ME 4451 Lab 3 Handout
3RRR Inverse Kinematics

Anirban Mazumdar, Bryan Blaise, Nikhil Nandikanti

(1 Session Lab)

Fall 2023

1 Objectives

1. Use inverse kinematics to control the motion of 3RRR parallel robots.
2. Measure accuracy and repeat-ability.
3. Use the 3RRR parallel robot to draw a line.

2 Prelab Assignment

- Read Lab 3 Handout.
- Be prepared to identify/point to the following variables on the physical robot. The TA will stop by at the beginning of the lab to ask you the following questions
  - Identify which of the three "virtual" robots each of the motors corresponds to (aka which is motor 1, 2, vs 3). Do this by disabling the torque, moving link 1 by hand, and reading the motors' position to see which changes accordingly
  - Point to the angles represented by β_1, β_2, and β_3 and determine their values
  - With a **dry-erase marker**, draw the vector representing the end-effector's orientation
  - With a **dry-erase marker**, draw a line between the two points that define the distance a_{33}
  - With a **dry-erase marker**, draw the arcs representing the angles δ_{e2} and ϕ_{e2}
- Create a Matlab script for Lab 2 tasks (Sec. 5-8). This includes but is not limited to:
  1. Updating *data_RRR3* file
  2. Updating *RRR_go_configuration* file, calling required functions *RRR3_reverse* and *RRR_go_angles*
  3. Updating *RRR3_draw_shape* function accordingly

3 Coordinate System

Fig. 1 shows the end-effector position (x_e, y_e) and orientation (ϕ_e), and link lengths (a_{ij}). Furthermore, each joint angle (θ_{ij}) is defined positive in the counter-clockwise direction. The end-effector origin is at the center of the aluminum base plate.

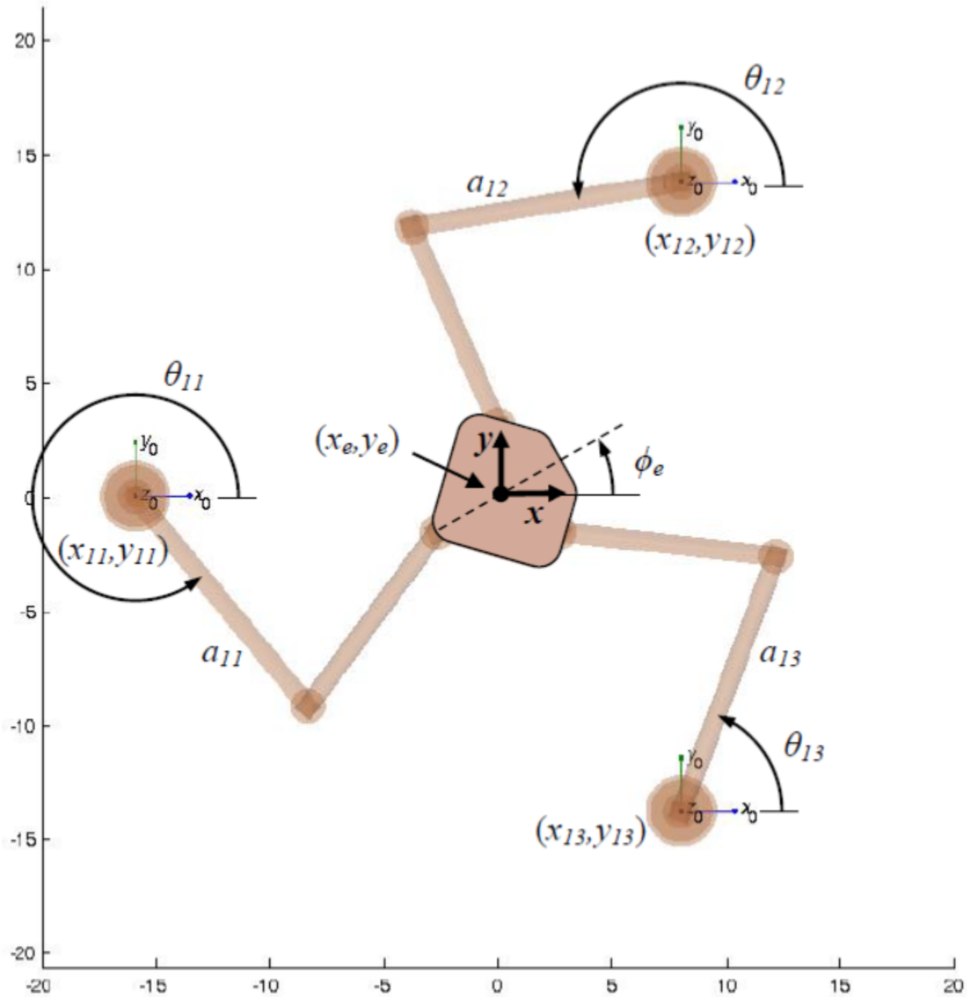

Figure 1: Coordinate frame for the 3RRR parallel robot.

4 Instructions and Tasks for This Lab

The first step in this lab is to test the inverse kinematics algorithm for the 3RRR parallel robot. You will use the parameters from *data_RRR3* along with the *RRR3_reverse* and *RRR_go_angles* functions to write *RRR3_go_configuration* for the 3RRR parallel robot. You will make accuracy and repeatability measurements similar to those in Lab 1 and draw a line similar to Lab 2.

5 RRR Inverse Kinematics Function

5.1 Offset Angles

The offset angles are input via the vector *dh.delta* in *data_RRR3*. The offset angles are the angles from the virtual links, a_{3j}, to the actual end effector orientation, ϕ_e. This is shown in Fig. 2. Also, recall the equations below to find these angles.

$$\phi_e = \delta ej + \phi_{ej} \tag{1}$$

$$\begin{bmatrix} \delta_{e1} & \delta_{e2} & \delta_{e3} \end{bmatrix} \tag{2}$$

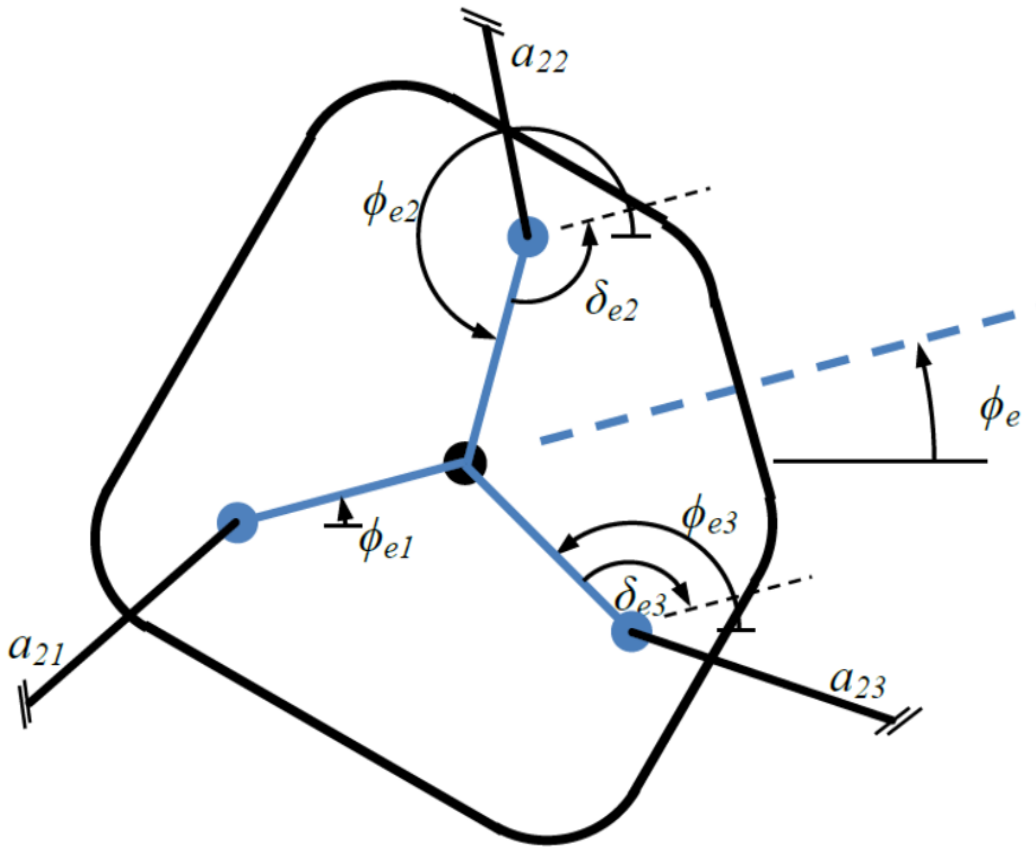

Figure 2: Virtual links and offset angles.

5.2 Elbow Configuration

In this lab, always use the positive elbow angle configuration, *elbowplus = 1*. This is set in the *RRR3_draw_shape* script that is provided.

5.3 Base Locations

The virtual serial robot base locations (in cm) are:

$$posx = \begin{bmatrix} -15.906 & 7.953 & 7.953 \end{bmatrix} \tag{3}$$

$$posy = \begin{bmatrix} 0 & 13.775 & -13.775 \end{bmatrix} \tag{4}$$

These locations are included in the *data_RRR3* structure.

5.4 Virtual Robot to Servo Rotation

Figure 3 shows how joint angles, θ_{ij} are measured from the positive x axis of the end-effector coordinate system. However, recall that on a Dynamixel servo, the angle is measured from the centerline of the servo. Therefore, the angles calculated by your inverse kinematics code need to be offset accordingly. The angles, θ'_{ij} will be the angles sent to the servos. These are found using the following expression.

$$\theta'_{ij} = \theta_{ij} - \beta_j \tag{5}$$

6 3RRR Robot Control

Before you can complete the *RRR3_go_congfiguration* function, you need to do the following.

- Set new servo angle constraints to [-90, 90] since the parallel structure limits the movement of joint 1 (update the *RRR_go_angles* file).
- Determine offset angles δ_{ej} for *dh.delta*, as shown in Fig. 2.
- Determine servo coordinate rotation angles (*dh.beta*).

Complete the function *RRR3_go_configuration(dh, displ, elbowplus)* that calls the following functions: *RRR3_reverse* (reverse kinematics) and *RRR_go_angles*. You also need to do the following in the *RRR3_go_configuration* function:

- Extract robot parameters (a, delta, etc.) from the *dh* structure.
- Calculate θ'_{ij} using *dh.beta* as shown in Fig. 3.
- Constrain the calculated angles to between $\pm 180°$.
- Utilize the sample code in *RRR3_reverse* to complete the function by extending it for virtual robots 2 and 3.

Test your function first **without commanding the servos** by commenting out the *RRR_go_angles* line. Verify that the angles are calculated correctly. Try a few different configurations. Show the TA your calculated angles from *RRR3_go_configuration* for *displ = [0, 0, 0]* and *displ = [0, 4, 0]*.

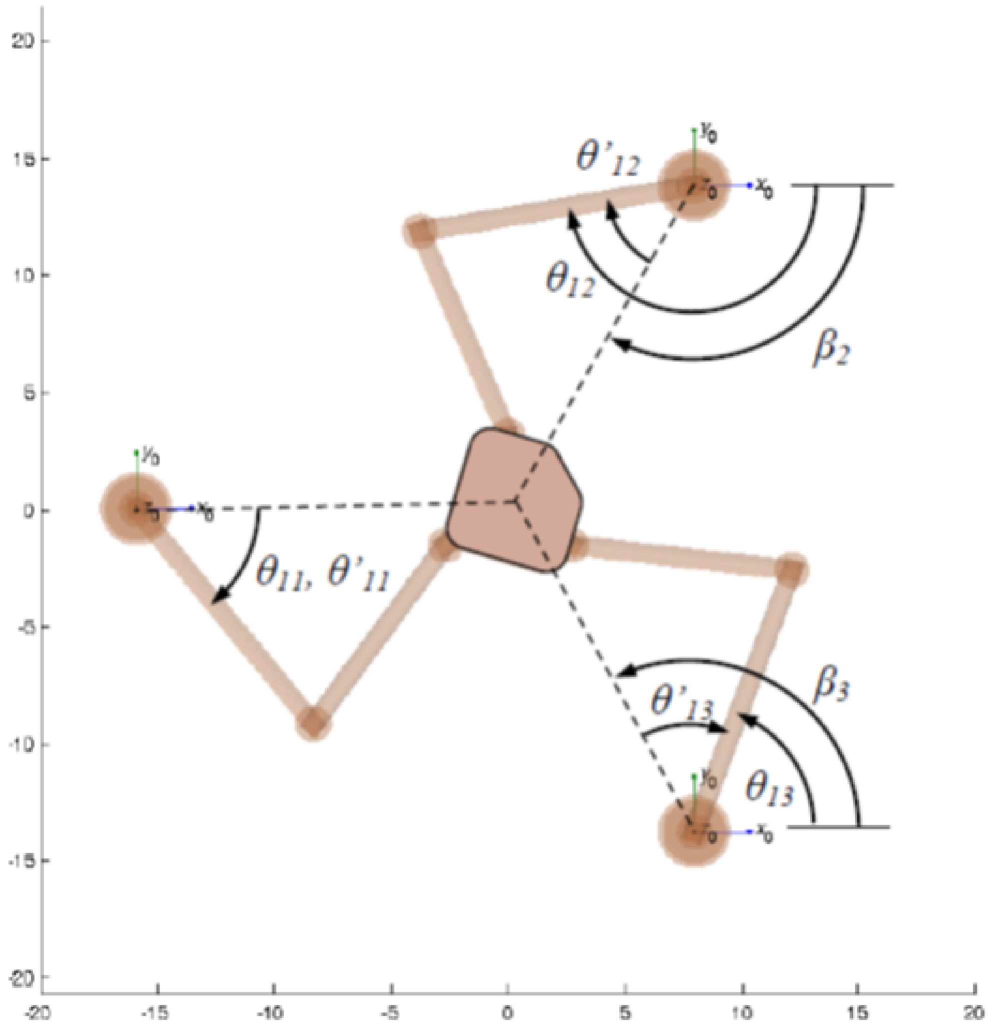

Figure 3: Servo coordinate rotation angles. **The dotted lines are pointing from the motor/joint axis to the origin, not the end-effector.** The end-effector just happens to be located near the origin in this figure.

7 Accuracy and Precision

As in the previous Labs, you will test the accuracy and precision of the end-effector position. This time however, you will do so using the end-effector position instead of sets of joint angles. In addition, all measurements will come from a "star" pattern. Use your repeat-ability script from Lab 1 (that drew the star shape), but make the following changes:

- Open the connection using *dxl_SerialOpen_RRR3* instead.
- Use *RRR3_go_configuration* for the move commands.
- use *displ = [0, 0, -15]* as the starting point

- Move to each of the following points, and return to the starting position in between each one:
- $[0, 6, -15]$
- $[4, 1, -15]$
- $[3, -5, -15]$
- $[-3, -4, -15]$
- $[-6, 2, -15]$

Pick three of these points to measure point to point accuracy:

- Measure the distance between points.
- Compare with calculated nominal distance

Record your measurements in the table below.

Point	Measured	Desired	% Error
1			
2			
3			

1. How accurate is the parallel robot? What is the average percent error?
2. Now draw a circle that encloses all the starting points and measure the diameter of the circle.
3. What does this tell you about the precision of this robot?

Upon completion, check results off with TA.

8 Path Motion

8.1 Line Motion

Uses the function *path_line(displ1, displ2, N)* from Lab 2 that returns a $(N + 1)$ x 3 matrix of points and orientations along a line. Call this function inside your *RRR3_draw_shape* function. This function should also allow rotation in place.

Test your program with a line from:

- $[3, 3, 0]$ to $[-1, 5, 0]$ with $N = 15$
- $[0, 0, 10]$ to $[0, 0, -60]$ with $N = 10$ (rotation in place)

When you are done with both line programs and have been checked off by the TA, close the serial connection and disconnect the robot.