Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина:

«Simulation of Robotic System»

ОТЧЕТ ПО ЛАБОРАТОРНЫЕ РАБОТЕ №3

Выполнили:
студент группы R4137C, Зулми Жудха Факрал

_____

(подпись)

Проверил:
Ракшин Егор Александрович

_____

(подпись)

Санкт-Петербург
2025

1. Introduction

The objective of this laboratory exercise was to model and simulate a passive mechanical system—specifically a Tendon Connected 2R Planar Mechanism—utilizing the MuJoCo physics engine. Passive mechanisms rely on constraints, springs, or fixed geometries rather than continuous external control inputs to define their motion. This particular design exemplifies how flexible components, like tendons, can be used to enforce complex kinematic couplings between rigid bodies. The entire simulation pipeline, from parameter definition to visualization, was managed using a Python script that dynamically modifies and executes the MuJoCo XML model.

System Parameters:

R1 = 0.018

R2 = 0.017

a = 0.054

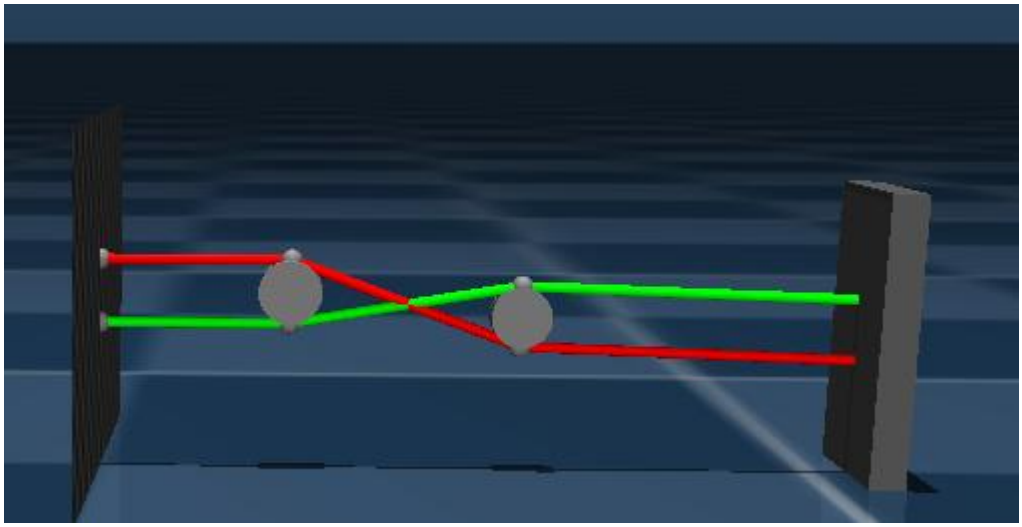
b = 0.065

c = 0.098

The simulation was executed for 20 seconds with a fine timestep of 0.001 seconds. After loading the model and data into the MuJoCo environment, the mujoco_viewer was initialized to observe the dynamics.

```
IMEND = 20
IMESTEP = 0.001
TEP_NUM = int(SIMEND / TIMESTEP)
imeseries = np.linspace(0, SIMEND, STEP_NUM)
```

As the simulation commenced, the mechanism was observed to stabilize quickly. Since no external forces or control signals were applied, the system's initial movement was solely dictated by gravity acting on the bodies and the tension equilibrium imposed by the stiff tendons. The system immediately demonstrated its passive nature: any potential for free movement in one joint (like the rotation of the first pulley) was instantaneously resisted or driven by the strain and stiffness of the connecting tendons, which are coupled to the vertical slide joints.



This demonstrates that the **spatial tendon** elements successfully enforced the desired kinematic relationship. If an external perturbation were applied—for instance, a downward force on the sliding wall—the tendons would stretch, increasing their tension. This increased tension would then force the first pulley to rotate until a new force equilibrium was reached, effectively showing the mechanical linkage established by the passive elements. The entire simulation

remained highly stable, confirming the correct implementation of the joints, equality constraint, and the high-stiffness tendons.

2. Conclusion

The laboratory work successfully modeled and simulated a Tendon Connected 2R Planar Mechanism in MuJoCo. By employing a Python-based dynamic parameterization scheme and leveraging MuJoCo's powerful **spatial tendon** element, we demonstrated a system where complex motion coupling is achieved passively. The simulation results confirm that the tendons effectively constrain the system, translating potential rotational energy into potential tensile energy and ultimately enforcing a closed-loop mechanical relationship between the rotational joint and the two translational joints. This model serves as an excellent illustration of how passive mechanisms can be defined and analyzed for applications where rigid, fixed motion ratios are required without the use of active control.

Full code :

Xml:

```
<mujoco model="tendon">

  <statistic center="0 0 0.55" extent="1.1"/>

  <visual>
    <headlight diffuse="0.6 0.6 0.6" ambient="0.3 0.3 0.3" specular="0 0 0"/>
    <rgba haze="0.15 0.25 0.35 1"/>
    <global azimuth="-90" elevation="-20"/>
  </visual>

  <asset>
    <texture type="skybox" builtin="gradient" rgb1="0.3 0.5 0.7" rgb2="0 0 0"
             width="512" height="3072"/>
    <texture type="2d" name="groundplane" builtin="checker" mark="edge"
             rgb1="0.2 0.3 0.4" rgb2="0.1 0.2 0.3" markrgb="0.8 0.8 0.8"
             width="300" height="300"/>
    <material name="groundplane" texture="groundplane" texuniform="true"
              texrepeat="5 5" reflectance="0.2"/>
```

```
    <texture name="grid" type="2d" builtin="checker" rgb1="0.1 0.1 0.1"
rgb2="0.6 0.6 0.6" width="300" height="300"/>
    <material name="grid" texture="grid" texrepeat="10 10" reflectance="0.2"/>
  </asset>

  <worldbody>

    <light pos="0 0 3" dir="0 0 -1" directional="false"/>
    <geom name="floor" size="0 0 0.125" type="plane" material="groundplane"
conaffinity="15" condim="3"/>

    <body name="wall_1" pos="0 0 0" euler="0 90 0">
        <geom name="wall_geom_1" type="plane" size="0.1 0.1 0.1"
material="grid"/>
    </body>

    <body name="Frame" pos="0 0 0">

        <!-- Pulley 1 -->
        <body name="pulley1_body" pos="{a} 0 0.05">
            <joint name="joint_pulley1" type="hinge" axis="0 1 0"
limited="true" range="0 360"/>
            <geom name="pulley1" type="cylinder" size="{R1/2} 0.005" euler="90
0 0"/>
            <site name="red1"   pos="0 0 {R1/2}" size="0.003"/>
            <site name="green1" pos="0 0 {- R1/2}" size="0.003"/>

            <!-- Pulley 2 -->
            <body name="pulley2_body" pos="{c} 0 0">
                <joint name="pulley2_slide_z" type="slide" axis="0 0 1"/>
                <geom name="pulley2" type="cylinder" size="{R2/2} 0.005"
euler="90 0 0"/>
                <site name="red2"   pos="0 0 {- R2/2}" size="0.003"/>
                <site name="green2" pos="0 0 { R2/2}" size="0.003"/>
                <site name="pulley2_connector" pos="0 0 0" size="0.003"/>

                <body name="wall_2" pos="{b} 0 0">
                    <joint name="wall2_slide_z" type="slide" axis="0 0 1"
limited="true" range="0 0.2"/>
                    <geom name="wall_geom_2" type="box" size="0.005 0.03 0.04"
material="grid" contype="1" conaffinity="1"/>
                    <site name="red3"   pos="0 0 {-R2/2}" size="0.003"/>
                    <site name="green3" pos="0 0 { R2/2}" size="0.003"/>
                    <site name="wall2_connector" pos="0 0 0"/>
                </body>

            </body>
```

```
        </body>

    </body>

    <site name="red0"   pos="0 0 {0.05 + R1/2}" size="0.003"/>
    <site name="green0" pos="0 0 {0.05 - R1/2}" size="0.003"/>

  </worldbody>

  <equality>
    <connect site1="pulley2_connector" site2="wall2_connector"/>
  </equality>


  <tendon>
    <spatial stiffness="100" rgba="1 0 0 1" width="0.0015">
      <site site="red0"/>
      <site site="red1"/>
      <site site="red2"/>
      <site site="red3"/>
    </spatial>
  </tendon>

  <tendon>
    <spatial stiffness="100" rgba="0 1 0 1" width="0.0015">
      <site site="green0"/>
      <site site="green1"/>
      <site site="green2"/>
      <site site="green3"/>
    </spatial>
  </tendon>

</mujoco>


Python:
import mujoco
import mujoco_viewer
import matplotlib.pyplot as plt
import numpy as np
import os
from lxml import etree
import mujoco.viewer
import time
import re


R1 = 0.018
R2 = 0.017
```

```
a = 0.054
b = 0.098
c = 0.065

variables = {
    'R1': R1,
    'R2': R2,
    'a': a,
    'b': b,
    'c': c
}

f1 = "D:\@Магистер\SRS\lab 3\lab3.xml"
f2 = "mujoco3_1.xml"


with open(f1, 'rb') as f:
    xml_bytes = f.read()

xml_str = xml_bytes.decode('utf-8')

def evaluate_expression(match):
    expression = match.group(1)
    try:
        result = eval(expression, {}, variables)
        return str(result)
    except Exception as e:
        print(f"Ошибка при вычислении выражения {expression}: {e}")
        return match.group(0)

pattern = r'\{([^}]+)\}'

xml_str_modified = re.sub(pattern, evaluate_expression, xml_str)
xml_bytes_modified = xml_str_modified.encode('utf-8')
tree = etree.fromstring(xml_bytes_modified)

etree.ElementTree(tree).write(f2, pretty_print=True, xml_declaration=True,
encoding='UTF-8')

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

SIMEND = 20
TIMESTEP = 0.001
STEP_NUM = int(SIMEND / TIMESTEP)
timeseries = np.linspace(0, SIMEND, STEP_NUM)

viewer = mujoco_viewer.MujocoViewer(model, data, title="tendon", width=1920,
height=1080)
```

```
while viewer.is_alive:

        mujoco.mj_step(model, data)
        viewer.render()


viewer.close()
```