University of Groningen

Robotics for IEM

Lab Practical Report

Author:
Jonathan Chandra
(s5161533) - Group 24

Lecturer:
Dr. Bahar Haghighat

October 24, 2023

1 Introduction

The lab practical covers the experimental application of Kinematics, Jacobians and Trajectories of the low-cost robotic manipulator AX-18A Smart Robotic Arm depicted in Figure 1.

Figure 1: AX-18A Smart Robotic Arm.

Recall the forward and inverse kinematics that was derived in the first and second Assignment. We have the value of each link length as $L_1 = L_2 = 17, L_3 = 7, L_4 = L_5 = 4, L_6 = 9$ (in cm). The forward kinematics solution can be defined with the DH parameter in the following table:

Link	a_i	α_i	d_i	θ_i
1	0	0	L_1	θ_1
2	0	$\frac{\pi}{2}$	0	$\theta_2+\frac{\pi}{2}$
3	L_2	$\frac{\pi}{2}$	0	$\theta_3+\frac{\pi}{2}$
4	L_4	$\frac{\pi}{2}$	$L_3 + L_5$	$\theta_4+\frac{\pi}{2}$
5	0	0	0	θ_5

Table 1: DH Parameters of AX-18A Smart Robotic Arm

While the inverse kinematic solution with $[\theta_4, \theta_5] = [0, 0]$ can be defined

with:

$$\begin{aligned}\theta_1 &= [\text{atan2}(y,x)]\text{or}[\text{atan2}(y,x)+\pi] \\ \theta_2 &= \text{atan2}(s_2, c_2) \\ \theta_3 &= \text{atan2}(\pm\sqrt{1-\cos(\theta_3+\beta)^2}, \cos(\theta_3-\beta)) - \beta\end{aligned} \tag{1}$$

where:

$$\begin{aligned}\begin{bmatrix} c_2 \\ s_2 \end{bmatrix} &= \frac{1}{-k_1^2-k_2^2}\begin{bmatrix} -k_1 & -k_2 \\ -k_2 & k_1 \end{bmatrix}\begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \\ k_1 &= (4\cos(\theta_3) + 20\sin(\theta_3)) \\ k_2 &= (-4\sin(\theta_3) + 20\cos(\theta_3) + 17) \\ b_1 &= \pm\sqrt{x^2+y^2} \\ b_2 &= z - 17 \\ \beta &= \text{atan2}(L_4, L_3+L_5+L_6)\end{aligned} \tag{2}$$

2 Practical Setup

In this lab practical, two main task were done, that is 1) MATLAB simulation and 2) Real time implementation of kinematic and trajectory on the AX-18A Smart Robotic Arm.

2.1 MATLAB simulation

A Matlab simulation is done before implementing the joint trajectories on the AX-18A smart robotic arm. The simulation was done with Peter Corke's Robotic Toolbox to create 1) the robot model, 2) Inverse kinematic solution, and 3) Joint trajectories of the initial to the end pose of the robot arm.

The robot depicted in Figure 1 consists of five links and an end effector. The model is generated based on the DH parameter presented in Table 1 with the end effector defined with a homogenous transformation matrix:

$$\texttt{pArb.tool} = {}^{5}_{6}T = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 9 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

In this Lab practical, 4 set of initial and end position of the robot end

effector in the world frame is defined, that is:

$$\begin{aligned}
[x, y, z]_1 &= [-4, 0, 54]_{(\text{initial})} \rightarrow [0, -21, 37]_{(\text{end})} \\
[x, y, z]_2 &= [-4, 0, 54]_{(\text{initial})} \rightarrow [0, 32, 25]_{(\text{end})} \\
[x, y, z]_3 &= [0, -21, 37]_{(\text{initial})} \rightarrow [-4, 0, 54]_{(\text{end})} \\
[x, y, z]_4 &= [0, 32, 25]_{(\text{initial})} \rightarrow [-4, 0, 54]_{(\text{end})}
\end{aligned} \tag{4}$$

The end effector's initial and end position was derived from a forward kinematic solution using `pArb.fkine(q)`, where q represents the joint angles (θ_i) of the robot arm joints. The first position, q_1, transitions from $[0; 0; 0; 0; 0]$ to $[\frac{\pi}{2}; \frac{\pi}{2}; -\frac{\pi}{2}; 0; 0]$. The second position, q_2, transitions from $[0; 0; 0; 0; 0]$ to $[-\frac{\pi}{2}; \frac{\pi}{4}; -\frac{\pi}{4}; 0; 0]$. Meanwhile, q_3 is the transition from $[\frac{\pi}{2}; \frac{\pi}{2}; -\frac{\pi}{2}; 0; 0]$ to $[0; 0; 0; 0; 0]$, which is the inverse of q_1. Similarly, q_4 transitions from the ending position of q_2 back to its starting position, making it the inverse of q_2.

Next we use `pArb.ikine` function to create the inverse kinematic solution of the initial and end position of the end effector. Note that in the `pArb.ikine` function, a mask function is defined to not consider the orientation (roll, pitch, yaw) of the end effector in the inverse kinematic calculation.

A trajectory between the initial and end point is then generated using `jtraj` function with the input of initial and the end inverse kinematic solution with an addition of n number of steps between those two points. `jtraj` uses fifth order polynomials to interpolate between the initial and final position with default zero boundary conditions for velocity and acceleration. Figure 2 to 5 shows the simulation result on the robot virtual model and plot trajectory of each joints

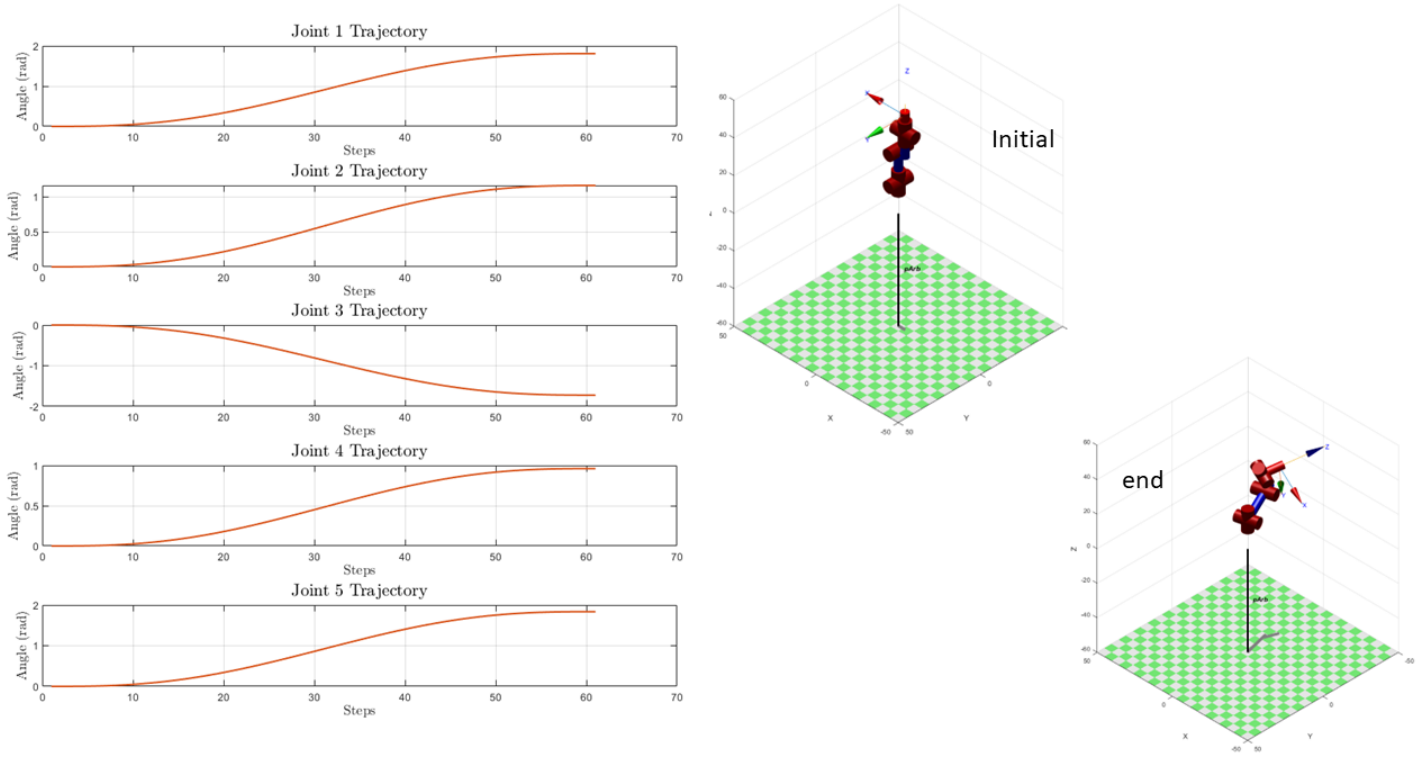

Figure 2: Result on $[x, y, z]_1 = [-4, 0, 54]_{(\text{initial})} \rightarrow [0, -21, 37]_{(\text{end})}$

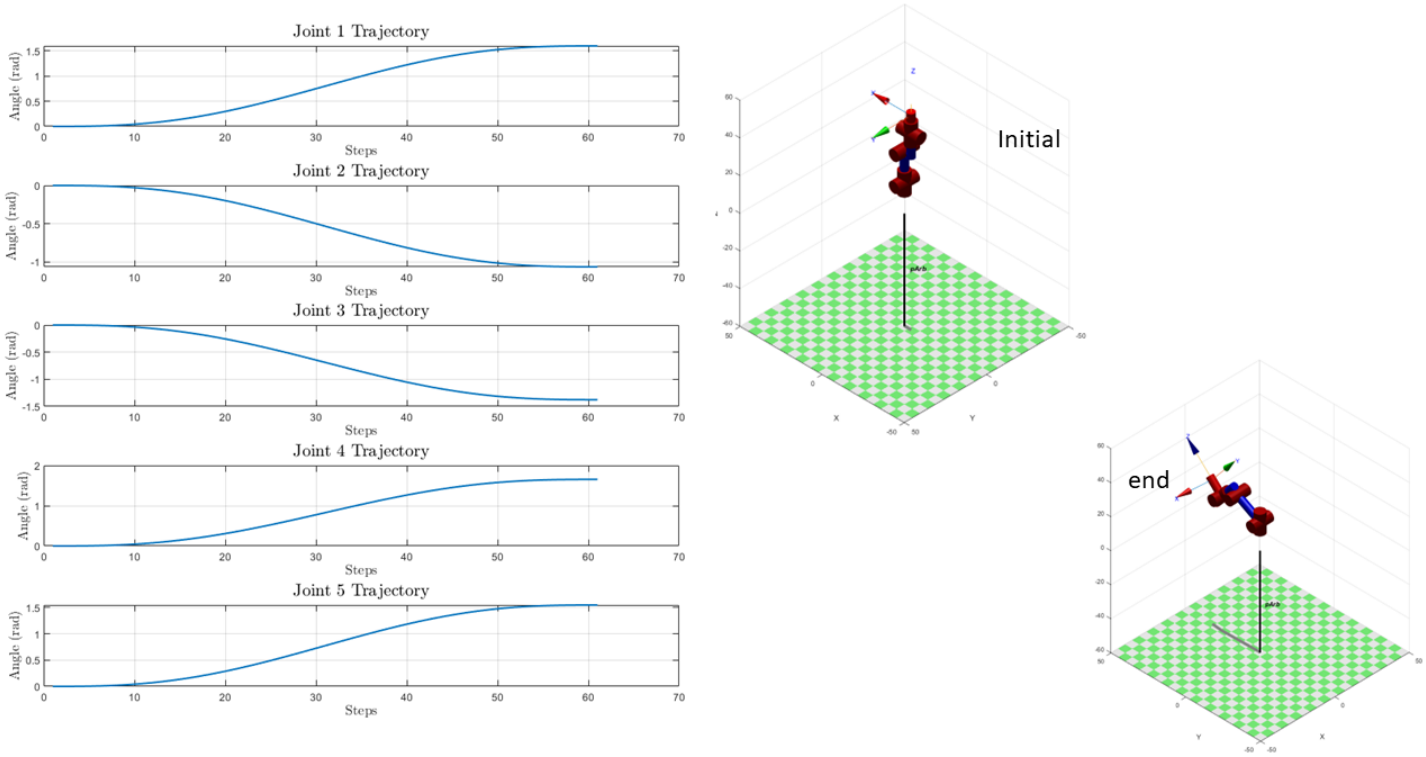

Figure 3: Result on $[x, y, z]_2 = [-4, 0, 54]_{(\text{initial})} \rightarrow [0, 32, 25]_{(\text{end})}$

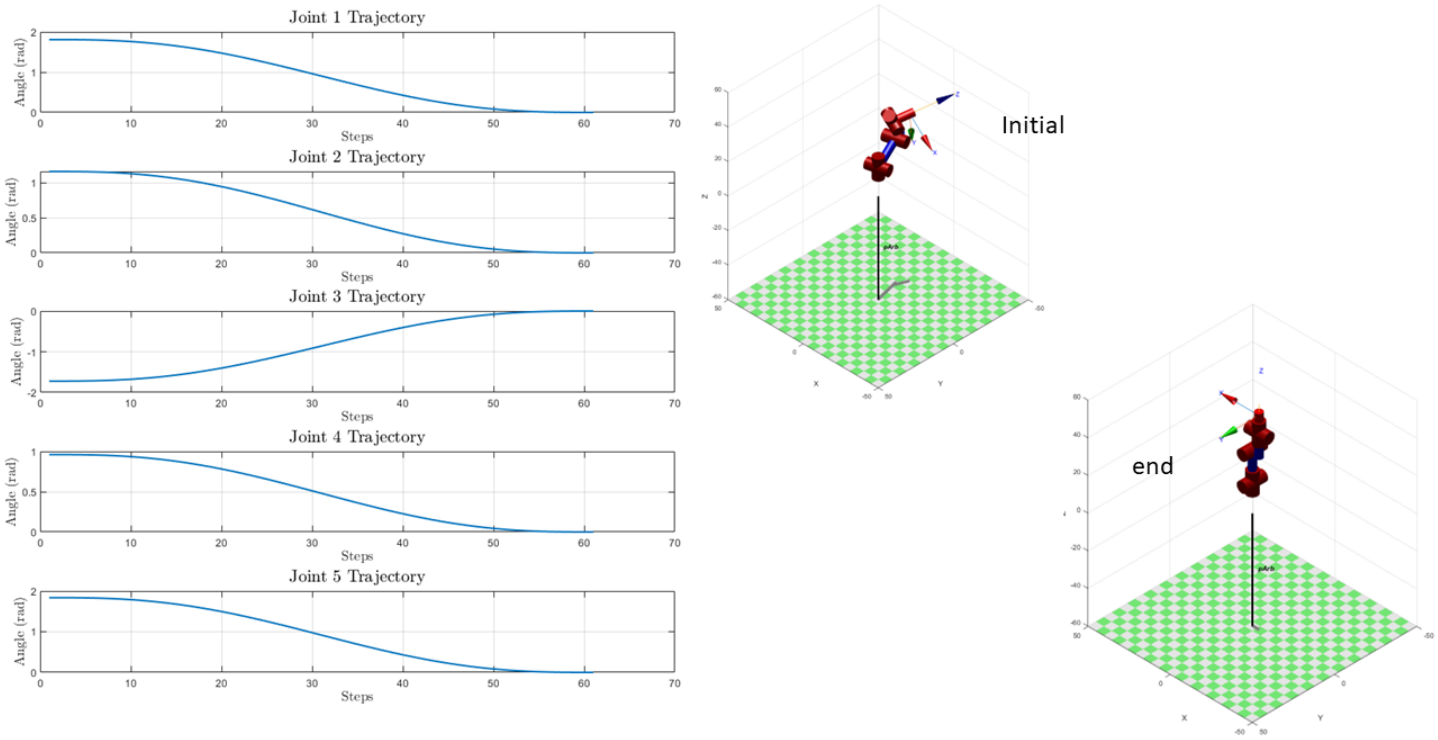

Figure 4: Result on $[x, y, z]_3 = [0, -21, 37]_{(\text{initial})} \rightarrow [-4, 0, 54]_{(\text{end})}$

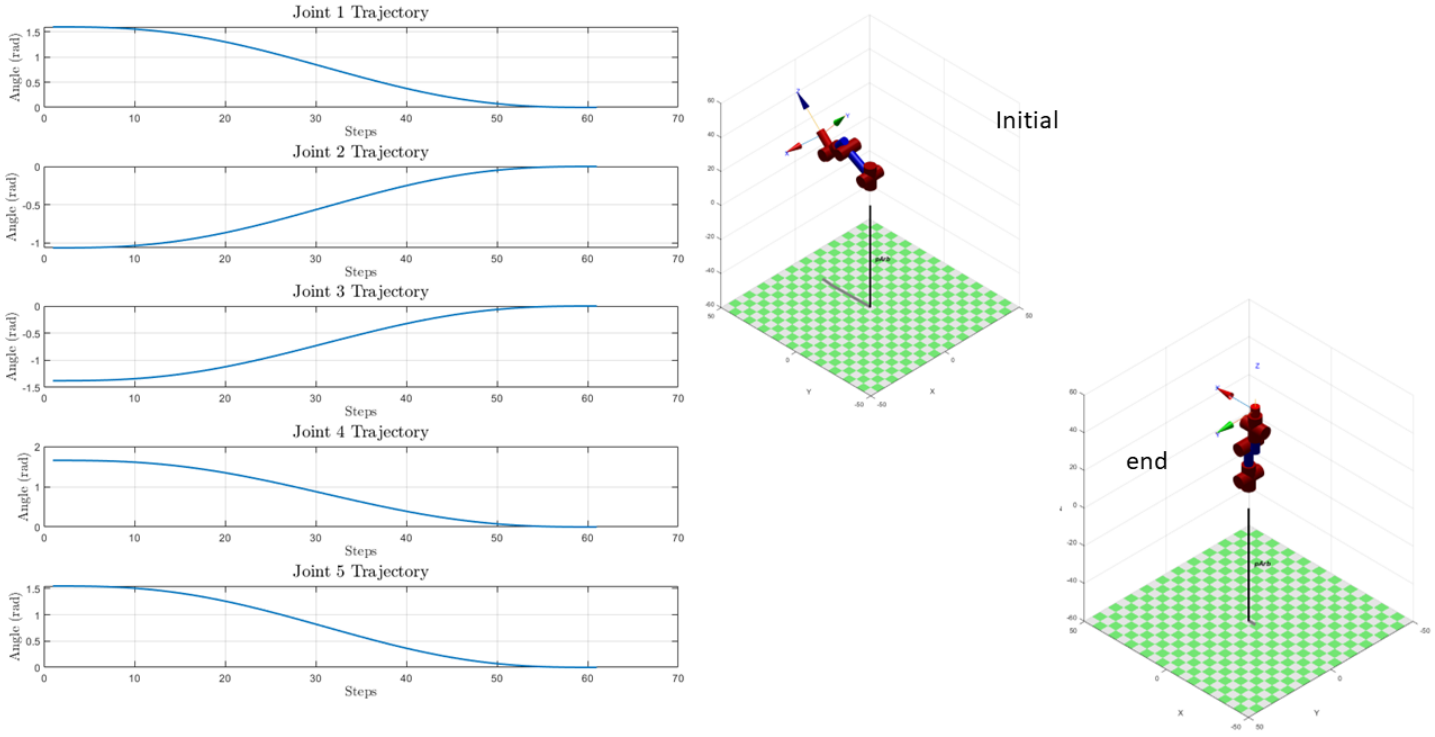

Figure 5: Result on $[x, y, z]_4 = [0, 32, 25]_{(\text{initial})} \rightarrow [-4, 0, 54]_{(\text{end})}$

2.2 Real Time Implementation

The trajectories generated in the previous section can be implemented in the AX-18A Smart Robotic Arm in Figure 1 by storing the generated trajectories q as a matrix in MATLAB. AX-18A Smart Robotic Arm uses a custom Arduino board called the Arbotix which interfaces the robotic arm with a personal computer. Figure 6 depicts the data flow of the whole AX-18A system

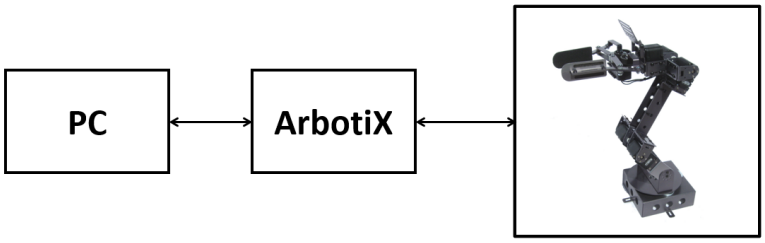

Figure 6: Data flow of the AX-18A system

A connection of Arbotix with the PC is defined in MATLAB with the function `arb = Arbotix('port','COMx','baud', 57610)` to determine which COM port and baud rate of the controller is used.

Several commands can be used to send order to or retrieve measurement data from the AX18-A robotic arm such as the servos current temperature and angle of each joint. In this practical, we are only interested to bring the end effector from a defined initial position to a defined end position.

The robotic arm is firstly set to the initial angle defined that is defined using `fkine` function with the input of end effector initial position. To move the joints with the values of the generated trajectories q from the previous subsection, a for loop can be used to load q with the function

```
for i =1:length(q)
    arb.setposall(q(i,1), q(i,2), q(i,3), q(i,4), q(i,5), 0)
end
```

Figure 7 shows the robot in zero configuration and this configuration also corresponds to the end position of the simulation shown in Figure 4 and 5.

Figure 7: Zero configuration of AX-18A system

Figure 8 shows the robot configuration that corresponds to the end position of the simulation shown in Figure 2.

Figure 8: AX-18A system set $[x, y, z] = [0, -21, 37]_{(\text{end})}$

Figure 9 shows the robot configuration that corresponds to the end position of the simulation shown in Figure 3.

Figure 9: AX-18A system set $[x, y, z] = [0, 32, 25]_{(\text{end})}$

Unfortunately, the real-time measurement of the joint angle was not saved as a variable during the lab practical. The joint angle can be measured using the function `arb.getpos(q)`. Therefore, a direct performance comparison by data is unfeasible.

In the process of transitioning from a simulated environment to real-world implementation of inverse kinematics using Jacobian trajectory, a notable contrast in performance has been observed. Specifically, the robot arm exhibits stuttering behavior in its joints, a phenomenon that was not seen in the simulation.

Several factors that may contribute to this divergence:

1. **Control Loop Timing**: Real-world systems are often subjected to latency and jitter, a factors that is not typically considered in simulated models.

2. **Motor Dynamics**: The behavior of motors under actual conditions subjected to factors like load, inertia, and friction is usually more complex than what is represented in simulations which is usually simplified.

3. **Computational Limitations**: The hardware responsible for running the control algorithms may not have the computational speed or reliability found in a simulation environment.

Note that the number of trajectories that were generated was using its default value, that is `t=0:0.05:3`.

A video demonstration of the real-time implementation issues can be found at Google Drive. The video specifically showcases the observed stuttering in the robot's joints when its moving from its initial position to the end position.

Recommendation for future work: Optimizing control loop timing and incorporate real-world motor dynamics into the models.

References

[1] Craig, J. (2021). Introduction to Robotics, Global Edition. United Kingdom: Pearson Education Limited.