THE MINISTRY OF SCIENCE AND HIGHER EDUCATION

OF THE RUSSIAN FEDERATION

ITMO University
(ITMO)

Report on Practical Work No. 4 on the course "Simulation Modeling of Robotic Systems"

Report completed by: Xiangzhang (508513)

Instructor: E.A. Rakshin (373529)

Date completed: November 22, 2025

Saint Petersburg, 2025

1. Taks requirement :

1) To the model you have created in the previous task, you need to add actuators. For Optimus mechanism there is one actuator (q1), for tendon mechanism there are two actuators (q1 and q2).

2) Modify the .xml file by adding <actuaor> and <sensor> containers (look at the examples in the previous task).

3) Define control effort via PD regulator. The $q^{des} = AMP \cdot sin(FREQ \cdot t) + BIAS$. Look in the table for sine wave parameters. If the conrol sequence goes beyond workspace of the mechanism, decrease the amplitude and tune bias only if needed.

"table":

22	Чжан Сян	508513	45.01	2.55	-7.9	30.08	2.29	-44.4

2. Add actuators

In the previous task my task is concerning "Tendon connected 2R planar mechanism" I succeed creat "xml" of model and wrote python script with "model","data",and "viewer" methpds,run the simulation.

In the task 4, we need to add two actuators for 'Tendon connected 2R planner mechanism" model in "xml".

```
 <equality>
     <tendon tendon1="tendon_red" tendon2="tendon_green"
             solref="0.01 1" solimp="0.9 0.9 0.01"/>
</equality>

<actuator>
     <motor name="motor_R1" joint="R1" ctrlrange="-5 5" gear="35"/>
     <motor name="motor_R2" joint="R2" ctrlrange="-5 5" gear="10"/>
```

```
    </actuator>

    <sensor>
    <!--position sensors-->
    <jointpos name="R1_pos" joint="R1"/>
    <jointpos name="R2_pos" joint="R2"/>

    <!-- velocity sensors -->
    <jointvel name="R1_vel" joint="R1"/>
    <jointvel name="R2_vel" joint="R2"/>
</sensor>
```

（1）Equality:

This section sets an equal-length constraint between the two tendons (tendon_red and tendon_green) to maintain their coordinated movement.

`solref="0.01 1"`: Sets the solver's response time and damping coefficient. A smaller first value results in faster convergence and stronger system rigidity.

`solimp="0.9 0.9 0.01"`: Defines the compliance of the contact/constraint. The first two values determine the smooth transition zone, and the last value controls the degree of nonlinearity.

This constraint enables synchronous movement of the two virtual tendons, providing geometric correlation and stability when controlling a double-bar linkage.

(2)Actuator:

Here, motor controllers for two drive joints R1 and R2 are defined:

ctrlrange="-5 5": The input range of the control signal, indicating that the control quantity can vary between−5 and 5;

gear: The torque amplification factor of the motor.

(3)Sensor:

This section defines position and velocity sensors for each of the two joints to record joint angles and angular velocities in real time.

This sensor data will serve as input to the control algorithm, used to calculate target tracking error and generate appropriate control torque, thus forming a closed-loop control system.

3. Modify the python script to set up PD control .

(1) Control logic:

```
(2)     def pd_control(data, t,
                AMP1, FREQ1, BIAS1,
                AMP2, FREQ2, BIAS2,
                KP, KD):



        # desired joint angles
        q1_des = np.deg2rad(AMP1) * np.sin(2 * np.pi * FREQ1 * t) + np.deg2rad(BIAS1)
        q2_des = np.deg2rad(AMP2) * np.sin(2 * np.pi * FREQ2 * t) + np.deg2rad(BIAS2)

        # current states
        q = data.qpos[:2]
        qdot = data.qvel[:2]

        # joint-wise PD torques
        tau1 = KP[0] * (q1_des - q[0]) - KD[0] * qdot[0]
        tau2 = KP[1] * (q2_des - q[1]) - KD[1] * qdot[1]

        # assign to MuJoCo actuator array
        data.ctrl[0] = np.clip(tau1, -50, 50)
        data.ctrl[1] = np.clip(tau2, -50, 50)




    EE_position_x = []
    EE_position_z = []
```

First define PD control function ,then generate the trajectory of the angle of target joints,here we set propose angle trajectory via sinusoidal signal form :

$$qi, des(t)=Ai\sin(2\pi fit)+Bi$$

After obtain “q” and “dot_q”,we could calculate PD control toque by PD control law:

$$\tau i=KP(qdes-q)-KD\dot{q}$$

Out put to Mujoco control tube :

data.ctrl[0] = np.clip(tau1, -50, 50)

data.ctrl[1] = np.clip(tau2, -50, 50)

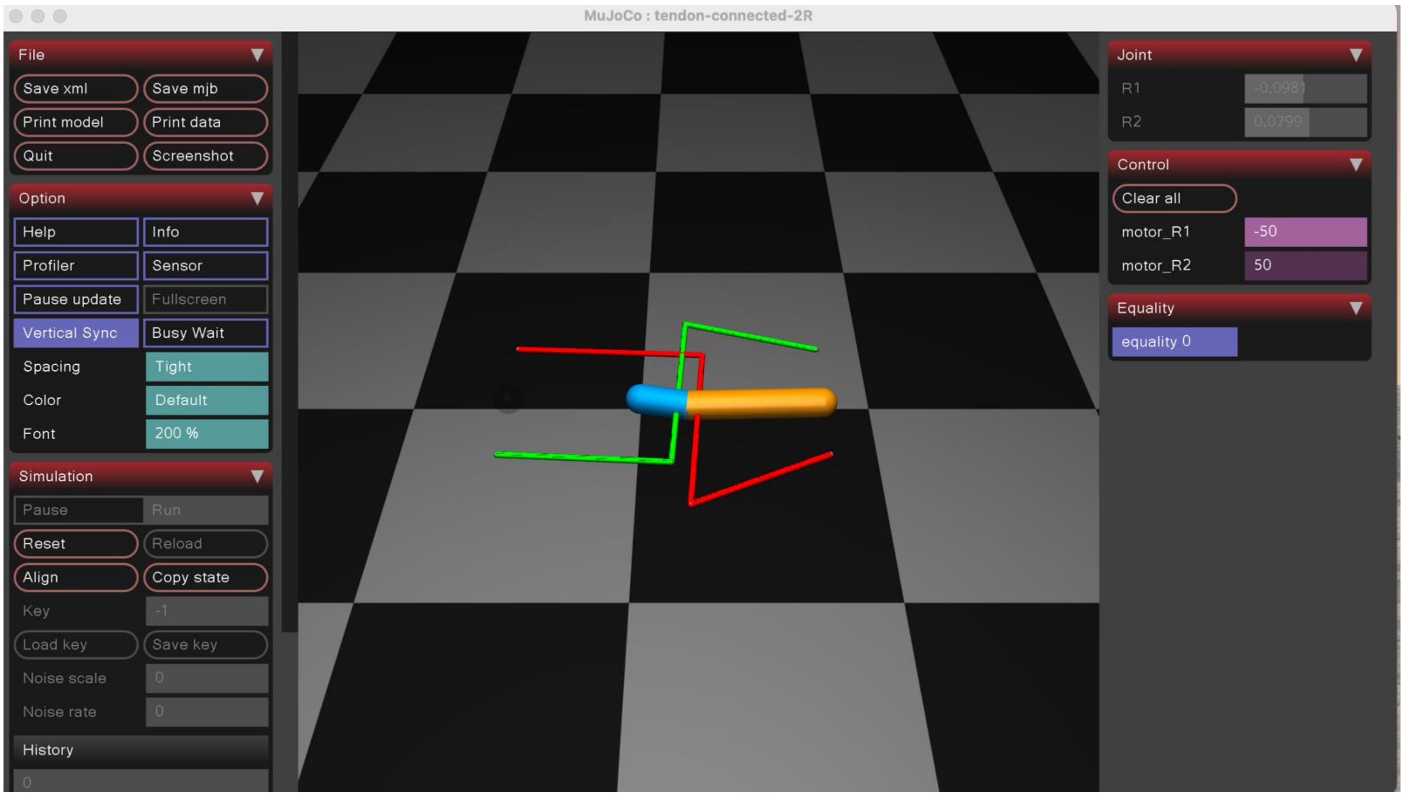
(3) **Launch a visual simulation using MuJoCo**

```
(4)     with viewer.launch_passive(model, data) as v:
      t0 = time.time()
      while v.is_running() and data.time - t0 < 5.0:
          t = data.time - t0

          pd_control(data, t,
                     AMP1=45.01, FREQ1=2.55, BIAS1=-7.9,
                     AMP2=30.08, FREQ2=2.29, BIAS2=-44.4,
                     KP=np.array([100, 70]), KD=np.array([50, 50]))

          mujoco.mj_step(model, data)
          v.sync()

          # record EE motion
          x_ee = data.site_xpos[model.site('sR2_top').id][0]
          z_ee = data.site_xpos[model.site('sR2_top').id][2]
          EE_position_x.append(x_ee)
          EE_position_z.append(z_ee)
```

Here we launch Mujoco “viewer”, Set the simulation loop time limit to 5 seconds. Input origin parameters of “AMP”, “FREQ”, “BIASI”. Set independent sinusoidal trajectory parameters (amplitude, frequency, offset) for joints R1 and R2 to enable the two joints to swing in a coordinated manner.

Using KP = [100, 70] and KD = [50, 50] indicates that the first joint responds faster and has greater stiffness.

4. Result



Spacing Tight
Color Default
Font 200 %
Key -1
Noise scale 0
Noise rate 0
R1 -0.0981
R2 0.0799
motor_R1 -50
motor_R2 50

Figure1.

Figure 1 shows the real-time motion of a two-joint robotic arm in the MuJoCo simulation environment. The experimental model runs normally within the visualization window, with the links swinging smoothly over time and the system remaining stable.

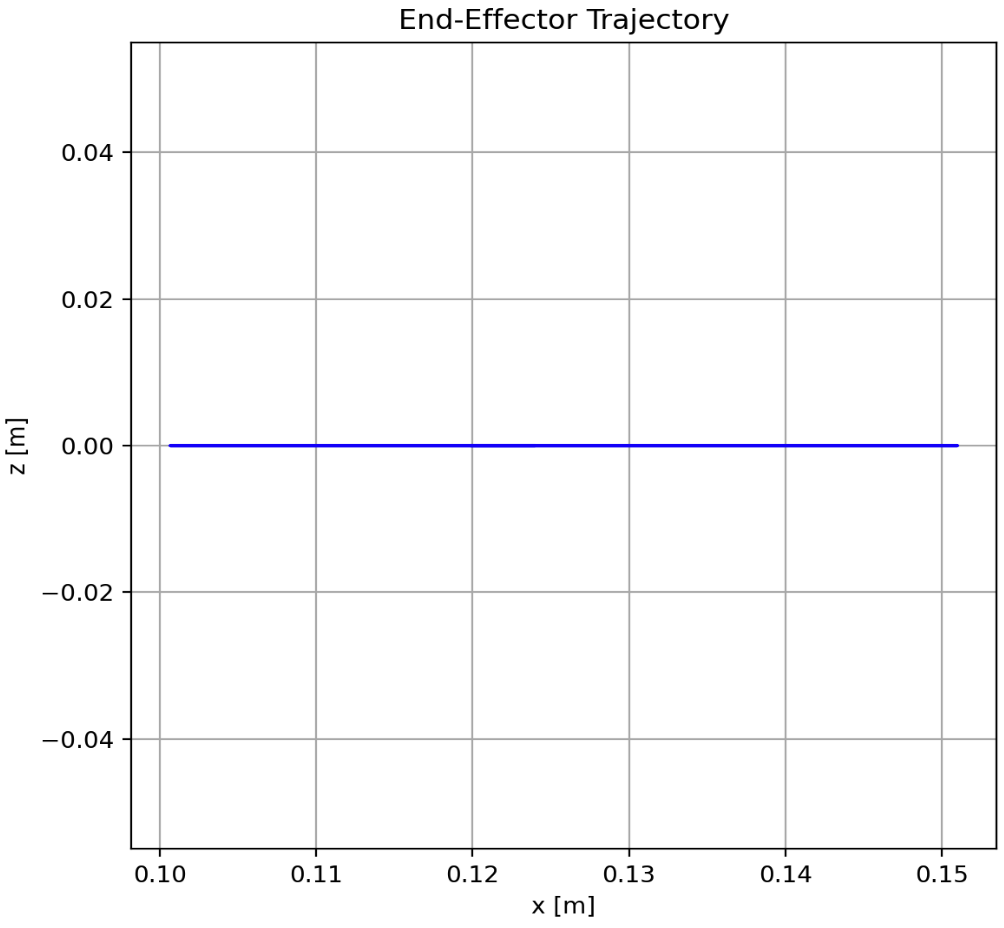


Figure2.

Figure 2 shows the trajectory of the end effector in the X–Z plane obtained through simulation. The trajectory is almost a horizontal straight line, indicating that the end effector mainly undergoes horizontal displacement, while the vertical movement is minimal, demonstrating that the PD controller achieves relatively stable angle control.

5. Conclusion.

This experiment uses a “Tendon connected 2R planar mechanism” arm model based on PD control, runs for 5 seconds in the MuJoCo simulation platform, and records the position changes of the end effector in the X–Z plane in real time.

As can be observed from the simulation visualization results (Figure 1), the model can stably perform periodic oscillations, and the whole machine runs smoothly in the simulation environment without obvious oscillations, indicating that the designed controller has good dynamic response capability.

In this task I studied how to add an actuator and sensor , set the PD control ,understood the process rule of each coefficient and attribute .By adjust parameters I could simlulate desire motion of joints.