Visualization and Forward Kinematics of a 2-DOF Planar Robotic Arm

Joel John Mathew
240499

December 18, 2025

1 Introduction

This document presents the forward kinematics and visualization of a planar robotic arm with two revolute joints and two rigid links of equal length ($l_1 = l_2 = 1$). The arm is fixed at the origin, and its configuration is defined by joint angles q_1 and q_2. The objective is to compute joint positions and visualize the arm for multiple configurations.

2 Forward Kinematics

The forward kinematics equations for the 2-DOF planar arm are given by:

$$x = l_1 \cos(q_1) + l_2 \cos(q_1 + q_2) \tag{1}$$
$$y = l_1 \sin(q_1) + l_2 \sin(q_1 + q_2) \tag{2}$$

The elbow joint position is computed as:

$$x_e = l_1 \cos(q_1) \tag{3}$$
$$y_e = l_1 \sin(q_1) \tag{4}$$

These equations are used to compute the Cartesian coordinates of the elbow and end-effector for any given configuration.

3 Visualization Method

The robotic arm is visualized in 2D by plotting:

- The base at the origin $(0, 0)$
- A line segment from base to elbow (first link)
- A line segment from elbow to end-effector (second link)

Matplotlib is used for plotting, with equal axis scaling to preserve geometric correctness.

4 Arm Configurations

The arm is visualized for three representative configurations:

4.1 Straight Configuration

In this configuration, both links are aligned, producing the maximum reach. (reach is l1 + l2, here 2)

4.2 Bent Elbow Configuration

Here, the elbow joint is visibly bent, resulting in a reduced reach and a change in workspace orientation.

4.3 Folded Configuration

In the folded configuration, the arm bends back toward the base, significantly reducing the end-effector distance from the origin.

5 Results

Figures below show the plotted configurations of the robotic arm.

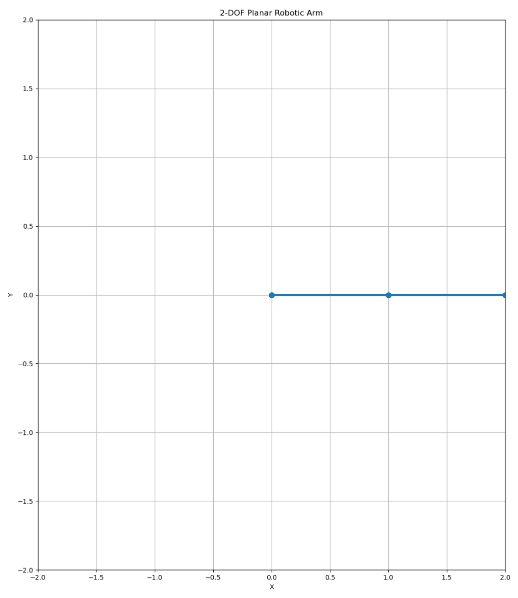

Figure 1: Straight arm configuration

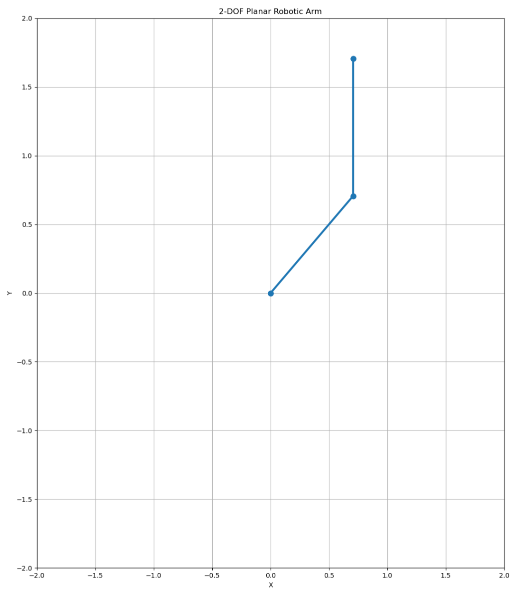

Figure 2: Bent elbow configuration

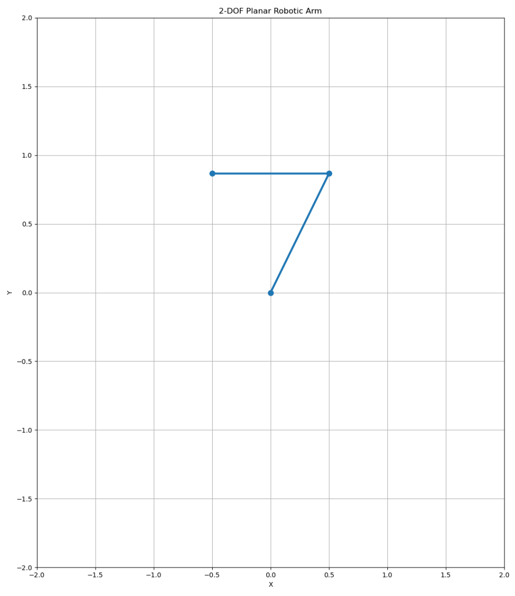

Figure 3: Folded arm configuration

6 Code Implementation

The Python implementation used to compute forward kinematics and generate the plots is available in the accompanying GitHub repository submitted as part of this assignment.

`RoboticArm.py`

```
from math import cos, sin

class Arm:
    def __init__(self, l1, l2):
        self.l1 = l1
        self.l2 = l2
        self.q1 = 0
        self.q2 = 0
        self.end_ = (self.l1+self.l2, 0) # intial end effector
        self.elb_ = (self.l1, 0) # initial elbow

    def f_kin(self, q1, q2):
        self.q1 = q1
        self.q2 = q2
        x = self.l1 * cos(q1) + self.l2 * cos(q1 + q2)
        y = self.l1 * sin(q1) + self.l2 * sin(q1 + q2)
        self.end_ = (x, y)
        self.elb_ = (self.l1 * cos(q1), self.l1 * sin(q1))
        return self.end_

    def get_points(self):
        # Return list of points: base, elbow, end effector.
        return (0.0, 0.0), self.elb_, self.end_
```

plotArm.py

```
from RoboticArm import Arm
import matplotlib.pyplot as plt
from math import pi

angles = [(0, 0), (pi/4, pi/4), (pi/2, -pi/4), (pi/3, 2*pi/3)]


def plot_arm(arm):
    base, elbow, end = arm.get_points()

    x = [base[0], elbow[0], end[0]]
    y = [base[1], elbow[1], end[1]]

    plt.figure(figsize=(3, 3))
    plt.plot(x, y, "-o", linewidth=3, markersize=8)

    # Keep proportions correct
    reach = arm.l1 + arm.l2
    plt.xlim(-reach, reach)
    plt.ylim(-reach, reach)
    # plt.gca().set_aspect("equal", adjustable="box")

    plt.grid(True)
    plt.xlabel("X")
    plt.ylabel("Y")
    plt.title("2-DOF Planar Robotic Arm")
    plt.show()



arm = Arm(l1=1.0, l2=1.0)

for angle in angles:
    q1, q2 = angle
    arm.f_kin(q1, q2)
    plot_arm(arm)
```