ITMO University

Faculty of Control Systems and Robotics

REPORT

For the Subject:

"SIMULATION OF ROBOTICS SYSTEMS"

Topic:

Simulation of RR Mechanism with Tendons

Student:

ISU No. 476937– Nagham Sleman

Tutor:

Professor – Borisov, Ivan

Assistant – Rakshin, Egor

Date: 14.11.2024

1. Introduction:

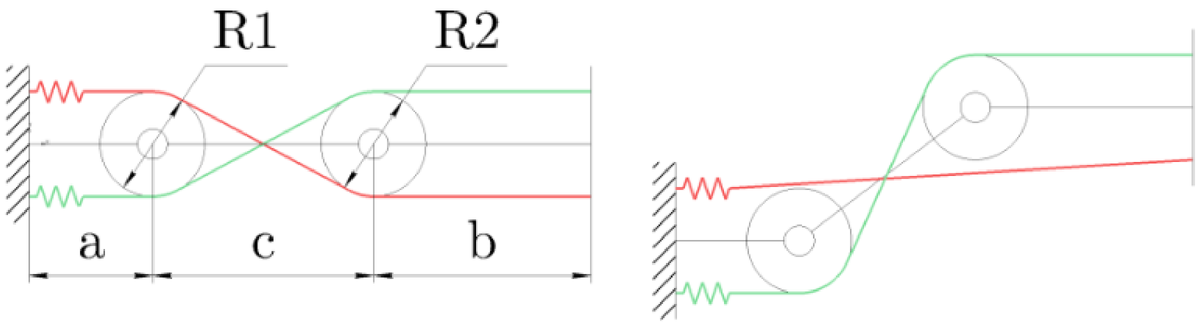
Tendon-driven robotic mechanisms are widely used in lightweight manipulators, prosthetics, and bio-inspired systems. Instead of applying torques directly at the joints, tensions are transmitted through routed tendons passing over pulleys. This allows passive motion generation and mechanically compliant behavior.

In this project, a **2-degree-of-freedom (2R) planar mechanism** is modeled using **MuJoCo**. The mechanism includes two rotational joints (R–R) actuated passively through two tendons routed through pulleys. The goal is to build the XML model, run a simulation in Python, and analyze the behavior of the tendon routing.

The goal:

From the table I have:

		R1(m)	R2(m)	a(m)	b(m)	c(m)
476937	TENDON	0.031	0.042	0.094	0.088	0.063



Note:

I encountered a problem in the tendon routing, since the distance ccc between the pulley centers was smaller than the sum of their radii $(r1 + r2)$ **to resolve** this, I modified the pulley radii so that the condition is satisfied and the geometry becomes physically valid.

So, the Parameters are:

		R1(m)	R2(m)	a(m)	b(m)	c(m)
476937	TENDON	0.014	0.02	0.094	0.088	0.063

Now we have to:

1. model .xml file for **tendon connected 2R planar mechanism**.
2. Write **python** script with model, data and viewer methods. Run the simulation.

2.Model Description:

2.1 Mechanical Structure:

The mechanism consists of:

- A fixed left base.
- First link attached via the elbow joint.
- Second link attached via the wrist joint.
- Two cylindrical pulleys:
    - One on the first link.
    - One on the second link.
- A right fixed boundary for tendon anchoring.

Gravity is disabled to prevent undesired falling or drifting and to focus purely on tendon mechanics.

2.2 Links and Joints

- Both joints use a single rotational axis:
  `axis = "0 1 0"` → rotation in the sagittal (2D) plane.
- A small amount of damping is added to prevent oscillations.
- Links are represented using capsule geoms for slender-rod visualization.

3. Tendon Routing Mechanism

Two tendons are defined: red tendon and green tendon.

3.1 Red Tendon Path

Route:

1. Site s1 (left anchor)
2. Pulley (first link), with side site s3
3. Site s3
4. Pulley2 (second link), with side site s6
5. Site s8 (right end)

Effect:
Pulling the red tendon tends to rotate the mechanism toward the **upper direction**.

3.2 Green Tendon Path

Route:

1. Site s2 (left anchor)
2. Pulley, side site s4
3. Site s4
4. Pulley2, side site s5
5. Site s7 (right end)

Effect:
Pulling the green tendon rotates the mechanism in the **lower direction**.

3.3 Tendon Properties

- **Stiffness = 10**
  → Tension increases as the tendon shortens.
- **Width = 0.0015**
- Unique color for visualization (red/green).

This creates a passive system where joint angles depend on tendon geometry rather than direct torque.

4. Simulation in Python

The simulation uses:

- mujoco.MjModel to load the XML model
- mujoco.viewer.launch_passive() to open an interactive viewer
- Camera adjustments for optimal visualization of the 2-link planar structure

The viewer allows manual joint interaction using MuJoCo's GUI, letting you observe tendon tension changes in real time.

5. Codes:

The provided **XML** and **Python** code form a reliable base for further development and experiments.

1. XML Model

```
<mujoco model="RR_Mechanism">

    <option gravity="0 0 0" integrator="Euler"/>

    <statistic center="0 0 0" extent="0.2"/>

    <visual>
        <rgba haze=".8 .3 .3 1"/>
    </visual>

    <default>
        <joint axis="0 1 0" damping="0.000005"/>
        <geom type="capsule"/>
    </default>

    <asset>
        <texture name="texplane" type="2d" builtin="checker"
                 rgb1=".7 .7 .25" rgb2=".25 .7 .7"
                 width="512" height="512" mark="cross" markrgb=".8 .8 .8"/>
        <material name="matplane" reflectance="0" texture="texplane"
                  texrepeat="1 1" texuniform="true"/>
    </asset>

    <worldbody>

        <light pos="0 0 0.25" />
        <light pos="0 0 3" dir="0 0 -1" directional="false" />

        <geom name="floor" pos="0 0 -0.14" size="0 0 1"
              type="plane" material="matplane" conaffinity="15" condim="3"/>

        <!-- LEFT SIDE -->
        <body pos="0 0 0">

            <geom name="left_bound" type="box" size="0.002 0.06 0.06"
                  rgba="0.1 0.1 0.7 1" pos="0 0 0"/>

            <geom fromto="0 0 0 0.094 0 0"
                  rgba=".5 .5 .1 .5" size=".002"
                  contype="0" conaffinity="0"/>

            <site name="s1" pos="0.005 0 0.035" size=".002" rgba="1 1 0 1"/>
            <site name="s2" pos="0.005 0 -0.035" size=".002" rgba="1 1 0 1"/>
            <!-- FIRST JOINT (ELBOW) -->
            <body pos="0.094 0 0">

                <joint name="elbow"/>

                <geom fromto="0 0 0 0.063 0 0"
                      rgba=".5 .5 .1 .5" size=".002"/>

                <geom name="Pulley" type="cylinder"
                      fromto="0 .007 0 0 -.007 0"
                      size=".014" rgba=".1 .1 .75 .85"/>

                <site name="s3" pos="0 0 0.015" size=".002" rgba="1 1 0 1"/>
                <site name="s4" pos="0 0 -0.015" size=".002" rgba="1 1 0 1"/>

                <!-- SECOND JOINT (WRIST) -->
                <body pos="0.063 0 0">

                    <joint name="wrist"/>

                    <geom fromto="0 0 0 0.088 0 0"
                          rgba=".5 .5 .1 .5" size=".002"/>

                    <geom name="Pulley2" type="cylinder"
                          fromto="0 .01 0 0 -.01 0"
                          size=".02" rgba=".1 .1 .75 .85"/>

                    <site name="s5" pos="0 0 0.021" size=".002" rgba="1 1 0 1"/>
                    <site name="s6" pos="0 0 -0.021" size=".002" rgba="1 1 0 1"/>
                    <site name="s7" pos="0.088 0 0.035" size=".002" rgba="1 1 0 1"/>
                    <site name="s8" pos="0.088 0 -0.035" size=".002" rgba="1 1 0 1"/>

                    <!-- RIGHT SIDE -->
                    <body>
                        <geom name="right_bound" type="box"
                              size="0.002 0.055 0.055"
                              rgba="0.1 0.1 0.7 1"
                              pos="0.088 0 0"/>
                    </body>

                </body>
            </body>
        </body>
    </worldbody>

    <!-- TENDONS -->
    <tendon>

        <!-- RED SPATIAL TENDON -->
        <spatial stiffness="10" rgba="1 0 0 1" width="0.0015">
```

```xml
            <site site="s1"/>
            <geom geom="Pulley" sidesite="s3"/>
            <site site="s3"/>
            <geom geom="Pulley2" sidesite="s6"/>
            <site site="s8"/>
        </spatial>

        <!-- GREEN SPATIAL TENDON -->
        <spatial stiffness="10" rgba="0 1 0 1" width="0.0015">
            <site site="s2"/>
            <geom geom="Pulley" sidesite="s4"/>
            <site site="s4"/>
            <geom geom="Pulley2" sidesite="s5"/>
            <site site="s7"/>
        </spatial>

    </tendon>

</mujoco>
```

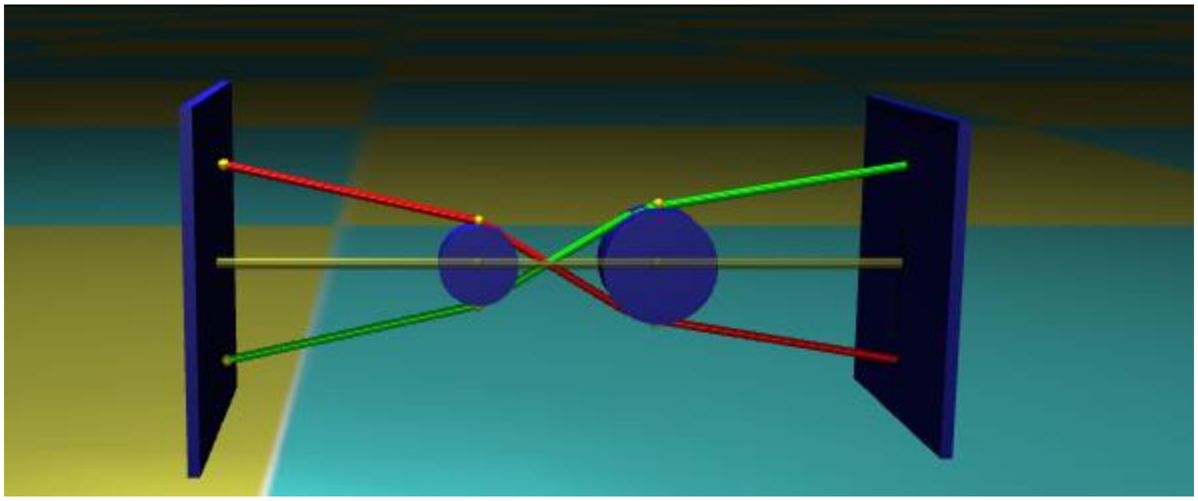
2. **Python Simulation Script**

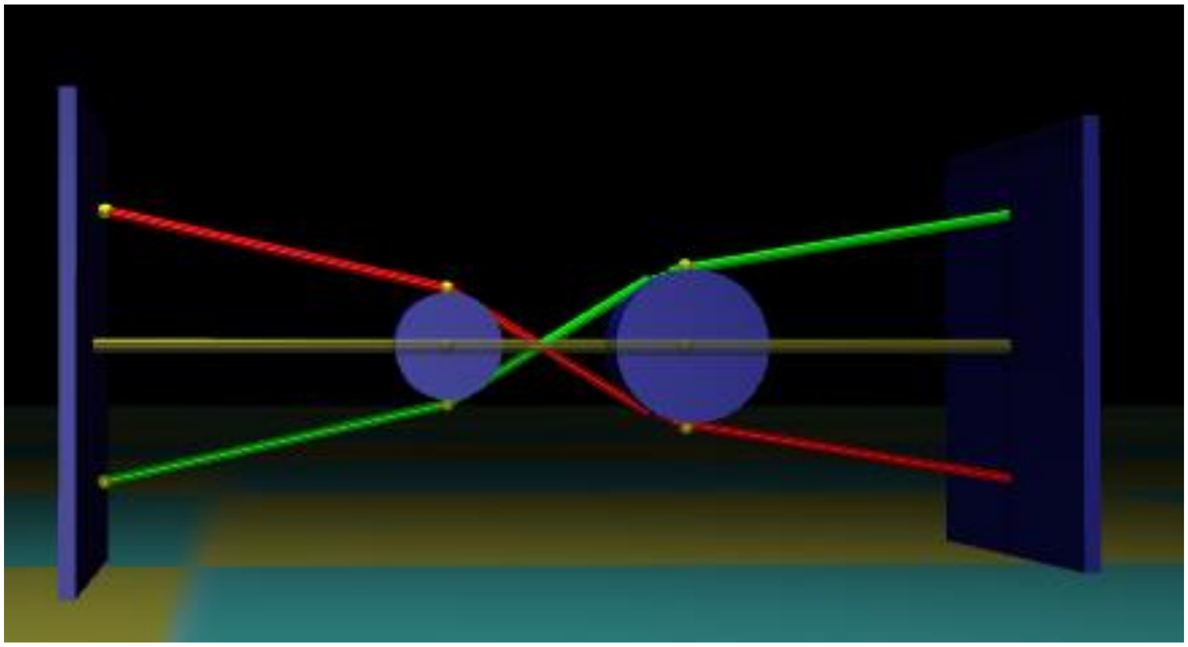
```python
import mujoco
import mujoco.viewer

# ---------------------------------------------------
# Load the XML model
# ---------------------------------------------------
model_path = r"C:\Users\Nagham\Documents\mujoco_models\model2.xml"

model = mujoco.MjModel.from_xml_path(model_path)
data = mujoco.MjData(model)

# ---------------------------------------------------
# Launch the passive viewer
# ---------------------------------------------------
viewer = mujoco.viewer.launch_passive(model, data)

# ---------------------------------------------------
# Configure the camera
# ---------------------------------------------------
viewer.cam.lookat[:] = [0.07, 0, 0]   # Look at the middle of the arm
viewer.cam.distance = 0.6             # Move camera further
viewer.cam.elevation = -10            # Slight tilt
viewer.cam.azimuth = 90               # Side view

print("Viewer is running. Press Enter to close...")
```

```
# ----------------------------------------------------
# Wait for user input and close viewer
# ----------------------------------------------------
input()
viewer.close()
```

6. Results and Analysis:

After we run the simulation in Python we have the model:





- The model successfully visualizes both links and pulleys.
- The tendon paths are routed correctly through the defined sites.
- When joints are moved manually:
    - Tendons stretch/shorten accordingly.
    - Tension propagates through the pulleys.
- The mechanism behaves exactly as expected for a **passively tendon-driven 2R mechanism**.

This verifies the correct geometric tendon routing and pulley interaction.

7. Model Simulation and Motion Testing:

After completing the construction of the mechanical model in MuJoCo, a Python simulation was developed to test the motion of the system under gravity and tendon constraints. The simulation initializes the model, applies slight initial joint offsets to trigger motion, and then runs the physics engine in passive mode. Throughout the simulation, the joint angles of the elbow and wrist are recorded at each time step.

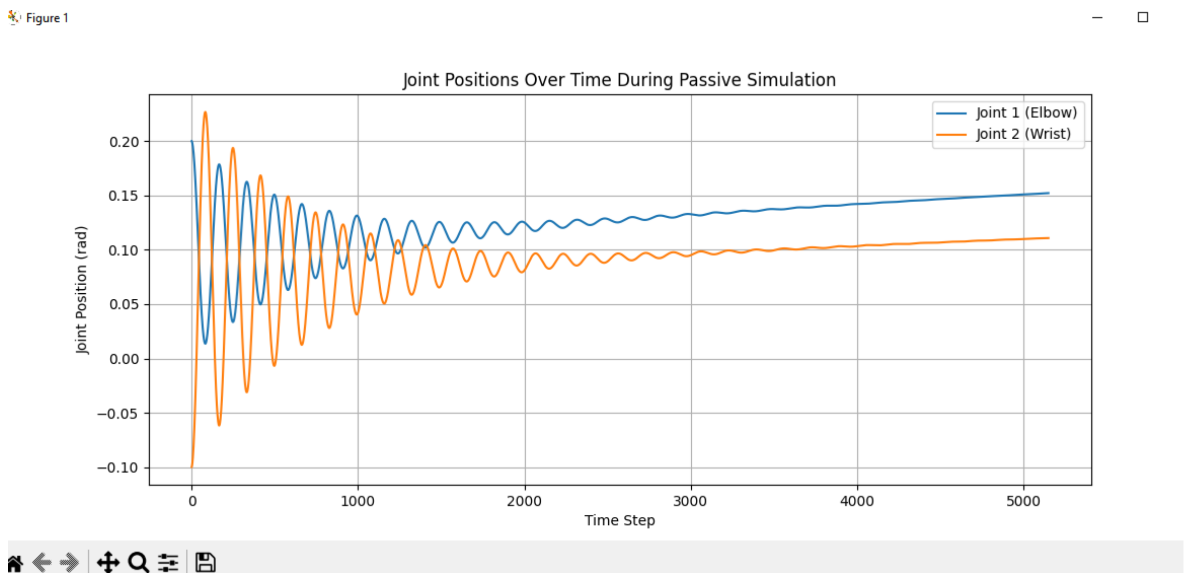
The purpose of this test is to verify that the model behaves dynamically as expected and that the tendons, joints, and geometric constraints produce correct motion. The following Python script was used to run the simulation and collect the joint trajectories:

The code:

```python
import time
import mujoco
import mujoco.viewer
import matplotlib.pyplot as plt

# Load the MuJoCo model
model_path = r"C:\Users\Nagham\Documents\mujoco_models\model2.xml"
model = mujoco.MjModel.from_xml_path(model_path)
data_ = mujoco.MjData(model)

# Storage for recording joint motion
joint1_positions = []
joint2_positions = []

# Get the joint indices by name
elbow_id = mujoco.mj_name2id(model, mujoco.mjtObj.mjOBJ_JOINT, 'elbow')
wrist_id = mujoco.mj_name2id(model, mujoco.mjtObj.mjOBJ_JOINT, 'wrist')

elbow_index = model.jnt_qposadr[elbow_id]
```

```python
wrist_index = model.jnt_qposadr[wrist_id]

# Apply initial conditions to trigger motion
data_.qpos[elbow_index] = 0.2
data_.qpos[wrist_index] = -0.1

# Run forward pass before starting the simulation
mujoco.mj_forward(model, data_)

# Launch viewer in passive mode
with mujoco.viewer.launch_passive(model, data_) as viewer:
    start = time.time()

    while viewer.is_running() and time.time() - start < 15:
        step_start = time.time()

        # Run one physics step
        mujoco.mj_step(model, data_)
        viewer.sync()

        # Record joint angles
        joint1_positions.append(data_.qpos[elbow_index])
        joint2_positions.append(data_.qpos[wrist_index])

        # Maintain stable real-time stepping
        dt = model.opt.timestep - (time.time() - step_start)
        if dt > 0:
            time.sleep(dt)

# Plot joint trajectories
plt.figure(figsize=(12,5))
plt.plot(joint1_positions, label="Joint 1 (Elbow)")
plt.plot(joint2_positions, label="Joint 2 (Wrist)")
plt.xlabel("Time Step")
plt.ylabel("Joint Position (rad)")
plt.title("Joint Positions Over Time During Passive Simulation")
plt.grid(True)
plt.legend()
plt.show()
```

After completing the construction of the model in **MuJoCo**, I tested its motion to verify the correctness of the joints and overall dynamics. When the simulation started, the joints moved and then gradually damped out due to the applied torque and model settings. After the motion settled and reached a damped state, I obtained the following plot, which shows the change in each joint angle over time.



8. Conclusion:

A full tendon-driven 2R planar mechanism was designed and simulated in MuJoCo. The system demonstrates how passive tendon mechanics can generate coordinated motion without applying direct torques at the joints. After constructing the model, a dynamic motion simulation was performed to observe the system's natural behavior under gravity and tendon elasticity.

This model provides a foundation for exploring more advanced tendon-based actuation strategies. It can be extended to include:

- **Active actuation**, where motors or actuators actively pull the tendons to produce controlled motion.
- **Optimization of tendon routing**, allowing improved force transmission, reduced friction, and more efficient mechanical behavior.
- **Investigations of compliant and biologically inspired motion**, enabling studies related to soft robotics, human biomechanics, and energy-efficient movement.

Overall, the system serves as a flexible platform for analyzing tendon dynamics, simulating realistic motion, and developing more advanced robotic mechanisms.