Robot Learning

Group 80:Lukas Schneider(2565695), Si Jun Kwon(2453134), Michael Erni()

TECHNISCHE UNIVERSITÄT DARMSTADT

Sommersemester 2020
Sheet 1

Task 1.1: Robotics in a Nutshell

1.1a)

$$\bar{\mathbf{x}}^0_{\text{end-eff}} = \underbrace{\begin{pmatrix}\cos(q_1) & -\sin(q_1) & 0 & 0\\ \sin(q_1) & \cos(q_1) & 0 & 0\\ 0 & 0 & 1 & L1\\ 0 & 0 & 0 & 1\end{pmatrix}}_{\mathbf{R}^0_1}\underbrace{\begin{pmatrix}\cos(q_3) & -\sin(q_3) & 0 & q_2\\ \sin(q_3) & \cos(q_3) & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}}_{\mathbf{R}^1_2}\underbrace{\begin{pmatrix}1 & 0 & 0 & q_4\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}}_{\mathbf{R}^2_{\text{end-eff}}}\underbrace{\begin{pmatrix}0\\0\\0\\1\end{pmatrix}}_{\bar{\mathbf{x}}^e_{\text{end-eff}}}$$

$$= \underbrace{\begin{pmatrix}\cos(q_1) & -\sin(q_1) & 0 & 0\\ \sin(q_1) & \cos(q_1) & 0 & 0\\ 0 & 0 & 1 & L1\\ 0 & 0 & 0 & 1\end{pmatrix}}_{\mathbf{R}^0_1}\underbrace{\begin{pmatrix}\cos(q_3) & -\sin(q_3) & 0 & q_4\cos(q_3)+q_2\\ \sin(q_3) & \cos(q_3) & 0 & q_4\sin(q_3)\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}}_{\mathbf{R}^1_{\text{end-eff}}}\begin{pmatrix}0\\0\\0\\1\end{pmatrix}$$

$$= \underbrace{\begin{pmatrix}a & b & 0 & q_4\cos(q_1)\cos(q_3)+q_2\cos(q_1)-q_4\sin(q_1)\sin(q_3)\\ c & d & 0 & q_4\sin(q_1)\cos(q_3)+q_2\sin(q_1)+q_4\cos(q_1)\sin(q_3)\\ 0 & 0 & 1 & L1\\ 0 & 0 & 0 & 1\end{pmatrix}}_{\mathbf{R}^0_{\text{end-eff}}}\begin{pmatrix}0\\0\\0\\1\end{pmatrix}$$

$$= \begin{pmatrix}q_4\cos(q_1)\cos(q_3)+q_2\cos(q_1)-q_4\sin(q_1)\sin(q_3)\\ q_4\sin(q_1)\cos(q_3)+q_2\sin(q_1)+q_4\cos(q_1)\sin(q_3)\\ L1\\ 1\end{pmatrix} =^1 \begin{pmatrix}q_4\cos(q_1+q_3)+q_2\cos(q_1)\\ q_4\sin(q_1+q_3)+q_2\sin(q_1)\\ L1\\ 1\end{pmatrix}$$

1: Refer to angle sum identities for $\cos$ and sin.

where $a = \cos(q_1)\cos(q_3) - \sin(q_1)\sin(q_3)$, $b = -\cos(q_1)\sin(q_3) - \sin(q_1)\cos(q_3)$, $c = \sin(q_1)\cos(q_3) + \cos(q_1)\cos(q_3)$ and $d = -\sin(q_1)\sin(q3) + \cos(q_1)\cos(q_3)$.

$\bar{\mathbf{x}}^0_{\text{end-eff}}$ is the homogenous coordinate of the end-effector in base-coordinates and $\bar{\mathbf{x}}^e_{\text{end-eff}}$ is the position of the end-effector in homogenous end-effector-coordinates. Hence:

$$\begin{pmatrix}x\\y\\z\end{pmatrix} = \begin{pmatrix}q_4\cos(q_1+q_3)+q_2\cos(q_1)\\ q_4\sin(q_1+q_3)+q_2\sin(q_1)\\ L1\end{pmatrix}$$

1.1b)

The main problem in a inverse kinematics problem is the ambiguity of its solution. Depending on the desired position and structure of the robot, there may be one, infinetly many or no joint configuration to accomplish the given desired position and rotation. This makes it often impossible to find an analytical solution and requires the problem to be solved numerically.

1.1c)

$$\mathbf{J}(\mathbf{q}) = \begin{pmatrix} -q_4 \sin(q_1+q_3) - q_2 \sin(q_1) & \cos(q_1) & -q_4 \sin(q_1+q_3) & \cos(q_1+q_3) \\ q_4 \cos(q_1+q_3) + q_2 \cos(q_1) & \sin(q_1) & q_4 \cos(q_1+q_3) & \sin(q_1+q_3) \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

The Jacobian matrix represents the proportion of how much the end-effector position(also orientation but not in our case) in task space changes when the joint values change. For instance $\mathbf{J}(\mathbf{q})_{1,1}$ tells us how the x-coordinate changes in task space when joint q_1 changes.

1.1d)

Mathematically speaking the robot is in a kinematic singularity if at least two columns are linearly dependent(meaning two joints of the same type are colinearly aligned meaning they could cancel each others movements out). Thus a singularity can be detected if the Jacobian is not invertible or in other words the determinant of the Jacobian is 0. Our robot enters a kinematic singularity as soon as q_3 equals to 0 or $\pm\pi$.

1.1e)

No, it is inappropriate for sorting items on a table unless it is attached to a mobile base. The main problem here is the static height the end-effector and the links have. Because of its fixed height, it cannot dodge objects effectively. This would mean it would have to sort the items to be placed farthest from it first or it has to arrange items in the order from left to right. The need of resorting just a single item could mean that the robot is required to move more objects than a robot that could adjust its height to dodge objects in its path, conclusively making it inefficient and inappropriate for this job.

Task 1.2

1.2a)

The PID controller extends the PD-controller by adding a integral component. This intergral component accumulates the error between the actual value and the desired value in the past and compensates it. This could be very helpful for controlling steady state systems where the desired value do not change frequently. In such cases the PID controller may compensate for a constant offset to the desired value. In a dynamic system where the desired value does change frequently like in tracking control this is problematic as the PID controller may to compensate errors that aren't present anymore but were present in the past.

1.2b)

The equation of forces for this robot consists of inertia, coriolis effect, gravitational forces and the torque generated by the engines.

$$M(q)\ddot{q} = -i(q,\ddot{q}) - c(q,\dot{q}) - g(q) + u$$

The inverse dynamics would then be:

$$u = M(q)\ddot{q} + i(q,\ddot{q}) + c(q,\dot{q}) + g(q)$$

1.2c)

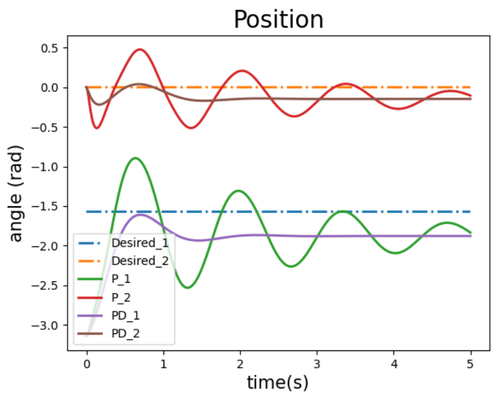


Figure 1: P and PD Controller tracking a desired point

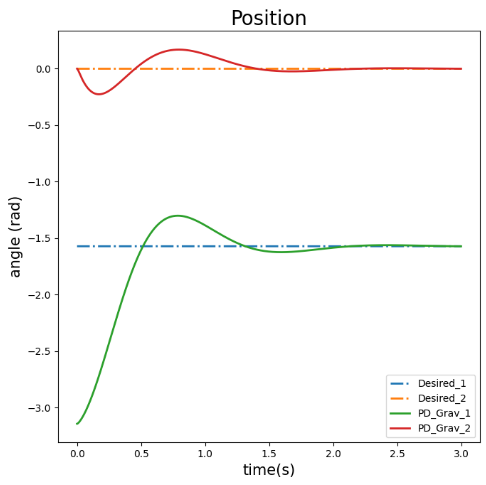


Figure 2: PD Controller with gravity compensation tracking a desired point

The P Controller did not manage to converge to the desired point in the simulated time interval. It oscillates due to the fact that the P Controller only considers the error between the current joint position and desired position and does not consider the velocity while approaching the desired position. This causes the robot to overshoot the desired position. It

also does not consider gravity and will therefore not converge to the desired point.
The PD Controller does converge in the simulated time interval but also does not converge to the desired point. This is due to the fact that it also does not consider gravitational forces. The PD controller with gravity compensation converges as fast as the PD-Controller but converges to the desired point without a constant offset and converges much faster.

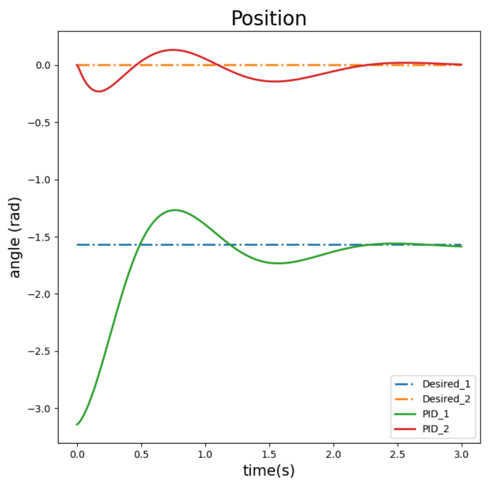


Figure 3: PID Controller tracking a desired point

The PID controller does not compensate gravity explicitly. It still converges to the desired as it accumulates the errors and compensates them. Plotting beyond 3 seconds of simulation time reveals that although the PID controller converges to the desired position the integral part causes additional oscillations.

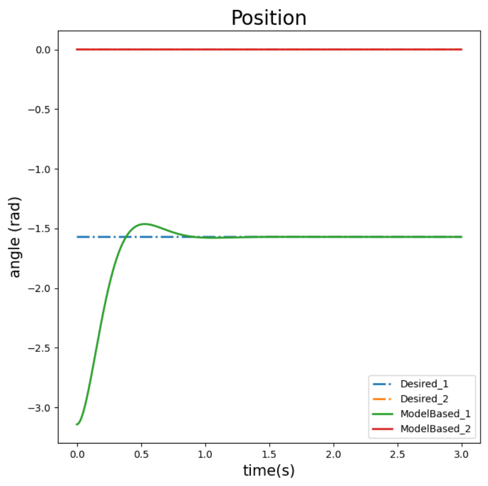


Figure 4: Model based Feedback control

The model based controller is the most accurate controller and the fastest to converge. Additionally unlike other controllers this controller holds the second joint's initial position which is also the desired position. The controller does this by compensating the gravitational forces and additionally compensating for the inertia caused by moving the first joint without moving the second.

In conclusion for this problem we would choose the model-based controller as the the given model has proven itself to be accurate and efficient. In general though obtaining an efficient model that is accurate is hard to do which is why a lot of the times a linear feedback controller is selected.

1.2d)

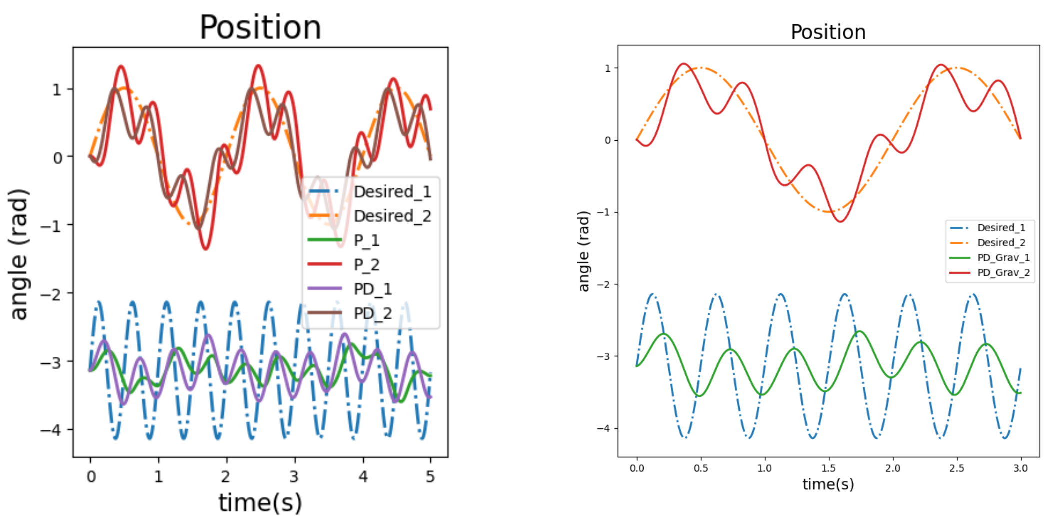


Figure 5: P and PD controller(left)/PD controller with gravity compensation(right) tracking a time-varying trajectory

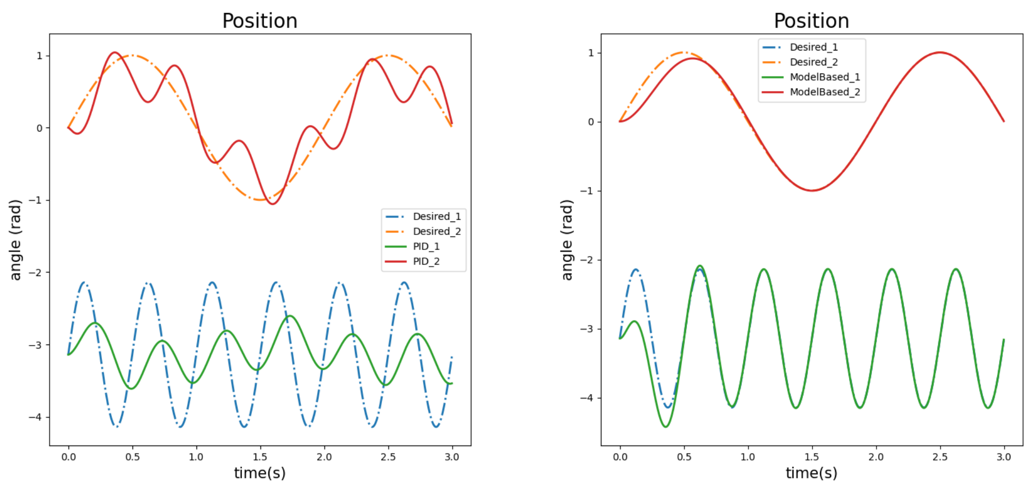
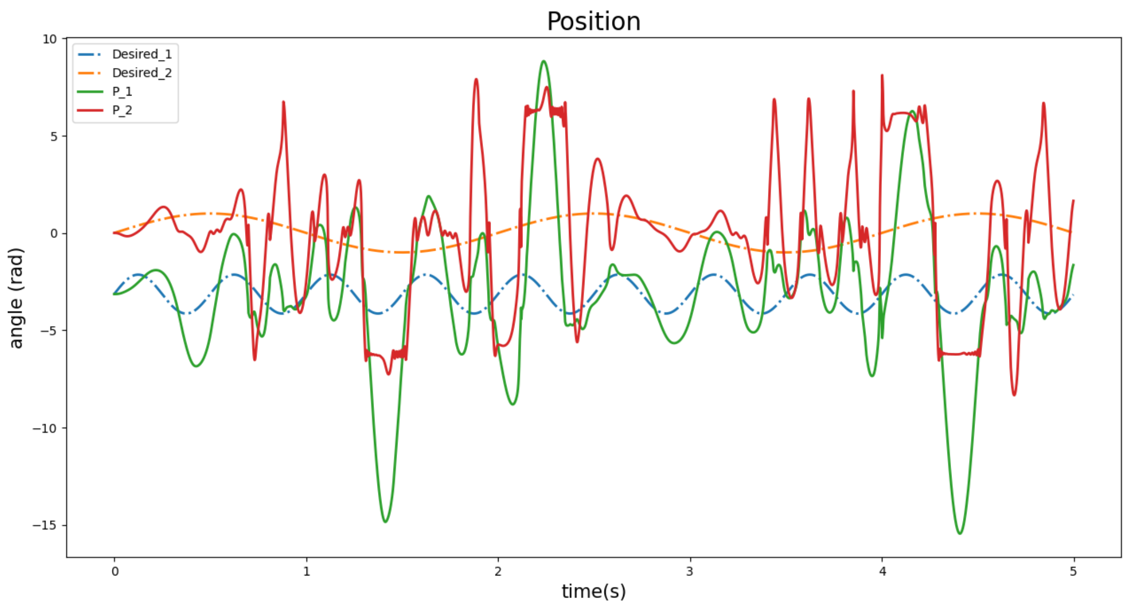


Figure 6: PID controller(left)/model based controller(right) tracking a time-varying trajectory

As we can see in the four plots above the model-based controller is the only controller that can follow the trajectory. The other controllers cannot keep up with the speed of change of the trajectory. This indicates that the gains are set too low for this problem.

1.2e)



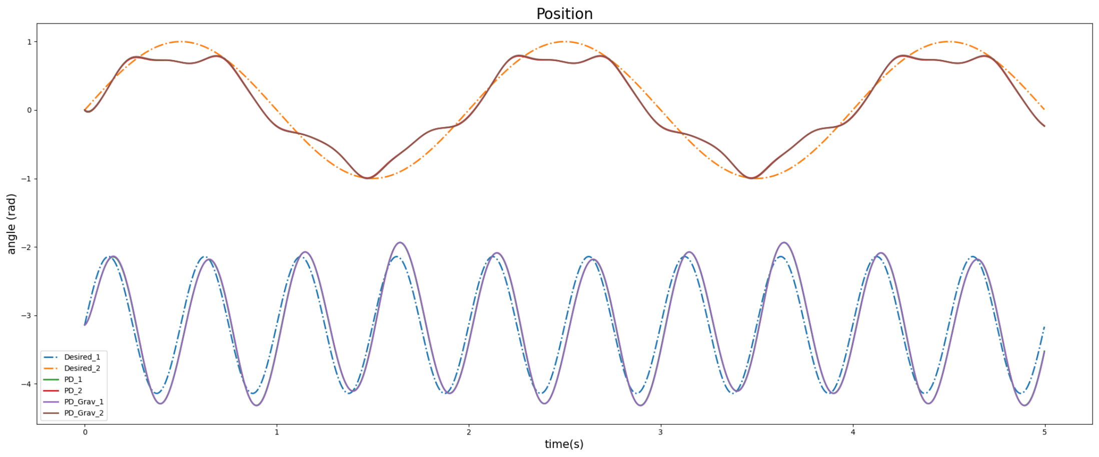
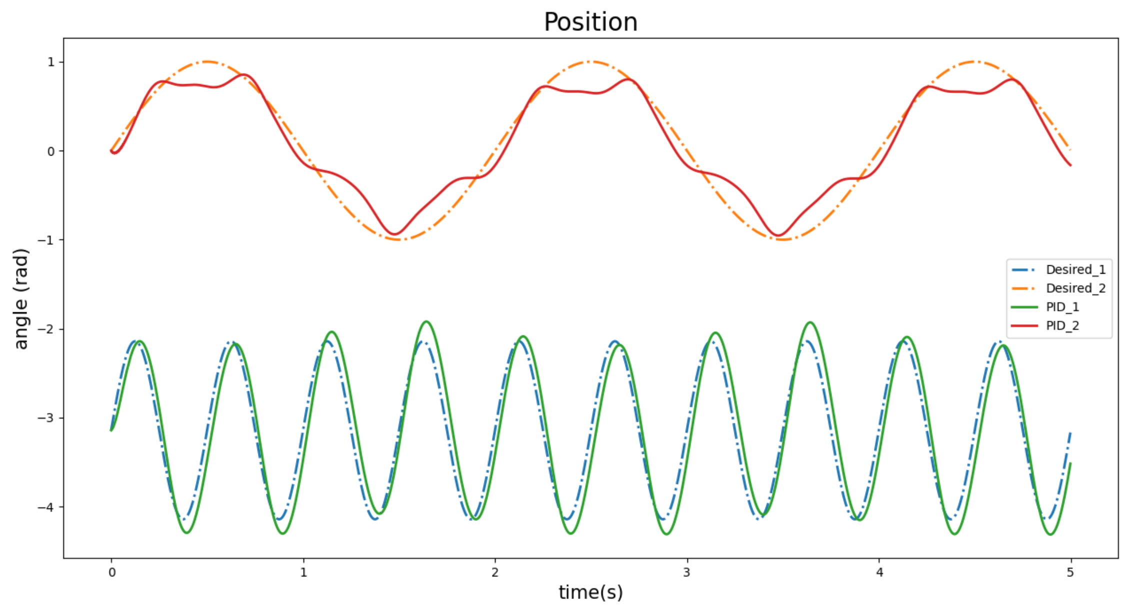


Figure 7: P controller(above)/PD controller(with and without gravity compensation)(below) tracking a time-varying trajectory with amplified gains



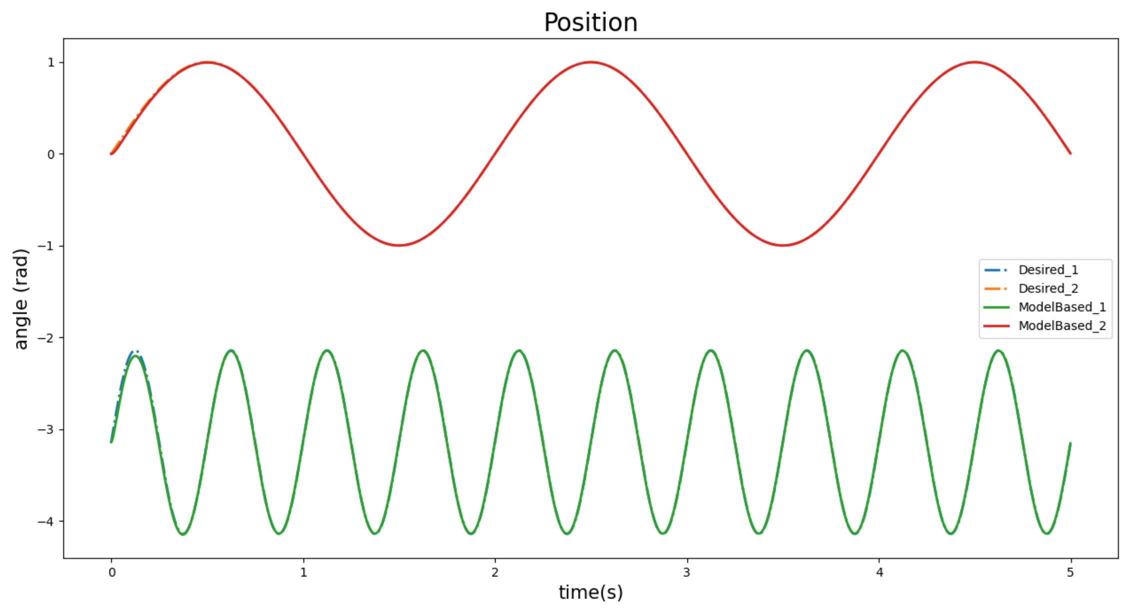


Figure 8: PID controller(above)/model based controller(below) tracking a time-varying trajectory with amplified gains

The amplification lead to a improvement in tracking the trajectory for all controllers except the P controller. The P controller became unstable as a result of increasing the gain. The PD(with and without gravity compensation) and PID controller was able to track the trajectory of the first joint very accurately. The PD and PID does not account for the inertia caused by moving the first joint and as a result of that the second joint still has some inaccuracy. The model-based which was already very accurate before the amplification of the gains has become accurate more faster, nearly having a very accurate tracking behaviour from the beginning. Increasing the gain has to be done with precaution though, as increasing the gain too much results in the system becoming unstable.

1.2f)

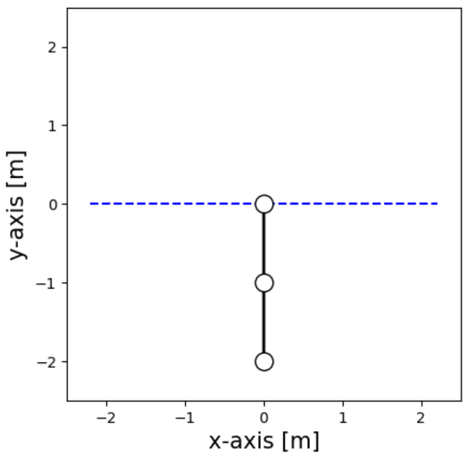


Figure 9: Initial position of the robot

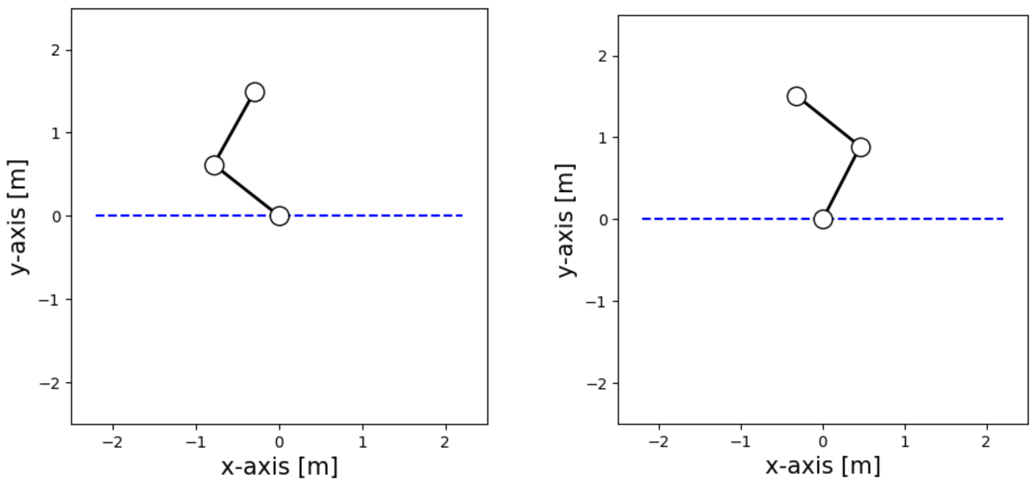


Figure 10: Final configuration of the robot with resting position $\begin{pmatrix}0 & \pi\end{pmatrix}$(left) and $\begin{pmatrix}0 & -\pi\end{pmatrix}$(right)

The resting position determines how the robot rotates. If the resting position $\begin{pmatrix}0 & \pi\end{pmatrix}$ is chosen, joint 1 rotates clockwise while choosing the other resting position causes it to rotate to the right side. This is due to the sign of the second joint of the two resting positions. π points to the clockwise rotation while $-\pi$ points to counter-clockwise rotation.