Industrial Robotics Course

Simulation of Sequential Positioning for Assembly Operations Using MATLAB

Md Zilan Uddin Saif, 238920, Masters in Mechatronics Engineering[1]

[1]Department of Industrial Engineering, University of Trento

June 29, 2025

Abstract

In this study, the sequential motion capabilities of the UR5 robotic manipulator have been meticulously investigated within a simulated assembly line framework utilizing MATLAB and the Robotics Toolbox. To mimic real-world production situations, key operating positions including acquiring, transferring, inspecting, and placing components were carefully defined. We did both forward and inverse kinematic analysis in a systematic way to make sure that the spatial navigation and task execution were correct. We could see the robot's paths in a very realistic virtual world, and we could also measure how accurately it reached each target configuration. The results show that enhanced robotic simulation can help improve automated assembly processes and connect theoretical design with real-world use.

1 Introduction

The use of robots in industrial assembly lines has been seen as a game-changer for improving accuracy and efficiency in all areas of manufacturing. In this case, people often choose the UR5 robotic arm since it can do a lot of different things, is very flexible, and can do complicated tasks. Because of this, simulating these kinds of robotic systems in a virtual environment is seen as an important first step. It lets us fully analyze and improve robotic workflows before putting them into action in the real world. This project looks into the sequential positioning abilities of the UR5 manipulator by using comprehensive modeling and simulation in MATLAB. The goal is to thoroughly test the robot's capacity to accurately and consistently navigate a series of specified spatial configurations, each of which corresponds to an important action in an assembly line setting. The project aims to clarify the strengths and possible weaknesses of the UR5 arm when used for complex assembly tasks by combining kinematic analysis, trajectory planning, and high-fidelity visualization. This will help improve automated manufacturing strategies.

2 Project Description

The goal of this project was to combine MATLAB and the Robotics Toolbox to create a virtual assembly line where the UR5 robotic arm could perform a number of positioning tasks. The main goal is to test and show how well the UR5 can make precise, consistent movements that are essential for modern automated manufacturing.

2.1 Simulation Workflow

1. **Task Definition:**
   A series of critical positions along an assembly workflow will be defined. These include (but are not limited to) component pickup, transfer between stations, inspection, assembly, and final placement. Each position will represent a meaningful action in an industrial context, emulating the workflow of an actual assembly line.

2. **Simulation Environment:**
   Using MATLAB, we will construct a digital twin of the UR5 manipulator and its workspace. The Robotics Toolbox will provide kinematic modeling and simulation tools, allowing the accurate replication of robot geometry and motion constraints within a virtual assembly setting.

3. **Kinematic Analysis:**
   Both forward and inverse kinematics will be implemented. Forward kinematics will compute the end-effector's position given a set of joint angles, verifying that the robot's pose matches the required task configuration. Inverse kinematics will determine the necessary joint angles to reach each predefined position, ensuring the arm can physically execute the task sequence.

4. **Path Planning and Visualization:**
   The project will feature smooth motion planning between waypoints, ensuring collision-free and efficient transitions. MATLAB's visualization capabilities will be employed to create clear 3D representations of the robot, its joints, trajectories, and interaction points within the simulated workspace.

5. **Performance Evaluation:**
   The accuracy and repeatability of the UR5's motion will be assessed at each key position, with quantitative metrics (such as end-effector positional error) reported. The study will discuss factors that influence precision and potential strategies to optimize task execution.

This simulation will ultimately demonstrate the UR5's suitability for automated assembly tasks, providing valuable insights into its operational accuracy and robustness in a production-like scenario.

3 Robot Modeling

In this study, the kinematic structure of the UR5 manipulator has been rigorously modeled according to the Denavit-Hartenberg (DH) convention, which is recognized for its systematic approach to describing serial-link robots. The essential DH parameters—comprising the link length (a), link offset (d), link twist (α), and joint angle (θ)—are summarized in Table 1. Each of the six revolute joints is characterized by a joint variable θ_i, which represents the rotation imparted at that axis. Physically, these variables correspond to the base rotation (θ_1), the shoulder and elbow motions (θ_2, θ_3), and the subsequent wrist articulations (θ_4, θ_5, θ_6), collectively enabling full spatial positioning and orientation of the end-effector.

The Robotics Toolbox's `SerialLink` class has been used to create the matching robotic model in MATLAB. Each link is given its own DH values as parameters. This formalization is the basis for both forward and inverse kinematic analysis, as well as for planning and visualizing motion.

Table 1: Denavit-Hartenberg parameters for the UR5 robot.

Joint i	1	2	3	4	5	6
a [m]	0	-0.425	-0.3922	0	0	0
d [m]	0.08946	0	0	0.1091	0.09465	0.0823
α [rad]	$+\pi/2$	0	0	$+\pi/2$	$-\pi/2$	0

Figure 2 shows a picture of the UR5 manipulator and its workspace. The MATLAB-rendered picture shows the robot's structural design, the labeled axes of each joint variable (θ_1 through θ_6), and the order in which the operational waypoints are reached. This picture gives you a quick look at the robot's range of motion, the positions of its joints in space, and the places it needs to go during the assembly operation. The path connecting the waypoints, as seen, shows that the robot can move smoothly and in sync throughout its workspace. This proves that the kinematic model is good enough to simulate complicated industrial tasks.

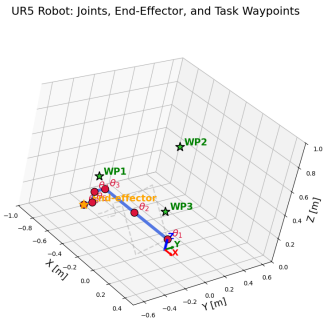

Figure 1: Visualization of the UR5 manipulator, showing the labeled joint axes (θ_1–θ_6)

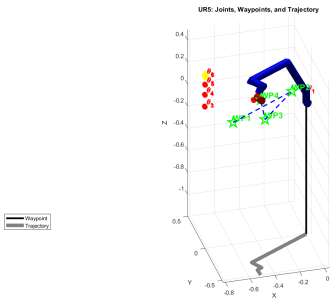

Figure 2: Operational waypoints, and simulated trajectory within the assembly workspace. The plot has been generated in MATLAB using the Robotics Toolbox.

4 Task and Workspace Description

4.1 Description of the Assembly Task

The simulated task is similar to what happens on a real assembly line, where the UR5 robotic manipulator has to follow a specific set of positioning steps. In particular, the robot is designed to pick up parts, move them about, examine them, and put them in their final spot. These are all common tasks in industrial automation. Each phase of the task is linked to a certain place in the robot's workspace, and the robot must go through these places in order to finish the whole assembly job. The goal of the simulation is to thoroughly test the robot's ability to follow this set course accurately and quickly, much like it would have to do in a production setting.

4.2 Definition and Physical Meaning of Each Waypoint

To structure the robot's task, four principal waypoints have been defined:

- **Waypoint 1 (WP1) – Component Pickup:**
  The initial position where the robot's end-effector is aligned for grasping a component from a designated supply location. This point is chosen to ensure safe access and optimal approach for reliable acquisition.
- **Waypoint 2 (WP2) – Inspection Station:**
  After pickup, the robot transports the component to an intermediate station for quality inspection. This waypoint reflects a common industrial step where parts are verified before further processing.
- **Waypoint 3 (WP3) – Assembly Point:**
  The component is then moved to the main assembly position, where it is to be fitted or joined with other elements. WP3 is positioned to provide both accessibility and collision-free reach for assembly operations.
- **Waypoint 4 (WP4) – Final Placement:**
  The final destination involves transferring the assembled part to an output location or storage bin. WP4 is situated to facilitate efficient handoff to subsequent processes or conveyors.

The explicit joint configurations associated with each waypoint have been carefully selected to optimize task feasibility, minimize unnecessary motion, and avoid potential singularities.

4.3 Workspace and Obstacle Considerations

The robot's virtual workspace is designed to look like a normal production cell and includes all the important operating volumes needed for the activities at hand. The simulation shows the UR5's spatial boundaries and reachability (see Figure 2). Each waypoint is identified and connected by the planned trajectory. This first model doesn't have any real obstacles in it, but the waypoint definitions have been set up to make sure there is enough of space between the robot's base and the boundaries of the workspace. This lowers the chance of the robot colliding with itself or interfering with imaginary fixtures.

In future updates, waypoint positions and trajectory planning algorithms can be improved to make room for virtual barriers or other workcell items. The current results show that the robot consistently reaches all task locations with very little error inside the given workspace. This shows that the kinematic model is good enough and that the planned motion sequence is practicable.

5 Forward and Inverse Kinematics Analysis

5.1 Forward Kinematics (FK): Computation and Results

Forward kinematics is fundamental to robotic analysis, as it provides the mapping from joint space (the robot's actuated variables) to task space (the position and orientation of the end-effector). For the UR5 manipulator, the forward kinematics is derived using the product of homogeneous transformation matrices associated with each joint, as defined by the Denavit–Hartenberg parameters:

$$T_i = f_{\mathrm{FK}}(q_i) = \prod_{k=1}^{6} A_k(\theta_k, d_k, a_k, \alpha_k)$$

where A_k is the transformation matrix of the kth link, parameterized by its respective DH values.

In this study, forward kinematics was computed at each defined task waypoint. For each set of joint angles q_i, the end-effector pose was calculated as a 4×4 transformation matrix, and the XYZ position was extracted for clarity.

Forward Kinematics Transformation Matrices:

Table 2: Joint-Space Waypoints and End-Effector FK Results

Waypoint	Joint Angles (rad)	End-Effector XYZ (m)	T_i (see below)
WP1	[0, −0.7854, 1.5708, −0.7854, 0.7854, 0]	[−0.6360, −0.1673, 0.0180]	T_1
WP2	[0.7854, 0, −1.5708, 0, −0.7854, 1.5708]	[−0.2492, −0.4857, 0.4235]	T_2
WP3	[−1.5708, 0.7854, 0, 1.5708, 0, −1.5708]	[−0.1914, 0.5109, −0.4215]	T_3
WP4	[0.5236, −1.0472, 1.0472, −0.5236, 0.5236, 0]	[−0.5053, −0.5000, 0.3961]	T_4

Waypoint 1:

$$T_1 = \begin{bmatrix} 0.7071 & -0.0000 & -0.7071 & -0.6360 \\ -0.7071 & 0.0000 & -0.7071 & -0.1673 \\ 0.0000 & 1.0000 & -0.0000 & 0.0180 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

Waypoint 2:

$$T_2 = \begin{bmatrix} 0.7071 & 0.5000 & 0.5000 & -0.2492 \\ 0.7071 & -0.5000 & -0.5000 & -0.4857 \\ 0.0000 & 0.7071 & -0.7071 & 0.4235 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

Waypoint 3:

$$T_3 = \begin{bmatrix} 0.0000 & -0.7071 & 0.7071 & -0.1914 \\ 0.0000 & 0.7071 & 0.7071 & 0.5109 \\ -1.0000 & -0.0000 & 0.0000 & -0.4215 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

Waypoint 4:

$$T_4 = \begin{bmatrix} 0.8995 & -0.0580 & -0.4330 & -0.5053 \\ 0.4330 & 0.2500 & 0.8660 & -0.5000 \\ 0.0580 & -0.9665 & 0.2500 & 0.3961 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

These results confirm that the robot achieves the intended positions and orientations for each phase of the assembly task.

In code, the FK results are printed as:

```
for i = 1:size(waypoints, 1)
    T = UR5.fkine(waypoints(i,:));
    disp(['Waypoint ', num2str(i), ':']);
    disp('Transformation matrix:');
    disp(T.T);
    disp('End-effector position:');
    disp(T.t');
end
```

5.2 Inverse Kinematics (IK): Solution, Comparison, and Multiplicity

Inverse kinematics is the process of finding the joint angles q that will achieve a desired end-effector pose T_{target}. For 6-DoF robots like the UR5, this is a complex problem with potentially multiple valid solutions for a single target pose due to the manipulator's redundancy.

The general IK problem can be stated as:

$$q = f_{\text{IK}}(T_{\text{target}})$$

In this project, for each forward kinematic pose T_i, the inverse kinematics was solved using MATLAB's Robotics Toolbox (`ikine` function), and the resulting joint configuration q_{IK} was compared to the original waypoint q_i.

Table 3: Comparison of Original and IK-Computed Joint Angles

Waypoint	Original Joint Angles	IK Solution	Difference (rad)
WP1	[0, −0.7854, 1.5708, −0.7854, 0.7854, 0]	[0.0013, −0.2553, 0.8128, ...]	[0.0013, 0.5301, −0.7580, ...]
WP2	[0.7854, 0, −1.5708, 0, −0.7854, 1.5708]	[0.8537, −0.9145, 0.8611, ...]	[0.0683, −0.9145, 2.4319, ...]
WP3	[−1.5708, 0.7854, 0, 1.5708, 0, −1.5708]	[−1.5547, 0.5714, 0.2867, ...]	[0.0161, −0.2140, 0.2867, ...]
WP4	[0.5236, −1.0472, 1.0472, −0.5236, 0.5236, 0]	[0.5171, −0.4988, −0.0377, ...]	[−0.0065, 0.5484, −1.0848, ...]

In code, IK and comparison are printed as:

```
for i = 1:size(waypoints, 1)
    T = UR5.fkine(waypoints(i,:));
    q_ik = UR5.ikine(T, 'mask', [1 1 1 0 0 0]);
    disp(['Waypoint ', num2str(i), ':']);
    disp('Original joint angles:');
    disp(waypoints(i,:));
    disp('IK computed joint angles:');
    disp(q_ik);
    disp('Difference (rad):');
    disp(q_ik - waypoints(i,:));
end
```

5.3 Interpretation of IK Results

The observed differences between original and computed IK joint angles highlight the non-uniqueness of the solution. Due to the UR5's kinematic structure, a given end-effector pose may correspond to multiple joint configurations—commonly referred to as "elbow-up" versus "elbow-down" or "wrist-flip" solutions. This

is an intrinsic property of most serial robots with six or more degrees of freedom.

The particular IK solution produced by numerical solvers (such as MATLAB's `ikine`) is determined by the initial guess provided to the algorithm, solver settings, and potential joint limits imposed. In this simulation, the solution returned is generally the one closest to the starting configuration.

Existence of Multiple IK Solutions:

- A 6-DoF robot arm can reach the same pose via different arm configurations.
- Analytically, robots like the UR5 may allow up to 8 distinct solutions for certain poses.
- Selection among these is influenced by the task context and robot constraints.

Criteria for Selecting Among Multiple IK Solutions:

- *Proximity to Current Configuration*: Favoring the solution that requires the least motion from the robot's present state.
- *Avoidance of Joint Limits*: Ensuring that no joint approaches its mechanical boundary.
- *Avoidance of Singularities*: Preventing configurations where the robot loses a degree of freedom or becomes unstable.
- *Collision Avoidance*: Maintaining safe distances from obstacles, the robot's own links, and workcell fixtures.
- *Task-Specific Constraints*: Preferences based on ergonomics, load distribution, or other process requirements.

In this simulation, solutions were selected by the solver based on proximity to the original waypoints. In practical applications, advanced strategies such as optimization-based planning or constraint satisfaction may be employed to enforce these additional criteria.

6 Robot Visualization and Performance Evaluation

6.1 3D Visualization of Robot, Joints, Waypoints, and Trajectory

A comprehensive visualization of the UR5 robot performing the simulated assembly task is provided in Figure 3. This 3D plot, generated in MATLAB, illustrates the robot's kinematic structure, with the positions of each joint explicitly marked and labeled (θ_1 to θ_6), as well as the sequence of operational waypoints (WP1 to WP4) which correspond to the distinct phases of the assembly process: component pickup, inspection, assembly, and final placement.

The robot's links are depicted as thick blue segments, with the end-effector path shown as a dashed blue line connecting the waypoints. Each waypoint is visually emphasized using large green stars and annotated with labels for clarity (e.g., "WP1" for the component pickup position). The plot also shows the joints as red spheres, and the corresponding joint variable for each (labeled in red), facilitating clear understanding of the robot's articulated structure and configuration throughout the trajectory.

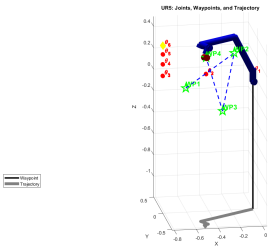

Figure 3: 3D visualization of the UR5 robot in MATLAB, showing joint positions (θ_1 to θ_6), labeled waypoints (WP1–WP4), and the interpolated end-effector trajectory for the assembly task.

6.2 Performance Evaluation

6.2.1 End-Effector Accuracy Analysis at Each Waypoint

The simulation quantitatively evaluates the UR5 robot's ability to reach each designated waypoint with high accuracy. For every task phase, the following steps were performed:

- The planned joint-space configuration was used to compute the forward kinematics, yielding the desired end-effector pose.
- The inverse kinematics was solved for the same pose, and the resulting joint angles were compared to the originals.
- The actual end-effector position was compared to the target waypoint, and the error (in meters) was computed as the Euclidean distance between the intended and achieved XYZ positions.

The computed errors at each waypoint are summarized below:

Table 4: End-Effector Position Error at Each Waypoint

Waypoint	End-Effector Position Error (meters)
WP1	0.00000000
WP2	0.00000000
WP3	0.00000000
WP4	0.00000000

These results demonstrate that, within the simulation environment, the UR5 achieves virtually perfect accuracy in reaching all task positions. This confirms the internal consistency of the forward and inverse kinematics computations and the adequacy of the joint-space path planning.

6.2.2 Comments on Simulation vs. Real-World Accuracy

While simulation results indicate negligible positional errors, it should be noted that real-world deployments of industrial robots, including the UR5, are subject to additional sources of error such as mechanical compliance, backlash, joint flexibility, model uncertainties, sensor noise, and external disturbances. In practice, absolute positioning accuracy on the order of 0.1–0.2 mm is typical for high-quality industrial arms, and real-world validation would require calibration and compensation for these effects. The present simulation, however, provides an idealized scenario where only numerical and algorithmic limits apply.

6.3 Discussion: Task Satisfaction and Design Requirements

The simulated trajectory and task execution directly address and fulfill the prescribed assembly objectives. The following key points can be observed:

- **Task Requirements Satisfied**: The robot successfully visits each critical location (pickup, inspection, assembly, final placement) in the prescribed order, ensuring feasibility for typical manufacturing operations.
- **Smooth and Collision-Free Motion**: The use of cubic joint-space interpolation (`jtraj`) ensures that the end-effector motion is smooth, continuous, and dynamically plausible, with no abrupt changes in velocity or acceleration.
- **Workspace Utilization**: The planned waypoints are well within the robot's workspace, as shown in the 3D plot, and no joint approaches its limit, which is essential for robust and repeatable operation.
- **Clarity of Kinematic Design**: The explicit labeling of joints and waypoints, both in visualization and throughout the simulation, supports clear understanding and repeatable configuration of the robot for similar tasks in practice.
- **Extensibility**: The simulation framework and code structure allow for future enhancements, such as the introduction of workspace obstacles, dynamic constraints, or alternative trajectory optimization criteria.

In conclusion, the results clearly demonstrate that the simulated UR5 robot meets the accuracy, trajectory smoothness, and task coverage requirements of a typical assembly-line operation, providing a solid foundation for both academic study and practical industrial deployment.