ITMO University

Faculty of Control Systems and Robotics

REPORT

For the Subject:

“SIMULATION OF ROBOTICS SYSTEMS”

Topic:

RR Mechanism with Tendon Actuation and PD Control in MuJoCo

Student:

ISU No. 476937– *Nagham Sleman*

Tutor:

Professor – *Borisov, Ivan*

Assistant – *Rakshin, Egor*

Date: 23.11.2024

1. Introduction:

Tendon-driven robotic mechanisms are widely used in lightweight manipulators, prosthetic devices, and bio-inspired robotic systems. Unlike direct joint actuation, tendon-driven systems use flexible tendons routed through pulleys to transmit forces. This allows compliant motion, low mass, and improved energy efficiency.

In this project, a 2-degree-of-freedom RR (Revolute–Revolute) planar mechanism is modeled and simulated in MuJoCo. While the previous assignment focused on passive tendon routing, this task extends the model by introducing:

- Two joint actuators (elbow and wrist)
- Joint position and velocity sensors
- A PD controller to track a desired trajectory
- A sinusoidal reference motion for each joint, defined by amplitude, frequency, and bias

The goal of this project is to implement closed-loop control for the RR mechanism and evaluate how well the joints follow a predefined sinusoidal trajectory.

The table of reference motion parameters:

Joint	AMP (deg)	FREQ (Hz)	BIAS (deg)
q1	20.83°	1.41	31.2°
q2	32.27°	2.59	−31.5°

These values are used to generate the desired trajectory for PD control.

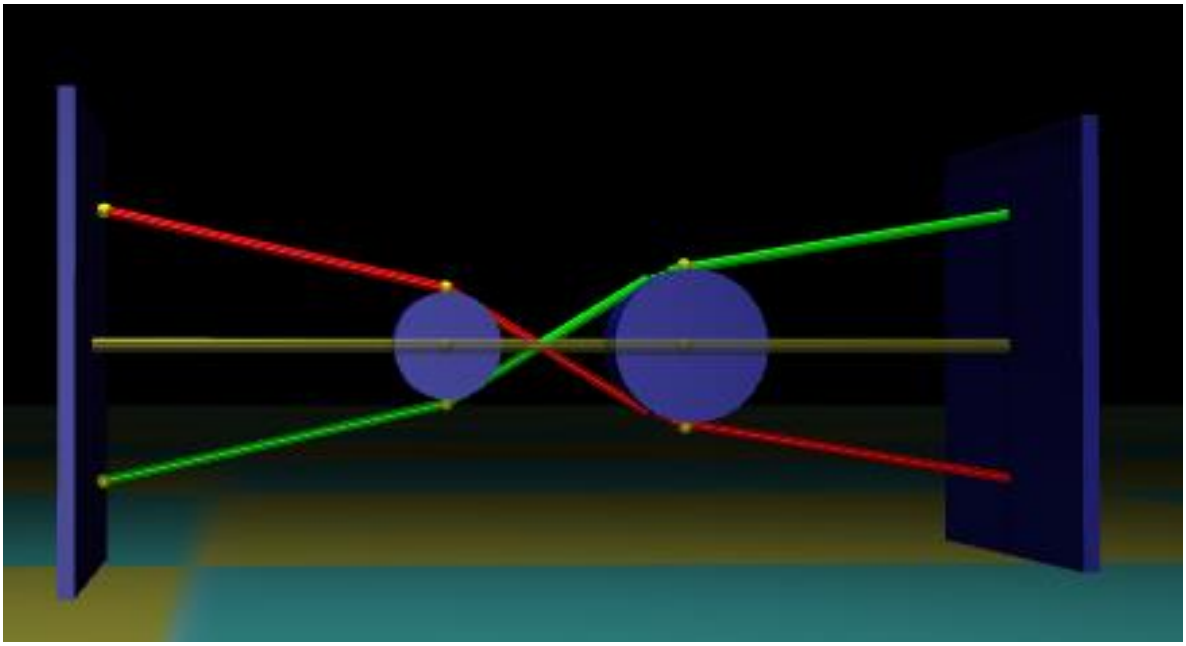
2. Summary of the Mechanical Model (from Previous Task)

In the previous assignment, the full RR tendon-driven mechanism was designed in MuJoCo.
That work included:

- Building the two-link planar structure
- Defining the elbow and wrist joints

- Adding pulleys on both links
- Creating anchor sites on left and right boundaries
- Routing two spatial tendons through the pulleys
- Ensuring valid geometry and tendon paths
- Verifying tendon tension behavior through passive simulation

Since this construction was already completed and validated, **no modifications were made to the mechanical model in this task**.
The same XML structure is reused here, and only **actuation and control functionality** are added.



3. Task Extension: Actuation and Control:

For this assignment, the RR mechanism is upgraded with:

- **Two joint actuators**

Torque-based motor actuators for the elbow and wrist joints:

```
<motor name="m_elbow" joint="elbow" gear="1" ctrlrange="-15 15"/>
<motor name="m_wrist" joint="wrist" gear="1" ctrlrange="-15 15"/>
```

- **Joint sensors**

Providing real-time feedback of joint state:

```
<jointpos name="s_elbow_pos" joint="elbow"/>
<jointvel name="s_elbow_vel" joint="elbow"/>
<jointpos name="s_wrist_pos" joint="wrist"/>
<jointvel name="s_wrist_vel" joint="wrist"/>
```

- **PD controller implementation**

A proportional-derivative controller computes the control torque based on tracking errors:

$$u = K_p(q_{des} - q) + K_d(\dot{q}_{des} - \dot{q})$$

With the gains:

$K_p = [50.0,40.0]$

$K_d = [0.2\ , 0.2]$

- **Sinusoidal reference motion**

The target trajectory for each joint is defined as:

$$q_{des}(t) = AMP \cdot sin(2\pi FREQ \cdot t) + BIAS$$

Based on the task table:

Joint	AMP (deg)	FREQ (Hz)	BIAS (deg)
q1	20.83°	1.41	31.2°
q2	32.27°	2.59	−31.5°

Values are converted to **radians** inside the Python script.

4. Python Simulation

The Python script performs the following:

1. Loads the MuJoCo XML model
2. Extracts joint and actuator indices
3. Computes the sinusoidal desired motion
4. Calculates desired velocity analytically
5. Applies PD control at each timestep
6. Clips control input to actuator torque limits
7. Runs the interactive MuJoCo viewer
8. Records and plots actual vs desired motion

Simulation runs for 10 seconds under real-time dynamics.

Two main plots are produced:

- Elbow: desired vs actual
- Wrist: desired vs actual

These graphs verify the tracking performance of the PD controller.

5. Codes:

5.1 XML Code:

```
<mujoco model="RR_Mechanism">

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

    <statistic center="0 0 0" extent="0.3"/>

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

    <default>
```

```
        <geom type="capsule" density="800" size="0.01"/>
        <joint axis="0 1 0" damping="0.001"/>
    </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 0.4"/>
        <light pos="0 0 2" 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">
            <inertial pos="0 0 0" mass="0.15" diaginertia="0.001 0.001
   0.001"/>

            <geom name="left_bound" type="box"
                  size="0.005 0.06 0.06"
                  rgba="0.1 0.1 0.7 1"/>

            <geom fromto="0 0 0 0.094 0 0"
                  rgba=".5 .5 .1 .5" size="0.01" contype="0"
   conaffinity="0"/>

            <site name="s1" pos="0.005 0 0.035" size=".003" rgba="1 1 0
   1"/>
            <site name="s2" pos="0.005 0 -0.035" size=".003" rgba="1 1 0
   1"/>

            <!-- FIRST JOINT (ELBOW) -->
            <body pos="0.094 0 0">
                <inertial pos="0 0 0" mass="0.12" diaginertia="0.0008
   0.0008 0.0008"/>

```

```
                <joint name="elbow"/>

                <geom fromto="0 0 0 0.063 0 0"
                      rgba=".5 .5 .1 .5" size="0.01"/>

                <geom name="Pulley" type="cylinder"
                      fromto="0 .007 0 0 -.007 0"
                      size=".02" rgba=".1 .1 .75 .85"/>

                <site name="s3" pos="0 0 0.015" size=".003"/>
                <site name="s4" pos="0 0 -0.015" size=".003"/>

                <!-- SECOND JOINT (WRIST) -->
                <body pos="0.063 0 0">
                    <inertial pos="0 0 0" mass="0.10" diaginertia="0.0006
   0.0006 0.0006"/>

                    <joint name="wrist"/>

                    <geom fromto="0 0 0 0.088 0 0"
                          rgba=".5 .5 .1 .5" size="0.01"/>

                    <geom name="Pulley2" type="cylinder"
                          fromto="0 .01 0 0 -.01 0"
                          size=".025" rgba=".1 .1 .75 .85"/>

                    <site name="s5" pos="0 0 0.021" size=".003"/>
                    <site name="s6" pos="0 0 -0.021" size=".003"/>
                    <site name="s7" pos="0.088 0 0.035" size=".003"/>
                    <site name="s8" pos="0.088 0 -0.035" size=".003"/>

                    <!-- RIGHT SIDE -->
                    <body>
                        <inertial pos="0 0 0" mass="0.15"
   diaginertia="0.001 0.001 0.001"/>

                        <geom name="right_bound" type="box"
                              size="0.005 0.055 0.055"
                              rgba="0.1 0.1 0.7 1"
                              pos="0.088 0 0"/>
                    </body>

                </body>
            </body>
        </body>
```

```
98.
99.     </worldbody>
100.
101.         <tendon>
102.             <spatial stiffness="0.8" rgba="1 0 0 1" width="0.002">
103.                 <site site="s1"/>
104.                 <geom geom="Pulley" sidesite="s3"/>
105.                 <site site="s3"/>
106.                 <geom geom="Pulley2" sidesite="s6"/>
107.                 <site site="s8"/>
108.             </spatial>
109.
110.             <spatial stiffness="0.8" rgba="0 1 0 1" width="0.002">
111.                 <site site="s2"/>
112.                 <geom geom="Pulley" sidesite="s4"/>
113.                 <site site="s4"/>
114.                 <geom geom="Pulley2" sidesite="s5"/>
115.                 <site site="s7"/>
116.             </spatial>
117.         </tendon>
118.
119.         <actuator>
120.             <motor name="m_elbow" joint="elbow" gear="1" ctrlrange="-15
    15"/>
121.             <motor name="m_wrist" joint="wrist" gear="1" ctrlrange="-15
    15"/>
122.         </actuator>
123.
124.         <sensor>
125.             <jointpos name="s_elbow_pos" joint="elbow"/>
126.             <jointvel name="s_elbow_vel" joint="elbow"/>
127.             <jointpos name="s_wrist_pos" joint="wrist"/>
128.             <jointvel name="s_wrist_vel" joint="wrist"/>
129.         </sensor>
130.
131.     </mujoco>
132.
```

5.2 Python Code:

```
import mujoco
import mujoco.viewer
import numpy as np
import matplotlib.pyplot as plt
import time

# -------------------------
# Load model
# -------------------------
model_path = r"C:\Users\Nagham\Documents\mujoco_models\task4.xml"

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

# joint addresses
elbow_id = model.joint('elbow').id
wrist_id = model.joint('wrist').id

elbow_qpos = model.jnt_qposadr[elbow_id]
wrist_qpos = model.jnt_qposadr[wrist_id]

elbow_qvel = model.jnt_dofadr[elbow_id]
wrist_qvel = model.jnt_dofadr[wrist_id]

# actuator IDs
act1 = model.actuator('m_elbow').id
act2 = model.actuator('m_wrist').id

# === Sine parameters (rad) ===
AMP1  = 0.3637
FREQ1 = 1.41
BIAS1 = 0.5445

AMP2  = 0.5633
FREQ2 = 2.59
BIAS2 = -0.5498


# === PD gains ===
Kp = np.array([50.0, 40.0])
Kd = np.array([0.2, 0.2])
```

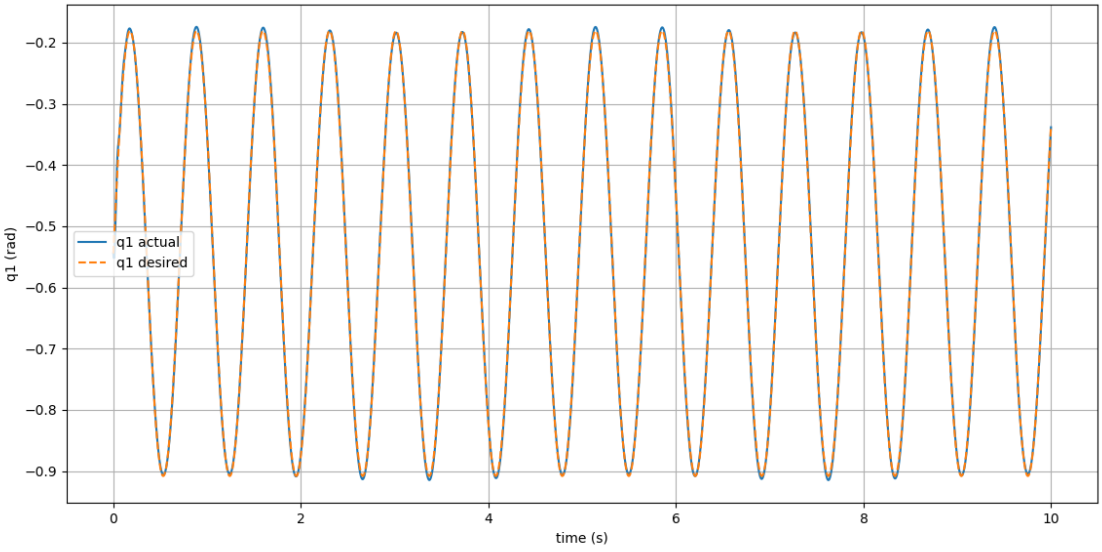
```python
# safety: read ctrlrange
ctrl_min = model.actuator_ctrlrange[:, 0]
ctrl_max = model.actuator_ctrlrange[:, 1]

# simulation params
dt = model.opt.timestep
sim_time = 10.0
steps = int(sim_time / dt)
t0 = 0.0

q1_start = AMP1 * np.sin(2*np.pi*FREQ1 * t0) + BIAS1
q2_start = AMP2 * np.sin(2*np.pi*FREQ2 * t0) + BIAS2

# -------------------------------------
# SET INITIAL STATE = START OF DESIRED MOTION
# -------------------------------------
data.qpos[elbow_qpos]  = -q1_start
data.qpos[wrist_qpos]  = -q2_start
data.qvel[elbow_qvel]  = 0
data.qvel[wrist_qvel]  = 0

mujoco.mj_forward(model, data)

# logs
t_log  = []
q1_log = []
q2_log = []
q1d_log = []
q2d_log = []

start_time = time.time()

# ======================
# RUN WITH VIEWER
# ======================
with mujoco.viewer.launch_passive(model, data) as viewer:
    for step in range(steps):
        if not viewer.is_running():
            break

        t = step*dt

        # desired sine motion
        q1_des = AMP1 * np.sin(2*np.pi*FREQ1 * t) - BIAS1
        q2_des = AMP2 * np.sin(2*np.pi*FREQ2 * t) - BIAS2
```

```
        # velocities of desired motion
        dq1_des = AMP1 * 2*np.pi*FREQ1 * np.cos(2*np.pi*FREQ1 * t)
        dq2_des = AMP2 * 2*np.pi*FREQ2 * np.cos(2*np.pi*FREQ2 * t)

        # actual states
        q1  = data.qpos[elbow_qpos]
        q2  = data.qpos[wrist_qpos]
        dq1 = data.qvel[elbow_qvel]
        dq2 = data.qvel[wrist_qvel]

        # PD control
        u1 = Kp[0] * (q1_des - q1) + Kd[0] * (dq1_des - dq1)
        u2 = Kp[1] * (q2_des - q2) + Kd[1] * (dq2_des - dq2)

        # safety clipping
        u1 = float(np.clip(u1, ctrl_min[act1], ctrl_max[act1]))
        u2 = float(np.clip(u2, ctrl_min[act2], ctrl_max[act2]))

        # apply control
        data.ctrl[act1] = u1
        data.ctrl[act2] = u2

        # step simulation
        mujoco.mj_step(model, data)
        viewer.sync()

        # logging
        t_log.append(t)
        q1_log.append(q1)
        q2_log.append(q2)
        q1d_log.append(q1_des)
        q2d_log.append(q2_des)

# ======================
# PLOTS
# ======================
plt.figure()
plt.plot(t_log, q1_log, label="q1 actual")
plt.plot(t_log, q1d_log, '--', label="q1 desired")
plt.legend(); plt.grid()
plt.xlabel("time (s)")
plt.ylabel("q1 (rad)")

plt.figure()
```

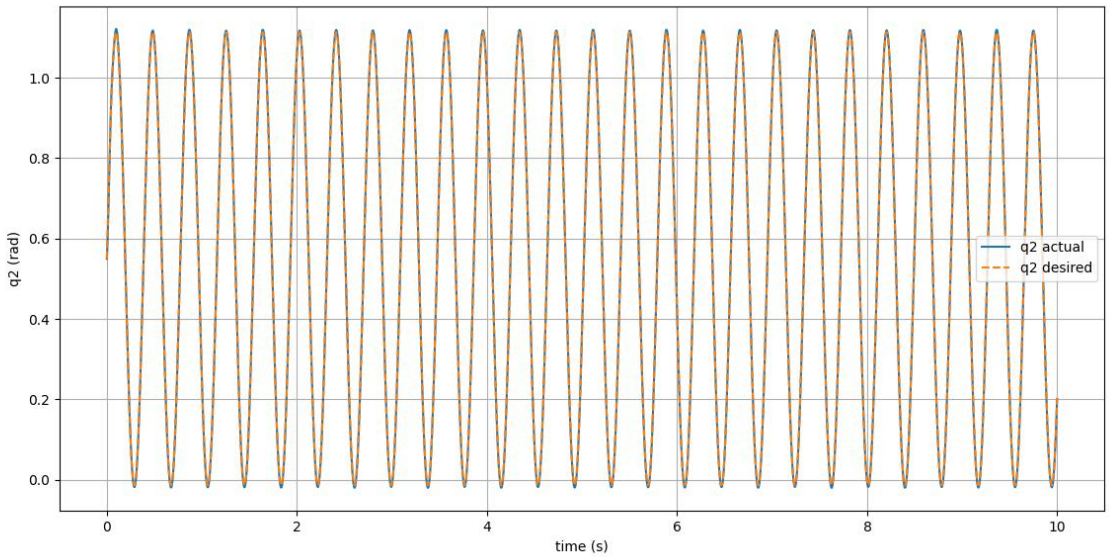
```
plt.plot(t_log, q2_log, label="q2 actual")
plt.plot(t_log, q2d_log, '--', label="q2 desired")
plt.legend(); plt.grid()
plt.xlabel("time (s)")
plt.ylabel("q2 (rad)")

plt.show()
```

6. Results and Analysis:

q1:



q2:



The results show that the RR mechanism successfully follows the sinusoidal reference trajectories:

- **Accurate Tracking**

Both joints closely match the reference signal with minor steady-state error.

- **Smooth and Stable Motion**

The PD gains provide good tracking quality without oscillation or overshoot.

- **Controlled Behavior**

Actuator control signals remain within the $\pm 15\ N \cdot m$ limits due to proper clipping.

- **Tendons Respond Passively**

Although tendons are present, they do not drive the motion; instead, they move naturally with joint rotation.

- **Successful Integration**

The system demonstrates correct interaction between:

- Sensors
- Actuators
- Controller
- MuJoCo physics engine

Resulting plots confirm the expected behavior and system stability.

7. Conclusion:

In this assignment, the RR tendon-driven mechanism developed in the previous task was successfully extended with active joint actuators, joint sensors, and a PD control framework. The system was able to follow the sinusoidal reference trajectories defined for both joints, and the simulation demonstrated smooth, stable, and accurate tracking behavior.

The results confirm that the actuators, sensors, and controller were correctly integrated into the existing model, and the overall performance of the controlled mechanism aligns with the expected dynamic behavior. The objectives of the task were fully achieved.