University of South-Eastern Norway

Faculty of Technology, Natural Sciences and Maritime Sciences

IIA3120 Control For Robotic

Simulator for ReactorX-150 Robotic Arm Manipulator

Authors:
Torstein Solheim Ølberg: 263054; Magnus Eide: 253551;

17.12.2024

Table of Contents

1 Introduction

The use of robotic arms in industrial production and processes is widespread, particularly for performing repetitive, challenging, or hazardous tasks in a quick and efficient manner. Simulators play a crucial role in assessing the feasibility of robotic setups and evaluating their performance in terms of speed and precision. They also help in determining the overall value of automation compared to human labor. Many companies [1, 2] provide simulation tools for specific robotic systems, however this is not true for all robots.

This paper focuses on the development of simulation software tailored to the ReactorX-150 robotic arm manipulator, as inspired by the commission description [5]. The simulation aims to test both direct control and simulated control where a trajectory is generated and executed by the robot. A detailed drawing of the robot, including its joints and measurements, is shown in Figure 1.

The following sections describe the creation of the simulation software, the algorithms and models used for the simulator, and the results of testing the model. Finally, the paper discusses the outcomes, potential future improvements, and concludes with the key findings.

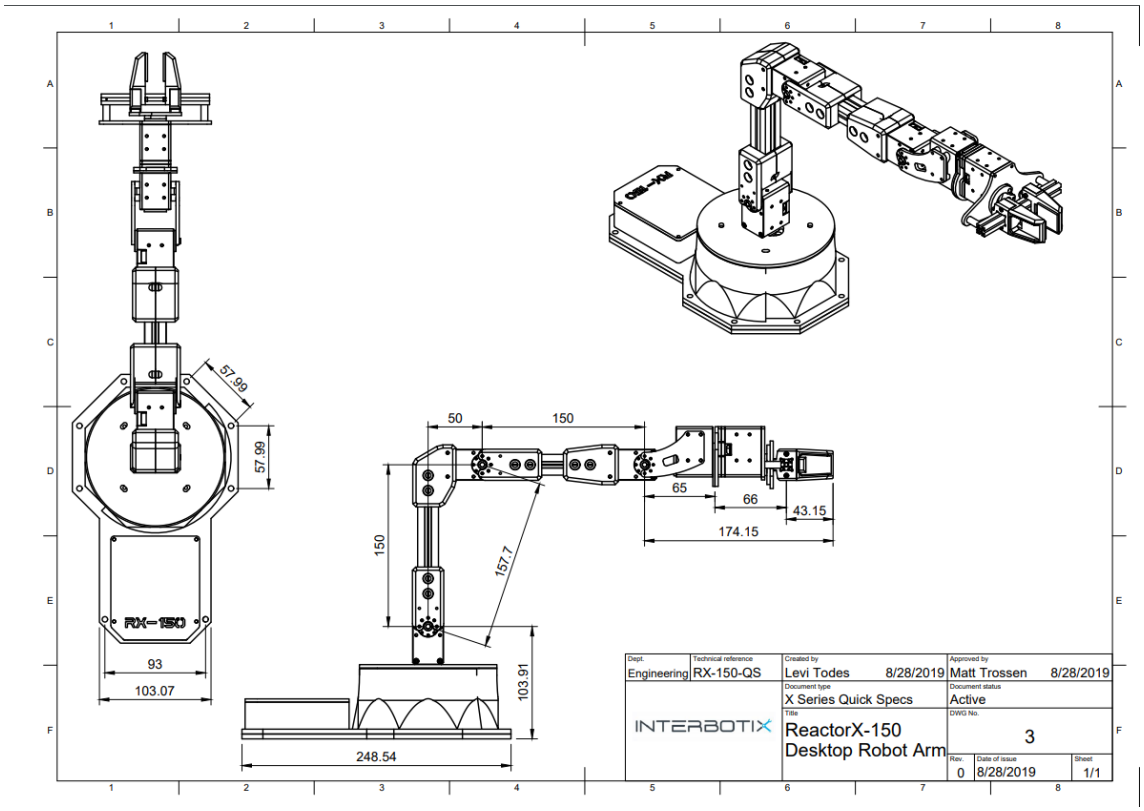


Figure 1: Descriptive drawing of the ReactorX-150 robotic arm manipulator.

2 Methods

Here, the tools and methods used will be described. First, the general setup of the robot model, the tools used for simulation and, how the testing of the model using direct control is performed. Then, tools for generating a trajectory through specific way-points. Finally, a model for performing forward kinematics on the joint angle data after the simulation of the movement is performed.

2.1 Simulator and robotic model

To build the simulator and render the simulation, the MATLAB programming language, and its add-ons, Simulink [7] and the Robotics System Toolbox [3] were used.

A simulator which can use joint angles as input and display the robot's movement in 3D space is needed to simulate the robot following a trajectory. For this, a URDF file [6] is used to explain the relations between the links and the joints and to give the robot body parts, in the form of 3D mesh files, to each link.

After importing the URDF into MATLAB and loaded it into Simulink with the `smimport` function, the joints then had to be made manipulable in the setting of each joint. The model was put into a subsystem to keep the program organized, and the subsystem got an input for each joint and one for the gripper. Inside the subsystem the inputs had to be converted from Simulink signals to physical system signals, for the joint blocks to accept the signal. The final subsystem is shown in Figure 2 and the control dashboard is in Figure 3.

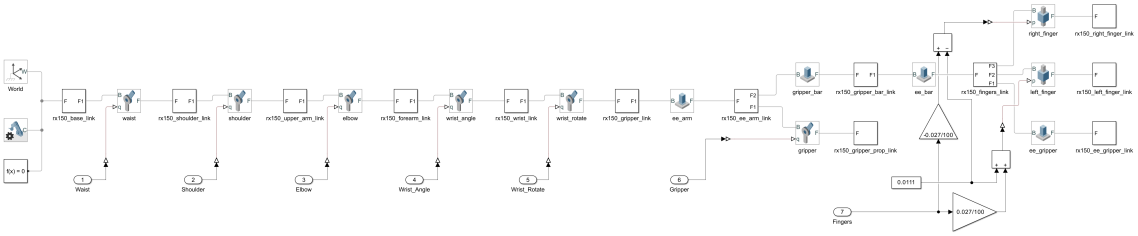


Figure 2: The Simulink model of the RX150 manipulator, models the behavior of the robot based on input from the dashboard.

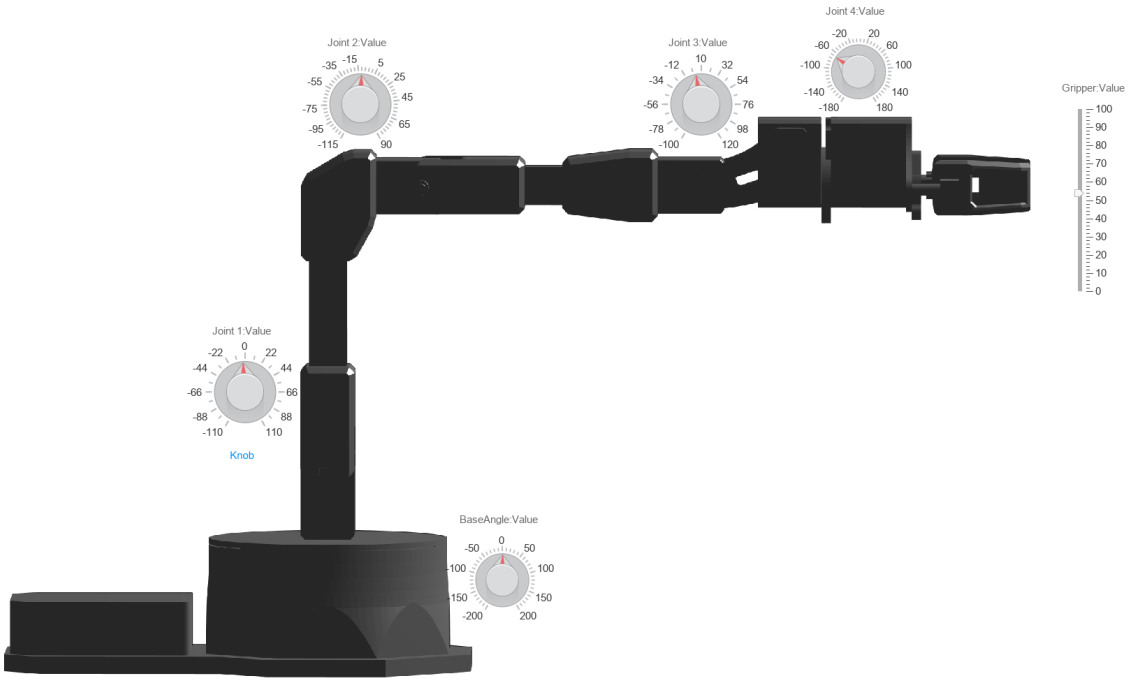


Figure 3: The control dashboard used for manual control and verification of the simulator model.

2.2 Trajectory and Inverse Kinematics

A 5-point waypoint mission was implemented with waypoints, waypoint velocities, and waypoint times. To simulate a waiting position, the third and fourth waypoints are the same. The inverse kinematics algorithm `cubicpolytraj` from MATLAB was used to create the path for the manipulator. In Figure 4, the trajectory is shown, about the manipulator.

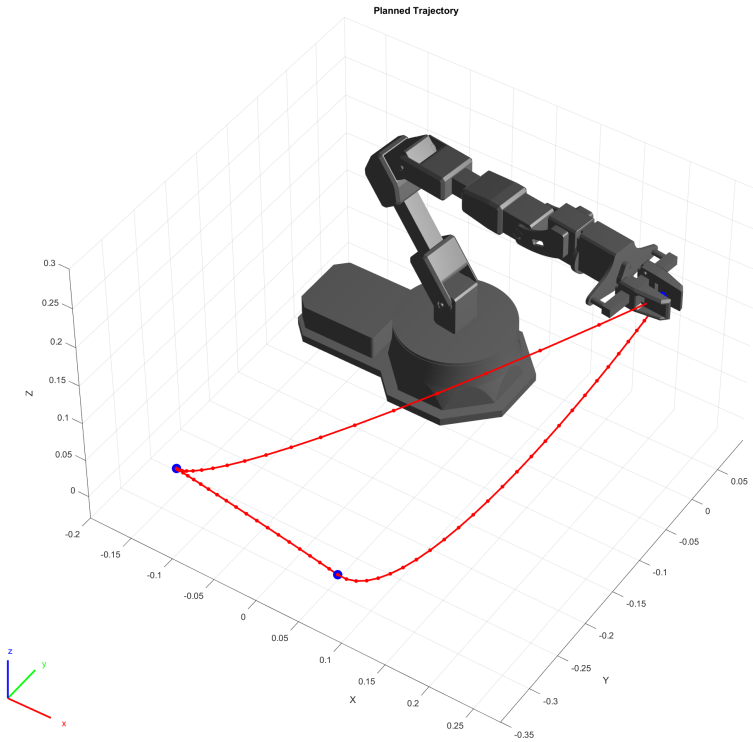


Figure 4: The planned trajectory in relation to the manipulator

To simulate the robot's movements, the Simulink program in Figure 5 replaces the dashboard from the previous section and uses the same waypoint parameters and the rigidBodyTree representing the manipulator to create the same trajectory as in Figure 4. The program also utilizes the inverse kinematic function to calculate the joint angles and gives them to the robot control subsystem previously created.

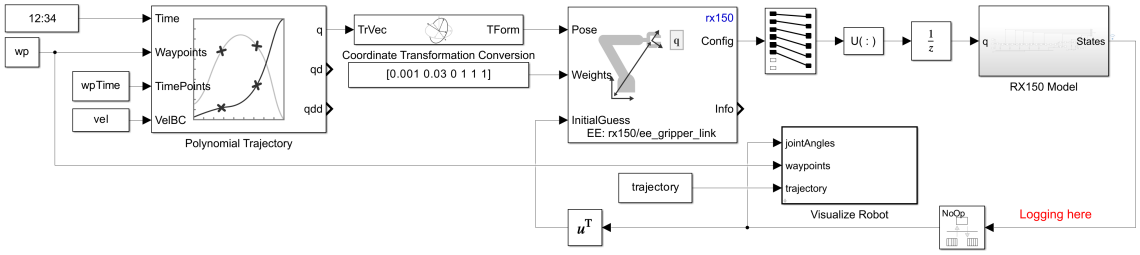


Figure 5: Simulink controller program for the manipulator, with trajectory generation and inverse kinematic functions.

The manipulator model subsystem had to be given two small changes as seen in Figure 6. The changes are:
1. Allow for a single input for all the joints.
2. The joints are set to be sensed and forwarded as an output to be used as the initial guess for the inverse kinematic algorithm, after being converted back to Simulink signals.

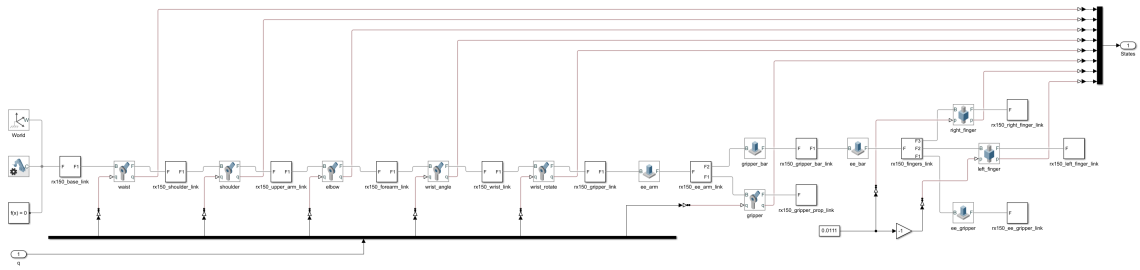


Figure 6: The modified RX150 Simulink model with inputs and outputs on the joint angles.

2.3 Forward Kinematics

Using the Denavit-Hartenberg (DH) convention to set the coordinate frames using the manufacturer's drawing(Figure 1) resulted in the coordinate frames in Figure 7. To calculate the position of the end effector only four frames are needed, pretending joint five is static, but for good practice, all five frames were included. Because of the offset in link two, coordinate frame two had to be adjusted $\frac{\pi}{2} - tan^{-}(\frac{50}{150})$ radians to comply with the first law of the Denavit-Hartenberg convention:

> x_i has to intersect both z_i and z_{i-1} perpendicular.

For the same reason, the third coordinate frame had to be rotated back to the same orientation as frame 1.

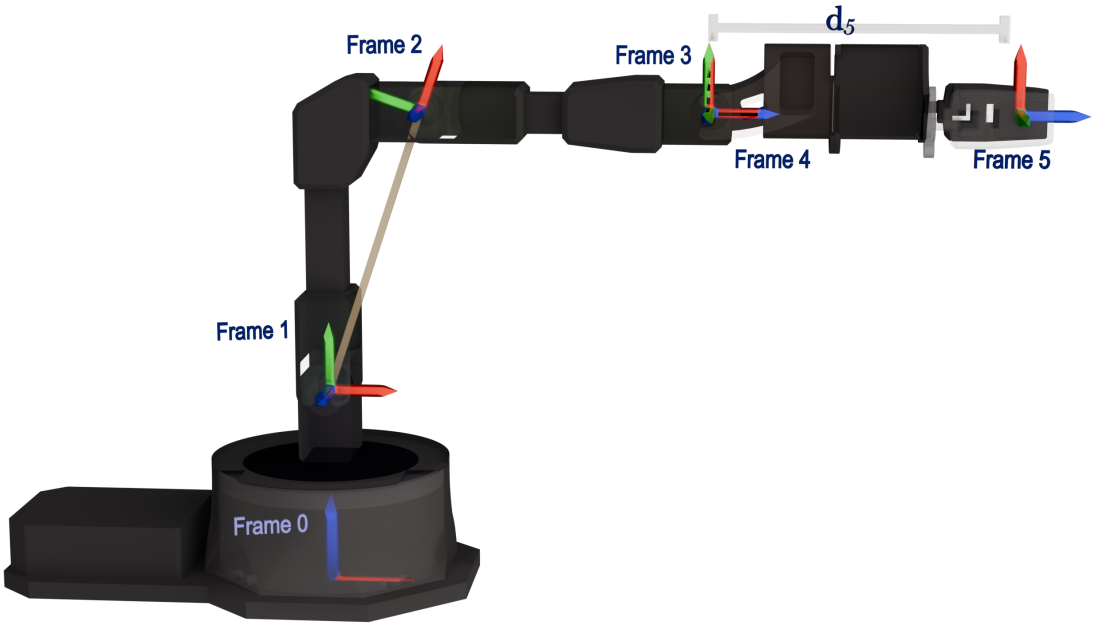


Figure 7: Coordinates on Denavit-Hartenberg convention. The red arrows are the x-axes, the green arrows are the y-axes and the blue arrows are the z-axes.

The DH parameters are used to describe the kinematics of the robot by defining the transformation between consecutive coordinate frames attached to each joint. The four DH parameters are:

d_i = Distance from j_{i-1} to j_i along z_{i-1}.
θ_i = Angle between x_{i-1} and x_i measured about z_{i-1}.
a_i = Distance between z_{i-1} and z_i measured along x_i.
α_i = Angle between z_{i-1} and z_i measured about x_i.

To follow the convention's four rules, the origin of the fourth coordinate frame was moved to the same position as coordinate frame three, extending the distance of d_5 by the length of link four, in addition to the distance to the end effector position from the end of link five. The DH parameters for each joint(j) is presented in 1.

i	d_i	θ_i	a_i	α_i
1	0.10391	q_1	0	$\frac{\pi}{2}$
2	0	$q_2 + \frac{\pi}{2} - \tan^{-1}\left(\frac{50}{150}\right)$	0.1577	0
3	0	$q_3 - \frac{\pi}{2} + \tan^{-1}\left(\frac{50}{150}\right)$	0.150	0
4	0	$q_4 + \frac{\pi}{2}$	0	$\frac{\pi}{2}$
5	0.15857	q_5	0	0

Table 1: Denavit-Hartenberg table with rotation measured in radians and distance in meters

$$ {}^{i-1}A_i = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i)\cdot\cos(\alpha_i) & \sin(\theta_i)\cdot\sin(\alpha_i) & a_i\cdot\cos(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i)\cdot\cos(\alpha_i) & -\cos(\theta_i)\cdot\sin(\alpha_i) & a_i\cdot\sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

The parameters can be used in a transformation matrix to mathematically describe the orientation of one joint in relation to the previous joint. By multiplying all of the A matrices the pose of the end effector is found, but since we are only interested in the position we can ignore the last joint angle and set $\theta_5 = 0$. This simplifies the final transformation matrix to:

$$ {}^0A_5 = \begin{bmatrix} {}^0R_5 & {}^0T_5 \\ 0 & 1 \end{bmatrix}, $$

$$ {}^0R_5 = \begin{bmatrix} -\frac{sin(q_1+q_2+q_3+q_4)+sin(q_2-q_1+q_3+q_4)}{2} & sin(q_1) & \frac{cos(q_1+q_2+q_3+q_4)+cos(q_2-q_1+q_3+q_4)}{2} \\ \frac{cos(q_1+q_2+q_3+q_4)-cos(q_2-q_1+q_3+q_4)}{2} & -cos(q_1) & \frac{sin(q_1+q_2+q_3+q_4)-sin(q_2-q_1+q_3+q_4)}{2} \\ cos(q_2+q_3+q_4) & 0 & sin(q_2+q_3+q_4) \end{bmatrix}, $$

$$ {}^0T_5 = \begin{bmatrix} \frac{cos(q_1)(15857\cdot cos(q_2+q_3+q_4)+15000\cdot cos(q_2+q_3)-15770\cdot sin(q_2-\phi))}{100000} \\ \frac{sin(q_1)(15857\cdot cos(q_2+q_3+q_4)+15000\cdot cos(q_2+q_3)-15770\cdot sin(q_2-\phi))}{100000} \\ \frac{15857\cdot sin(q_2+q_3+q_4)}{100000} + \frac{3\cdot sin(q_2+q_3)}{20} + \frac{1577\cdot cos(q_2-\phi)}{10000} + \frac{10391}{100000} \end{bmatrix}. $$

$$ \text{Where } \phi = \frac{5796142707547873}{18014398509481984} \approx 0.322 $$

The X, Y, and Z positions of the end effector, in relation to the base, are here represented by the three equations in ${}^0T_5$.

The forward kinematics required inverting joint angles q_2, q_3, and q_4, due to a mismatch between the DH parameters used in the calculation and the configuration in the URDF file. This inversion corrected discrepancies caused by differing approaches to the conventions, ensuring comparable end-effector positions in the calculations.

3 Results

3.1 Simulation

The manual simulator provided insight into how the manipulator behaves on joint angles. This feature allows for an accurate way of familiarizing a user with the capabilities of the manipulator. In Figure 8 the manipulator is set in a position, using the dashboard in Figure 3.

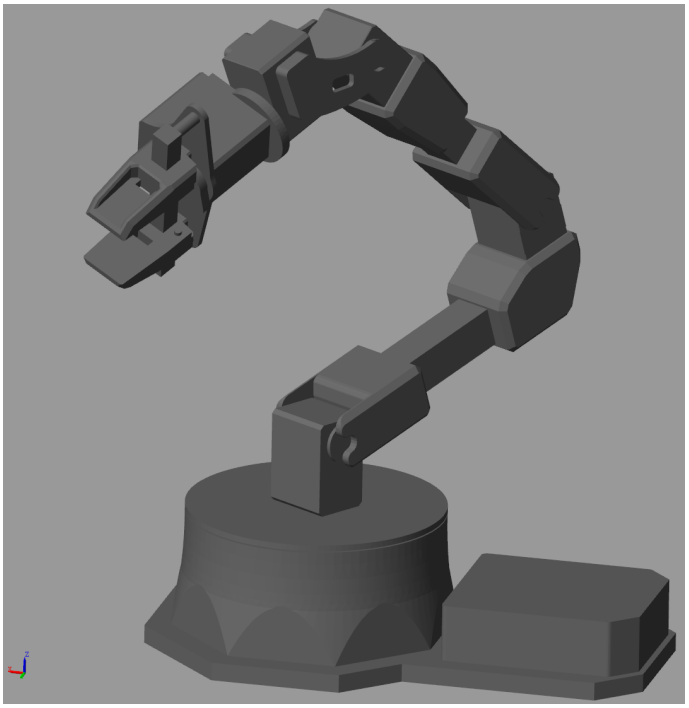
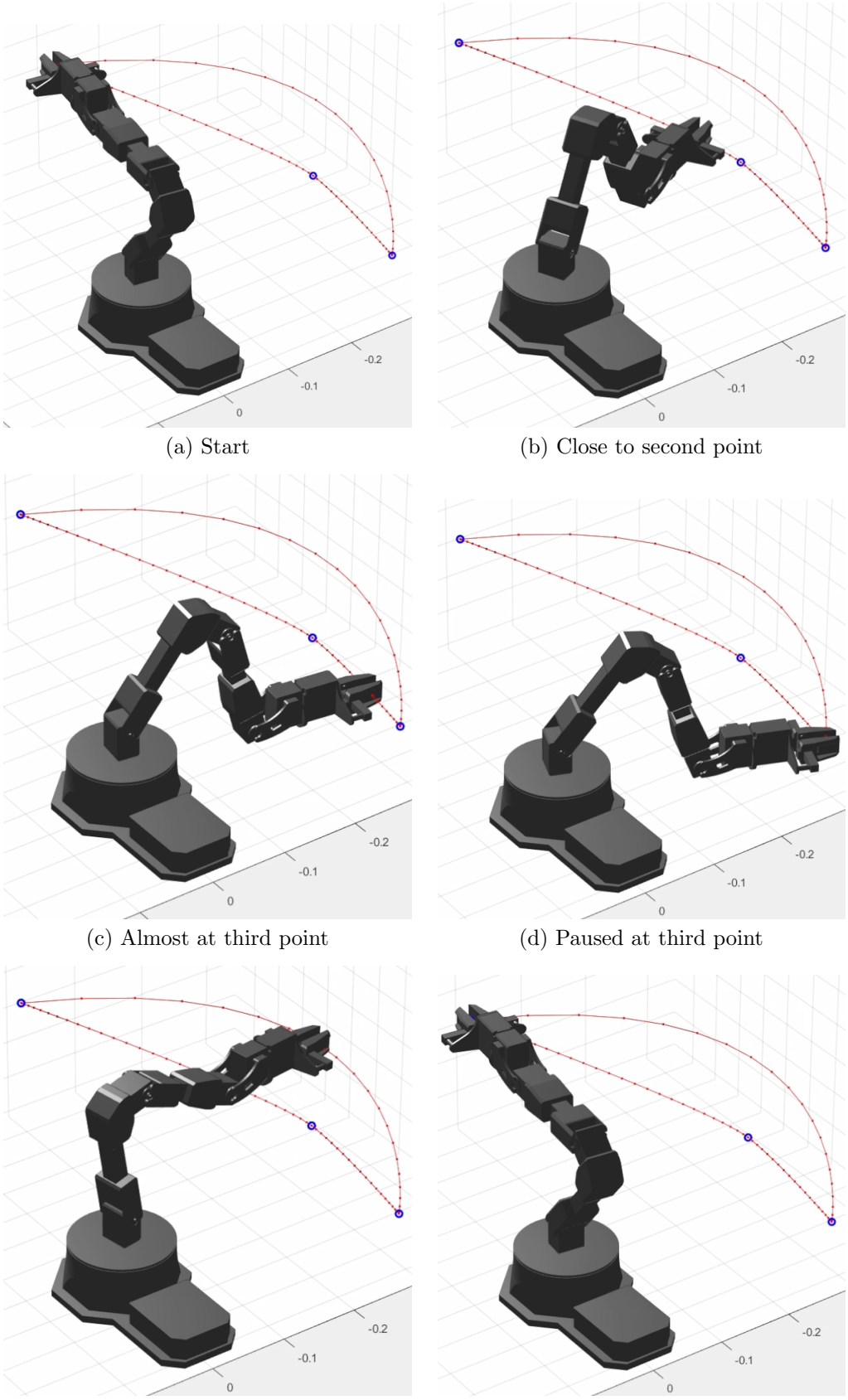


Figure 8: Manipulator in a position set from the dashboard.

3.2 Inverse Kinematics

The simulation of the inverse kinematic solution following the trajectory can be seen in Figure 9. A video of the simulated robot following the trajectory has also been recorded [4].



(a) Start

(b) Close to second point

(c) Almost at third point

(d) Paused at third point

(e) On its way back to the start

(f) Back at starting point

Figure 9: Snapshots of the simulated robot following the predefined trajectory through five points, where point three and four are at the same position, and point one and five are in the same position.

3.3 Forward Kinematics

The forward kinematic simulation produced a simulated movement along a trajectory, comparable to the path followed in the inverse kinematic simulation. This trajectory can be seen, along with the robot in the starting position, in Figure 10. The dots in the figure are the recorded steps of the robot, where the robot moves along the color gradient from red to blue.

Because the initial joint angles are zero the manipulator starts with a corrective action to get backward to the planned path. This is clear when looking at Figure 10 where the calculated trajectory is shown and proves the correctness of the DH parameters.

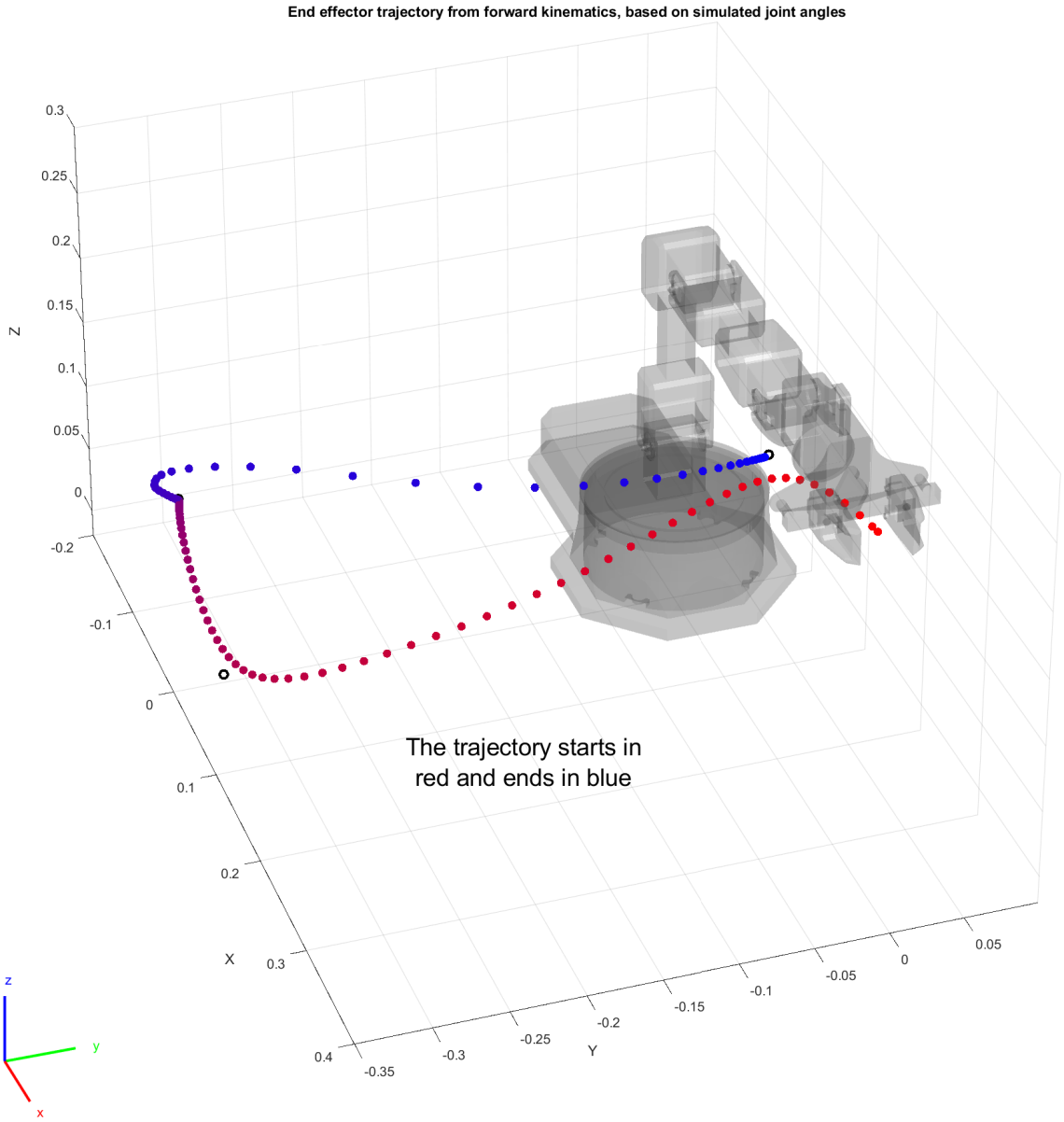


Figure 10: The calculated trajectory of the end effector with initial angles of zero for all joints.

Figure 10 also shows that waypoint two is not reached before moving on to the next waypoint. In Figure 11 the planned and calculated trajectories are compared, but the calculated graph had to be shifted 800 ms to the left because it was lagging behind the planned trajectory due to an apparent delay in the logged values. After correcting for this lag it is clear that the calculations are correct and that the trajectory may have been too fast for this manipulator.

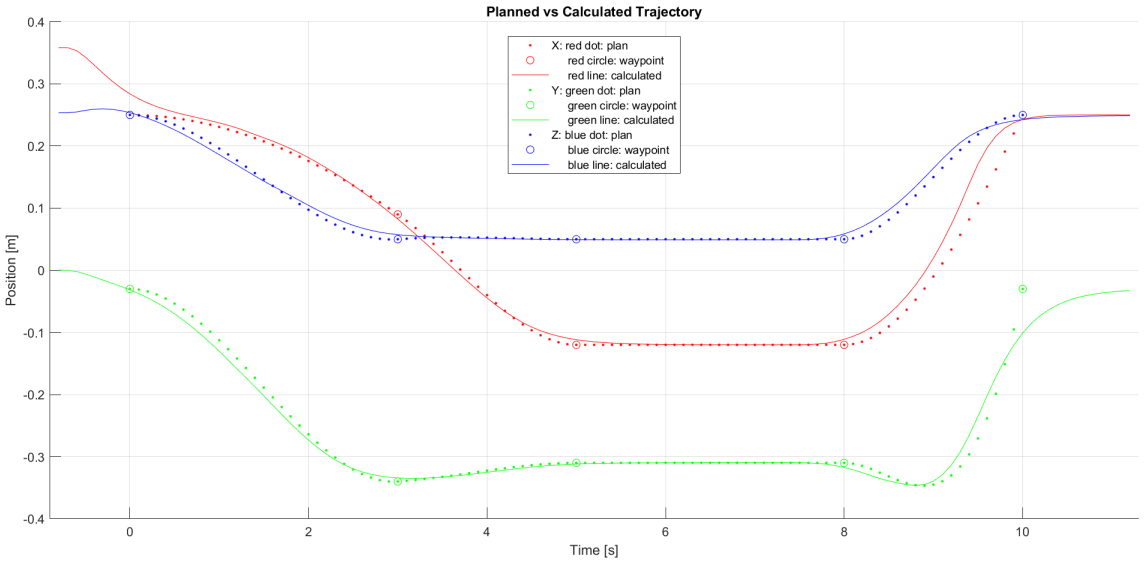


Figure 11: The planned and measured trajectory. The measured values had to be shifted to adjust for delayed in logged values.

4 Discussion

From the results of the manual control simulator, the inverse kinematic simulation and the forward kinematic simulation, the goal of creating a tool which can be used to simulate the ReactorX-150 robot arm has been satisfied.
To evaluate the usefulness of the tool further, exploring the possibility of a simulation of a robot with a visual representation of its surroundings, such as tables and boxes would be a good addition to make the simulations more easily interpreted.
There is also a possibility of acquiring a real ReactorX-150 robot arm and using the simulator to give instructions of the joint angles to the real robot and comparing its movements to the simulated robot to ensure they react in the same way.
Finally, a full project test with a real or simulated project setup and an actual comparison based on a person performing a task would allow a close to real test of the simulator and a more complete description of steps needed to use the tool set.

5 Conclusion

This report presents the methodology for simulating and analyzing a robotic manipulator using MATLAB, Simulink, and the Robotics System Toolbox. The robotic model is based on a URDF file that defines the relationships between the robot's links and joints. This manipulator includes five joints and a gripper, and its setup and integration into Simulink are described in detail.

The system incorporates a simulation environment that enables direct control and visualization of the robot's movements. The URDF file provides the geometric and dynamic data required for the manipulator, and the simulator accepts joint angle inputs to simulate the robot's trajectory. A control dashboard gives control in an intuitive way for the testing and verification of the robot model.

Trajectory planning is implemented using a 5-point waypoint mission, with the trajectory calculated using the `cubicpolytraj` function. The simulation program calculates the joint angles needed for executing the planned path, and the end effector follows the trajectory.

Forward kinematics is derived using the Denavit-Hartenberg (DH) convention. The DH parameters are calculated based on the manufacturer's specifications, adjusting for link offsets to maintain compliance with the DH rules. These parameters describe the transformations between consecutive coordinate frames, allowing the calculation of the end effector's position. For simplicity, the final transformation matrix is reduced by assuming a static fifth joint angle. The resulting expressions provide the X, Y, and Z coordinates of the end effector relative to the base. Using the joint angles that were logged during simulation in Simulink, the expressions were verified to be correct when compared to the planned trajectory.

Bibliography

[1] KUKA AG. *KUKA.Sim*. URL: https://www.kuka.com/en-us/products/robotics-systems/software/simulation-planning-optimization/kuka_sim. (accessed: 15.12.2024).

[2] ABB Ltd. *RobotStudio Suite*. URL: https://new.abb.com/products/robotics/robotstudio?PF=5785. (accessed: 15.12.2024).

[3] Mathworks. *Robotics System Toolbox*. URL: https://se.mathworks.com/help/robotics/. (accessed: 25.11.2024).

[4] MrTorstein123456789. *Simulated Movement of a Reactor-X 150 Robot Arm Using MATLAB and Simulink*. URL: https://youtu.be/ZwVZ7HUFQ2A?si=847TMsS1sKHuwoo_. (accessed: 14.12.2024).

[5] Roshan Sharma. *Group Project Description: Robot Arm*. University of South-Eastern Norway. URL: https://web01.usn.no/~roshans/cfr/downloads/group-project-description-robot-arm.pdf.

[6] Roshan Sharma. *rx150-urdf-and-geometry-files*. URL: https://web01.usn.no/~roshans/cfr/downloads/rx150-urdf-and-geometry-files.zip. (accessed: 25.11.2024).

[7] Inc. The MathWorks. *Simulink*. URL: https://www.mathworks.com/products/simulink.html. (accessed: 15.12.2024).