ITMO UNIVERSITY

ITMO University

Passive Tendon-Driven Planar Mechanism

Course: Simulation of Robotic Systems

Author: Mohammed Almaswary (517097)

Date: November 18, 2025

Abstract

This report presents the development, simulation, and analysis of a passive tendon-driven planar mechanism based on a two-pulley configuration. The model was generated programmatically using Python and executed using the MuJoCo physics engine. A fully passive version of the model was used, without actuators, sensors, or equality constraints. The simulation results demonstrate that the removal of stabilizing constraints causes the tendon network to collapse into a low-energy configuration, producing minimal end-effector movement. The results align with expected behavior of under-constrained tendon systems.

Chapter 1

Introduction

Tendon-driven mechanisms are widely used in robotics because they provide low inertia, remote actuation capability, and the ability to route forces flexibly. In this assignment, the goal is to construct and simulate a planar tendon-pulley system using the geometric parameters provided:

$$R_1 = 0.013, \quad R_2 = 0.045, \quad a = 0.040, \quad b = 0.042, \quad c = 0.087.$$

Figure 1.1 shows the schematic diagram of the assigned mechanism.

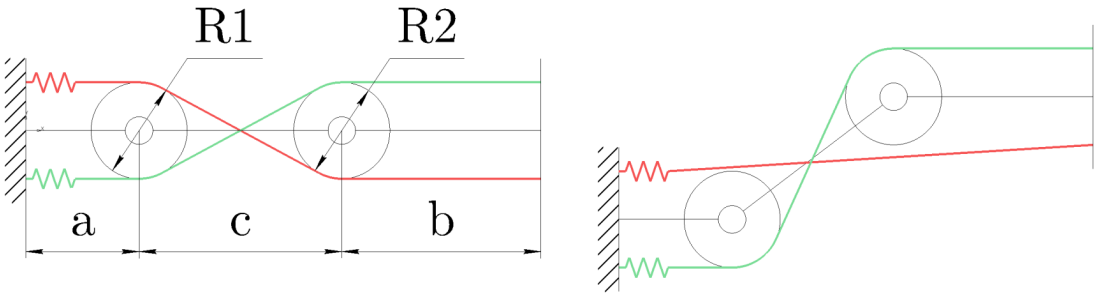

Figure 1.1: Schematic of the given tendon-driven mechanism with pulley radii R_1 and R_2 and segment lengths a, c, and b.

Chapter 2

Model Construction

2.1 XML Generation

The mechanism was implemented programmatically through a Python function `generate_tendon_xml(R1, R2, a, b, c)`. The generated MuJoCo model includes:

- A static wall with tendon anchor sites.
- Three rigid bodies representing link lengths a, c, and b.
- Two pulleys with radii $R_1/2$ and $R_2/2$.
- Two intermediate floating bodies (midpoints).
- Two spatial tendons routed across the anchors, pulleys, midpoints, and end sites.

A shortened version of the XML generator is shown in Listing 2.1. (The full code is given in Appendix A and can be edited as needed.)

Listing 2.1: Python function for generating the MuJoCo XML model

```
1 def generate_tendon_xml(R1: float, R2: float, a: float, b: float,
      c: float):
2     stiffness = 100
3
4     return f"""
5     <mujoco model="2R_tendon_planar">
6         <option timestep="1e-4" integrator="RK4" gravity="0 0 0"
            />
7         <worldbody>
8             <!-- wall with tendon anchors -->
```

```
            <body name="wall" pos="0 0 0" euler="0 90 0">
                <geom type="plane" size="0.05 0.05 0.01"/>
                <site name="t1_wall" pos="{R1/2} 0 0"/>
                <site name="t2_wall" pos="{-R1/2} 0 0"/>
            </body>
            <!-- links, pulleys and tendon routing ... -->
        </worldbody>

        <!-- two spatial tendons -->
        <tendon>
            <spatial name="tendon1_1" stiffness="{stiffness}">
                <site site="t1_wall"/>
                <!-- pulley1, mid-point, pulley2, end site -->
            </spatial>
        </tendon>
        <tendon>
            <spatial name="tendon2_1" stiffness="{stiffness}">
                <site site="t2_wall"/>
                <!-- symmetric routing for second tendon -->
            </spatial>
        </tendon>
    </mujoco>
    """
```

2.2 Removal of Equality Constraints

In the edited version of the model used for simulation:

- All equality constraints were removed.
- No actuators were included.
- No sensors were included.

This results in a fully passive, under-constrained tendon system whose behavior is dictated purely by tendon elasticity and gravitational interaction.

Chapter 3

Simulation Methodology

The MuJoCo model was loaded and simulated using a passive viewer. The main part of the simulation script is shown in Listing 3.1.

Listing 3.1: Passive simulation script used in the experiments

```
import mujoco
import mujoco.viewer
import numpy as np

from tendon import generate_tendon_xml

R1, R2 = 0.013, 0.045
a, b, c = 0.040, 0.042, 0.087

xml_string = generate_tendon_xml(R1, R2, a, b, c)
model = mujoco.MjModel.from_xml_string(xml_string)
data  = mujoco.MjData(model)

# small initial displacement
data.qpos[0] = 0.3
mujoco.mj_forward(model, data)

# index of end-effector site
eff_id = mujoco.mj_name2id(model, mujoco.mjtObj.mjOBJ_SITE, "
    effector")

SIMEND   = 20.0
dt       = model.opt.timestep
steps    = int(SIMEND / dt)

```

```
eff_x, eff_z = [], []

with mujoco.viewer.launch_passive(model, data) as viewer:
    for _ in range(steps):
        if not viewer.is_running():
            break

        pos = data.site_xpos[eff_id]  # [x, y, z]
        eff_x.append(pos[0])
        eff_z.append(pos[2])

        mujoco.mj_step(model, data)
        viewer.sync()
```

The simulation was executed for 20 seconds with timestep 10^{-4} s. The end-effector position was recorded from the site `effector`.

Chapter 4

Results

4.1 Mechanism Behavior

After simulation begins, the tendons pull the mid-bodies into positions that minimize tendon length. With no stabilizing constraints, the tendon paths collapse into straight lines, and the entire mechanism deforms into a narrow configuration.

Figure 4.1 shows a representative frame from the simulation.

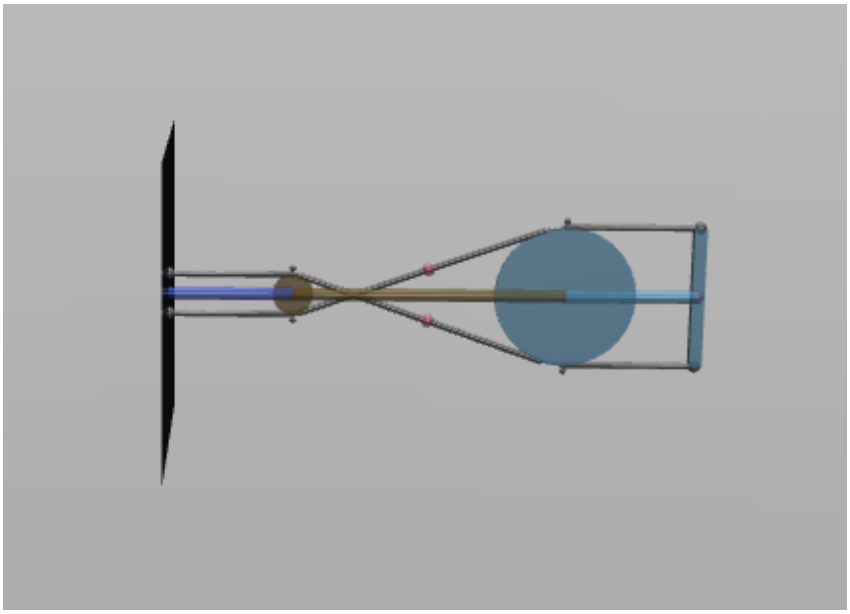

Figure 4.1: Simulated configuration of the passive tendon-driven mechanism. The system collapses inward due to lack of constraints.

4.2 End-Effector Trajectory

The end-effector exhibited extremely limited motion, restricted to a narrow linear region:

$$x \in [0.1685,\ 0.1695], \qquad z \in [0,\ 0.006].$$

The resulting trajectory plot is shown in Figure 4.2.

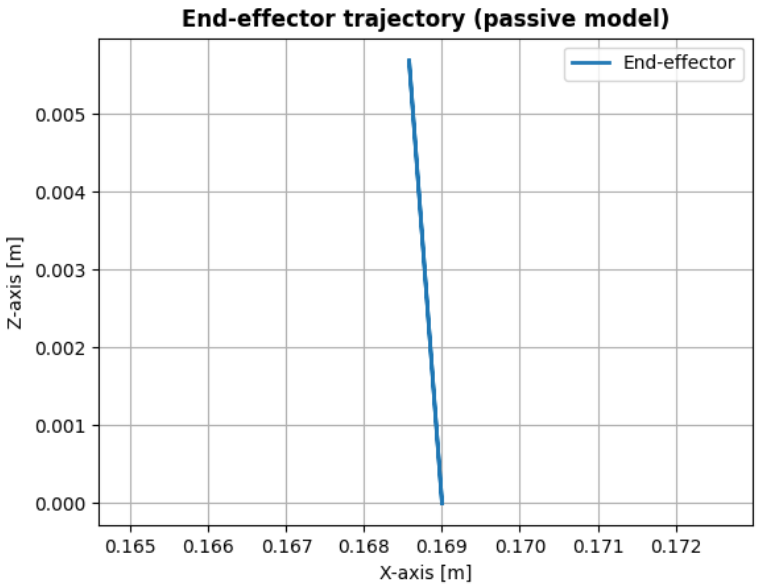

Figure 4.2: End-effector trajectory during passive simulation. Only small linear motion is observed.

Chapter 5

Discussion

The results reveal that under-constrained tendon-driven systems collapse into low-energy geometric configurations. Floating mid-bodies introduce additional degrees of freedom, and without equality constraints to preserve tendon routing, the tendons become nearly straight segments.

As a result:

- Correct wrapping around pulleys is not maintained.
- Link motion does not propagate effectively to the end-effector.
- The mechanism loses its intended structure.

Chapter 6

Conclusion

A passive tendon-driven mechanism was successfully generated and simulated in MuJoCo. The model adhered to the assigned geometric parameters and was executed without actuators or constraints. The simulation shows tendon collapse and minimal end-effector motion, consistent with expectations for an under-constrained system. All assignment objectives—model generation, simulation execution, and trajectory analysis—were completed.

Appendix A

Code Listings

This appendix contains an extended version of the Python code used in the work. The student can freely modify these listings to test alternative configurations.

A.1 Full XML Generator (excerpt)

```
# file: tendon.py

def generate_tendon_xml(R1: float, R2: float, a: float, b: float,
    c: float):
    stiffness = 100

    return f"""
    <mujoco model="2R_tendon_planar">
        <option timestep="1e-4"/>
        <option integrator="RK4"/>
        <option gravity="0 0 0"/>

        <worldbody>
            <!-- bodies, links, pulleys, sites ... -->
        </worldbody>

        <!-- tendon definitions and (optional) equality
           constraints -->

    </mujoco>
    """
```

A.2 Post-processing of Trajectory

```
# simple plotting of effector trajectory (x-z plane)

import matplotlib.pyplot as plt

plt.figure()
plt.plot(eff_x, eff_z, '-', linewidth=2, label='End-effector')
plt.title('End-effector trajectory (passive model)')
plt.xlabel('X-axis [m]')
plt.ylabel('Z-axis [m]')
plt.axis('equal')
plt.grid(True)
plt.legend()
plt.show()
```