THE MINISTRY OF SCIENCE AND HIGHER EDUCATION

OF THE RUSSIAN FEDERATION

ITMO University
(ITMO)

Report on Practical Work No. 3 on the course

"Simulation Modeling of Robotic Systems"

Report completed by: Xiangzhang (508513)

Instructor: E.A. Rakshin (373529)

Date completed: November 16, 2025

Saint Petersburg, 2025

1. Taks requirement :

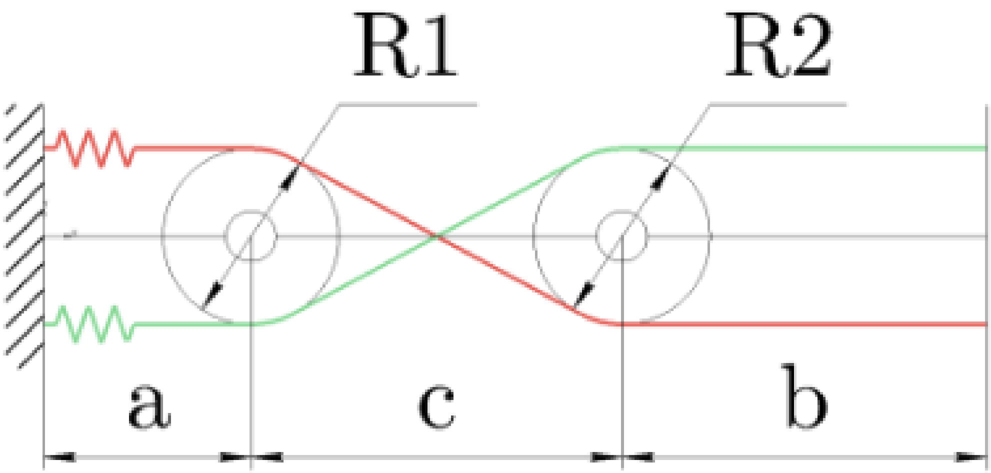
1) Look in the table and find yourself:

2) Choose one of the passive mechanisms according to your list and model .xml files.

3) Write python script with model, data and viewer methods. Run the simulation.

22	Чжан Сян	508513	TENDON	0.017	0.028	0.088	0.089	0.032

Variant 1 - tendon connected 2R planar mechanism:



2. Create xml model sample and implement in Mujoco.

1) Basic setting :

```
<mujoco model="tendon-connected-2R">
    <compiler angle="degree"/>
    <option timestep="1e-4" gravity="0 0 -9.8"/>

    <visual>
        <map znear="0.001"/>
        <quality shadowsize="2048"/>
    </visual>

    <asset>
        <texture type="skybox" builtin="gradient"
                 rgb1="0.7 0.8 1" rgb2="0.1 0.1 0.1"
                 width="512" height="256"/>
        <texture name="grid" type="2d" builtin="checker"
                 rgb1="0.1 0.1 0.1" rgb2="0.6 0.6 0.6"
```

```
                        width="512" height="512"/>
        <material name="grid" texture="grid" texrepeat="10 10" reflectance="0.1"/>
    </asset>
```

2) Construct Bodies and Joints:

```
<worldbody>
```

light source:

```
<light pos="0 0 3" dir="0 0 -1"/>
```

Ground:

```
<geom type="plane" pos="0 0 -1" size="5 5 0.001"
              material="grid" rgba="0.8 0.8 0.8 1"/>
```

Suspended fixed base (as tendon anchor point):

```
<body name="base" pos="0 0 0">
           <geom type="sphere" size="0.01" rgba="0 0 0 0.1"/>
```

Four anchors in the sky:

```
<site name="anchor_left_top"      pos="0.0   0.05 0" size="0.002"/>
            <site name="anchor_left_bottom"   pos="0.0 -0.05 0" size="0.002"/>
            <site name="anchor_right_top"     pos="0.6   0.05 0" size="0.002"/>
            <site name="anchor_right_bottom" pos="0.6 -0.05 0" size="0.002"/>
```

3) Bodies:

```
</body>

        <!-- link1 -->
        <body name="link1" pos="0.1 0 0">
            <joint name="R1" type="hinge" axis="0 0 1" limited="false"/>
            <geom name="link1_geom" type="capsule" fromto="0 0 0 0.25 0 0" size="0.01"
                  rgba="0.2 0.6 0.9 1"/>

            <!-- link1 tendon attachment point at the end -->
            <site name="sR1_top"      pos="0.25   0.05 0" size="0.002"/>
            <site name="sR1_bottom" pos="0.25 -0.05 0" size="0.002"/>

            <!-- The second link is attached to the end of link 1. -->
            <body name="link2" pos="0.25 0 0">
                <joint name="R2" type="hinge" axis="0 0 1" limited="false"/>
                <geom name="link2_geom" type="capsule" fromto="0 0 0 0.25 0 0" size="0.01"
```

```
                        rgba="0.9 0.6 0.2 1"/>

                <!--tendon attachment point at the origin of link2   -->
                <site name="sR2_top"      pos="0.0   0.05 0" size="0.002"/>
                <site name="sR2_bottom" pos="0.0 -0.05 0" size="0.002"/>
            </body>
        </body>
    </worldbody>
```

4) Tendon Definition:

```
<tendon>
        <!-- Red tendon: Up -> Down Crossing -->
        <spatial name="tendon_red" width="0.002"
                 springlength="0.6" stiffness="800" damping="0.5"
                 rgba="1 0 0 1">
            <site site="anchor_left_top"/>
            <site site="sR1_top"/>
            <site site="sR2_bottom"/>
            <site site="anchor_right_bottom"/>
        </spatial>

        <!-- Green tendon: Lower -> Upper crossing -->
        <spatial name="tendon_green" width="0.002"
                 springlength="0.6" stiffness="800" damping="0.5"
                 rgba="0 1 0 1">
            <site site="anchor_left_bottom"/>
            <site site="sR1_bottom"/>
            <site site="sR2_top"/>
            <site site="anchor_right_top"/>
        </spatial>
    </tendon>
```

5) Equality and Actuator:

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

    <actuator>
        <motor joint="R1" ctrlrange="-1 1" gear="50"/>
    </actuator>

</mujoco>
```

3. Drag the “xml” model in Mujoco and implement.

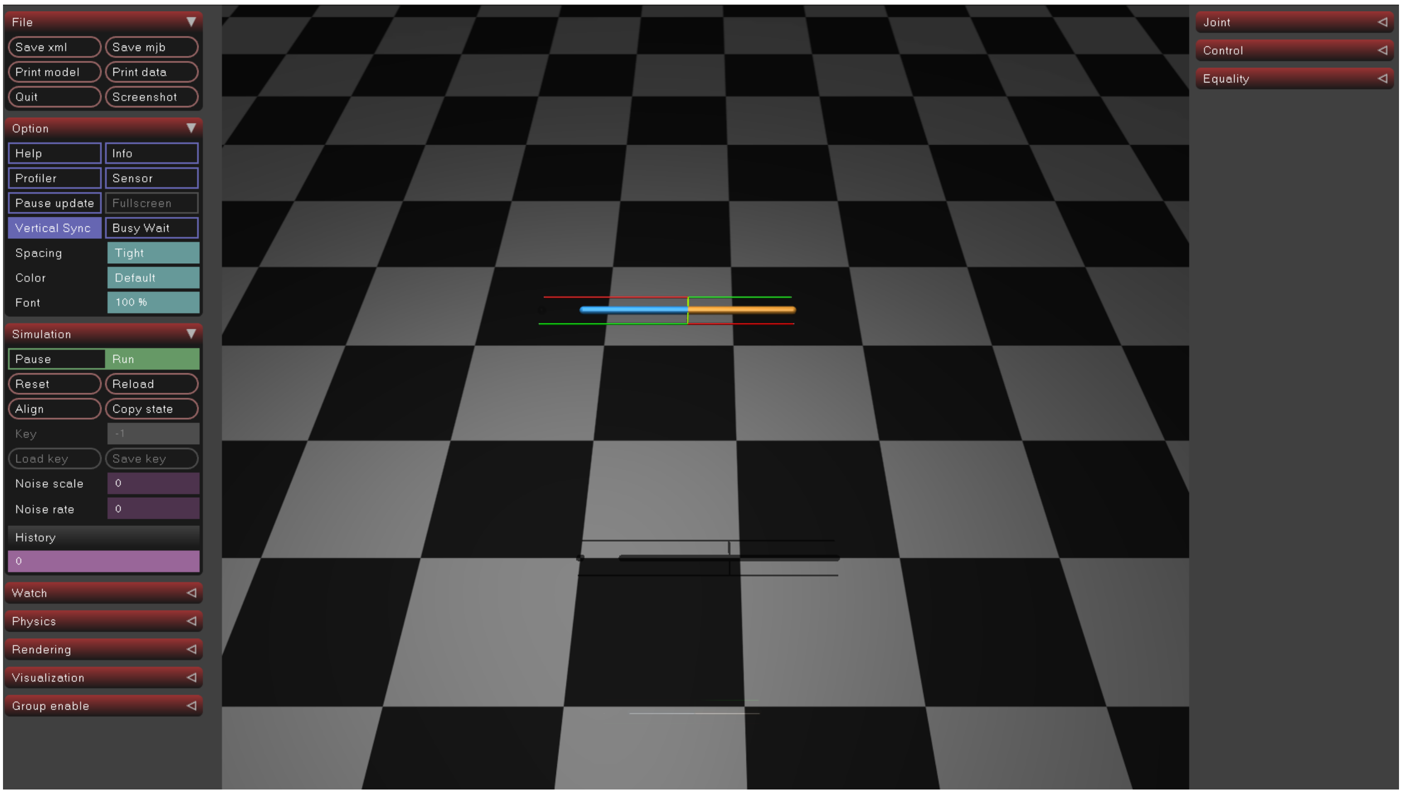


Figure1 .Display “xml”model in Mujoco

4. Write python script with “model”, “data” and “viewer” methods.

1) Set basic import :

```
2) import mujoco
   from mujoco import viewer
   import numpy as np
   import os
   from lxml import etree
   import time

   # ---------- Matplotlib
   import matplotlib
   matplotlib.use("Agg")
   import matplotlib.pyplot as plt
```

2) Setting parameters and modify functions :

```
3) f1 = "/Users/zhangx1ang/VS_Code/tendon-connected-2R__.xml"
   f2 = "/Users/zhangx1ang/VS_Code/tendon-connected-2R_modified__.xml"

   # parameter
   R1, R2, a, b, c = 0.017, 0.028, 0.088, 0.089, 0.032

   import xml.etree.ElementTree as ET

   def swap_par(tree, element_type, element_name, attr, new_value):

       element = tree.find(f'.//{element_type}[@name="{element_name}"]')
```

3) Modify parameters, attributes and save "xml":

```
4) tree = ET.parse(f1)
   root = tree.getroot()

   # link1
   swap_par(root, 'body', 'link1', 'pos', f"{a} 0 0")
   swap_par(root, 'geom', 'link1_geom', 'fromto', f"0 0 0 {c} 0 0")

   # link2
   swap_par(root, 'body', 'link2', 'pos', f"{c} 0 0")
   swap_par(root, 'geom', 'link2_geom', 'fromto', f"0 0 0 {b} 0 0")

   # tendon sites
   swap_par(root, 'site', 'sR1_top', 'pos', f"{c}  {R1 * 3} 0")
   swap_par(root, 'site', 'sR1_bottom', 'pos', f"{c} -{R1 * 3} 0")
   swap_par(root, 'site', 'sR2_top', 'pos', f"0.0  {R2 * 3} 0")
   swap_par(root, 'site', 'sR2_bottom', 'pos', f"0.0 -{R2 * 3} 0")

   # anchor
   x_right = a + c + b
   swap_par(root, 'site', 'anchor_right_top', 'pos', f"{x_right}  0.05 0")
   swap_par(root, 'site', 'anchor_right_bottom', 'pos', f"{x_right} -0.05 0")

   # save XML
   tree.write(f2, encoding='utf-8', xml_declaration=True)
   print("xml updated and saved", f2)
```

4) Loading model :

```
model = mujoco.MjModel.from_xml_path(f2)
data = mujoco.MjData(model)
```

5) Control logic and input simulation parameters:

```
6) def set_torque(mj_data, KP, KV, theta):
       mj_data.ctrl[0] = KP * (-mj_data.qpos[0] + theta) + KV * (0 -
   mj_data.qvel[0])



   SIMEND = 20
   TIMESTEP = 0.01
   STEP_NUM = int(SIMEND / TIMESTEP)
   timeseries = np.linspace(0, SIMEND, STEP_NUM)

   T = 2  # [s]
   FREQ = 1 / T
   AMP = np.deg2rad(-90)
   BIAS = np.deg2rad(-90)

   theta_des = AMP * np.sin(FREQ * timeseries) + BIAS

   #  record datas
   EE_position_x, EE_position_z = [], []

   #   MuJoCo Viewer
   with viewer.launch_passive(model, data) as v:
       t0 = time.time()
       while v.is_running() and data.time - t0 < 5.0:  # 模拟 5 秒
           t = data.time - t0

           # Apply periodic control to the first joint
           data.ctrl[0] = 0.3 * np.sin(2 * np.pi * 0.5 * t)

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

           # Get the location of the end effector site
           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)
```

6)Generate trajectory:

```python
print(f"Matplotlib backend: {plt.get_backend()}")

plt.figure(figsize=(6, 6))
plt.plot(EE_position_x, EE_position_z, color='b')
plt.xlabel("x [m]")
plt.ylabel("z [m]")
plt.title("End-Effector Trajectory")
plt.grid(True)
plt.tight_layout()
plt.savefig("ee_trajectory.png", dpi=200)
```

5. Run the simulation

1) In Mujoco:

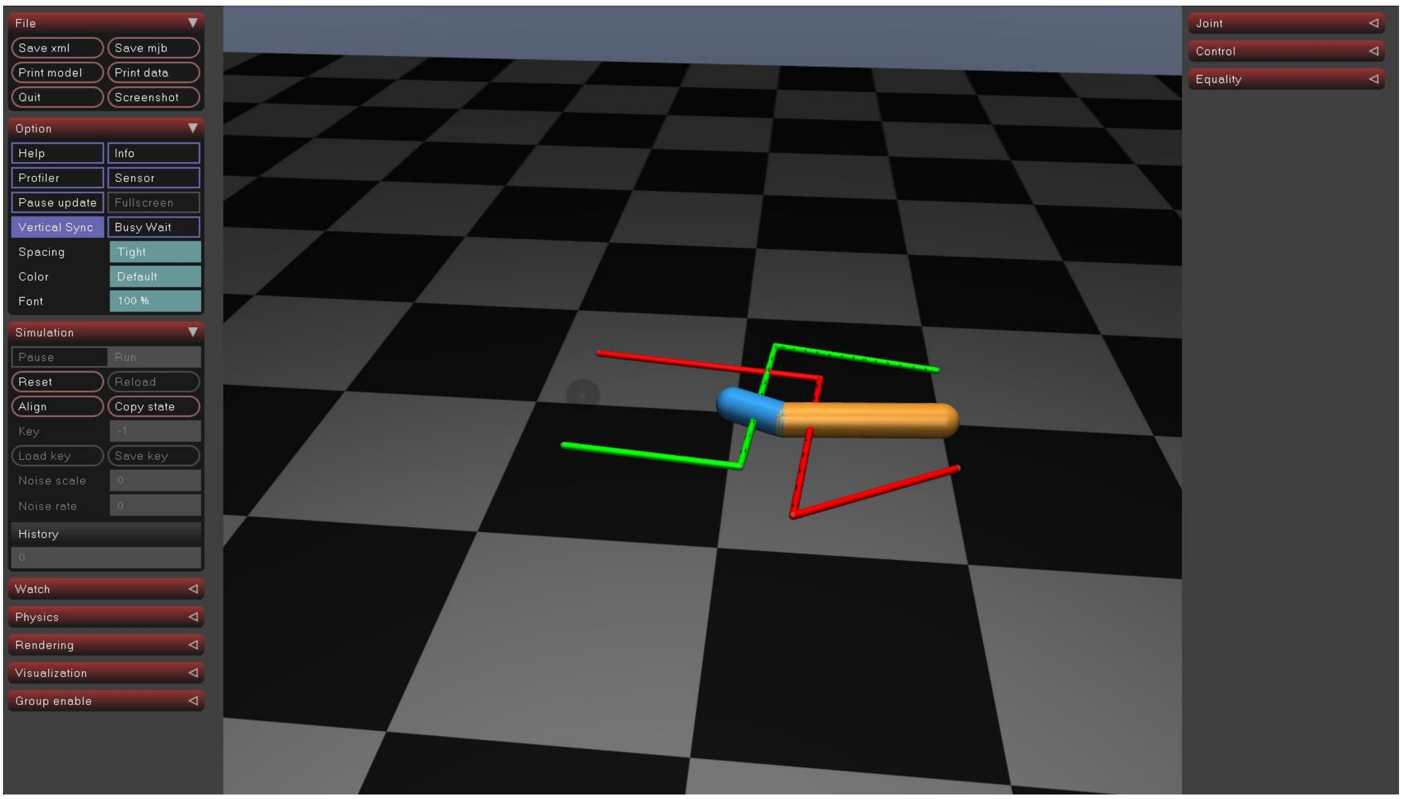
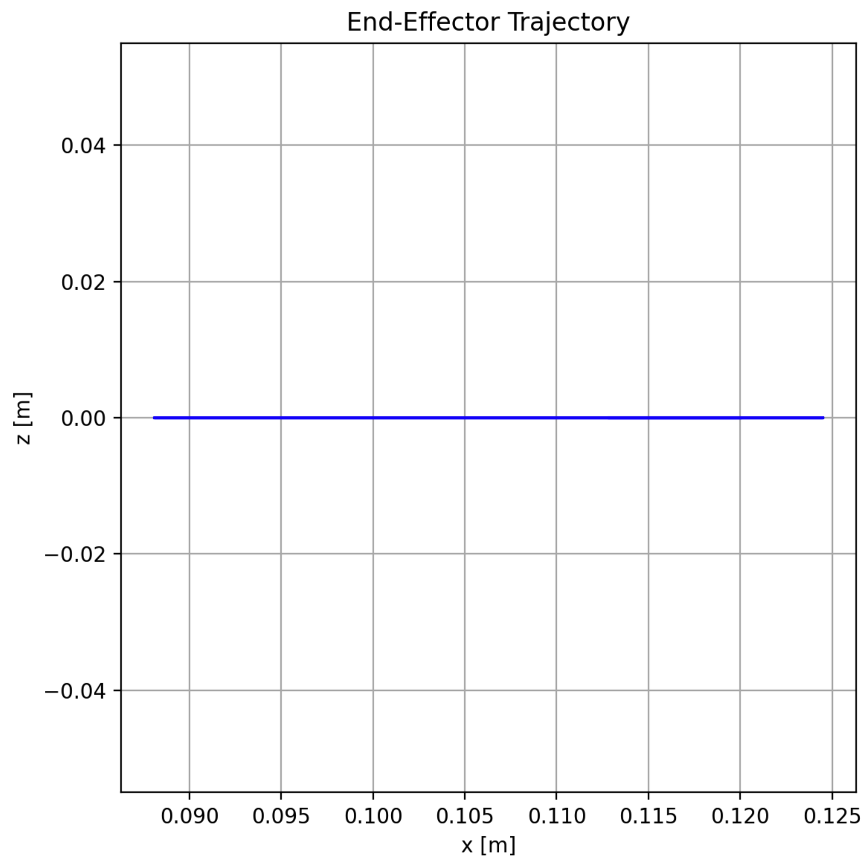


Figure2. tendon connected 2R planar mechanism

2) Graph of trajectory



6. Conclusion:

In this task I studied how to set up a mechanism model in “xml”,understood how to set the basic properties like: “asset”, “visual” ,“worldbody”,”equality” and “actuator” ,know the the hierarchical relationship and coordinate position of the body.

During writing the python script, I understood the logic to modify the parameters and attributes by functions ,it is more convience to adjust them in python ,although met a lot of problems in geometry .Then I understood the control logic to set a torque by function concerning “KP” and “KV” .Last generate the graph of trajectory.