot t \\ r \cdot \cos(\beta) \cdot t \end{bmatrix}$$

Linear segment:

Since the trajectory generation stops after 3 full circles (6π) and the angle of both blend regions equals to 2π, the linear angle should be $\theta_{rotated} = 6\pi - 2\pi = 4\pi$.

$$t_h = \frac{\theta_{rotated}}{\dot{\beta}} = 4\pi \cdot \frac{5s}{2\pi} = 10s$$

The desired angle β in this segment is computed with the following equation, since it is a simple linear segment:

$$\begin{aligned} \beta(t) &= \dot{\beta}(t_{b1})(t - (t_0 + t_{b1})) \\ &= \frac{\beta(t_h) - \beta(t_{b1})}{t_h - t_{b1}}(t - (t_0 + t_{b1})) \\ &= \frac{2\pi}{5s}(t - (t_0 + t_{b1})) \end{aligned}$$

With the help of the trigonometric identities $\cos(\beta + \pi) = -\cos(\beta)$ and $\sin(\beta + \pi) = -\sin(\beta)$:

$$\mathbf{x}_{des} = \mathbf{x}_{center} + \begin{bmatrix} -r \cdot \cos(\beta) \\ r \cdot \sin(\beta) \end{bmatrix}$$

$$\dot{\mathbf{x}}_{des} = \frac{2\pi}{5s} \cdot \begin{bmatrix} r \cdot \sin(\beta) \\ r \cdot \cos(\beta) \end{bmatrix}$$

Constant deceleration:

$$t_{b2} = t_{b1} = 5s$$

The angle β at the start of the deceleration phase, is equal to $\beta(15s) = \frac{1}{2}\dot{\beta}(t_{b2}) \cdot 15\text{s} = 5\pi$. In the deceleration segment β is computed using the following formula:

$$\beta(t) = \pi - \frac{1}{2} \cdot \frac{2\pi}{25s^2}(t - (t_{b2} + t_0))^2$$

$$\beta(t) + \pi = \frac{1}{2} \cdot \frac{2\pi}{25s^2}(t - (t_{b2} + t_0))^2 = \beta_{compute}$$

$$\mathbf{x}_{des} = \mathbf{x}_{center} + \begin{bmatrix} r \cdot \cos(\beta_{compute}) \\ -r \cdot \sin(\beta_{compute}) \end{bmatrix}$$

$$\dot{\mathbf{x}}_{des} = \frac{2\pi}{25s^2} \cdot \begin{bmatrix} -r \cdot (t - (t_0 + t_{b2})) \cdot \sin(\beta_{compute}) \\ r \cdot (t - (t_0 + t_{b2})) \cdot \cos(\beta_{compute}) \end{bmatrix}$$

Analogously to section 3.1, the following equation is implemented in the $proj3Control$ function in order to track well the trajectory:

$$\tau = G(q) + KJ^T F$$

The desired angle β and desired angular velocity $\dot{\beta}$ are depicted respectively in figure 15 and 16.

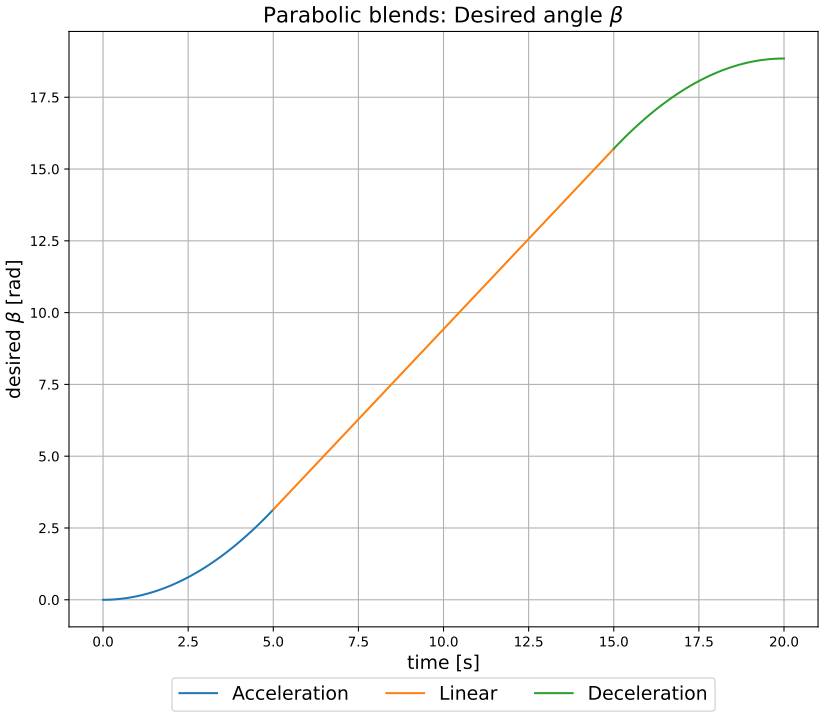


Figure 15: Desired angle β

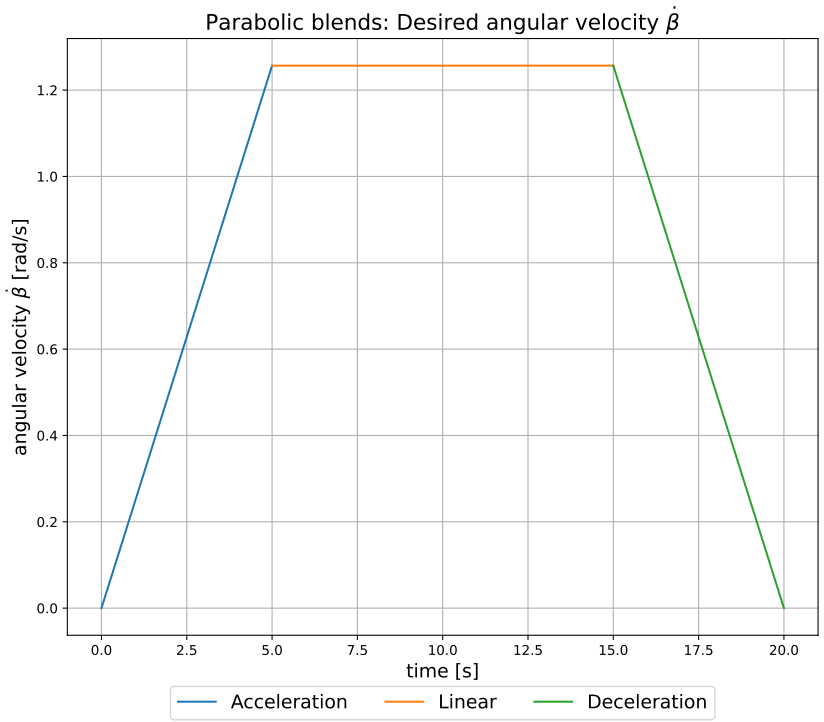


Figure 16: Desired angular velocity $\dot{\beta}$

Table 4: Contribution

Student Name	A2	A3	A4	B2	C1	C2	C3	C4
Nicolas Marco Hahn	x		x		x			x
Foued Larbi		x		x		x		x
Daniel Papp	x		x		x		x	

References

[1] NASA What are Newton's Laws of Motion? https://www1.grc.nasa.gov/beginners-guide-to-aeronautics/newtons-laws-of-motion/#:~:text=Newton's%20first%20law%20states%20that,state%20of%20motion%20is%20inertia.