МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»
(Университет ИТМО)

Международный научно-образовательный центр
Физики наноструктур

Практическая работа № 4
по дисциплине
«Имитационное моделирование робототехнических систем»

по теме:
Инициализация пассивного механизма в среде MuJoCo

Студент: Д.С. Черных
Группа № R4134c *И.О. Фамилия*

Предподаватель: Е.А. Ракшин
Инженер, младший научный сотрудник *И.О. Фамилия*

Санкт-Петербург 2025

Цель работы: исследовать механизм с сенсорами и актуаторами, создать симуляцию и сделать выводы.

Оборудование и программные среды: ПК, Visual studio insider.

Задание:

Составить скрипт на Python для моделирования механизма, согласно варианту.

Ход работы

Согласно варианту в ходе работы будет рассматриваться сухожильный плоский механизм со следующими параметрами:

$R_1 = 0{,}018$ м, $R_2 = 0{,}026$ м, $a = 0{,}034$ м, $b = 0{,}054$ м, $c = 0{,}084$м

Параметры актуаторов приведены далее:

Q1 AMP=25.05 deg, FREQ=3.02 Hz, Bias=5.3 deg

Q2 AMP=41.78 deg, FREQ=2.02 Hz, Bias=16.7 deg

Схема механизма представлена на рисунке 1.

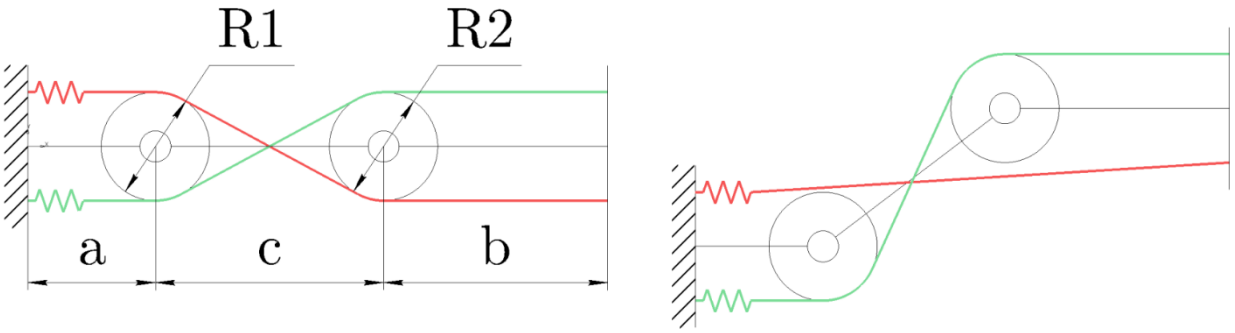

Рисунок 1 – Заданная схема механизма

Программный код решение представлен ниже:

```
import mujoco
import mujoco_viewer
import numpy as np

# Геометрия
LEN_A, LEN_B, LEN_C = 0.034, 0.084, 0.054
R_R1, R_R2 = 0.018, 0.026

# Траектории
AMP1_deg, FREQ1, BIAS1_deg = 25.05, 3.02, 5.3
AMP2_deg, FREQ2, BIAS2_deg = 41.78, 2.02, 16.7

deg2rad = np.pi / 180.0
AMP1 = AMP1_deg * deg2rad
BIAS1 = BIAS1_deg * deg2rad
AMP2 = AMP2_deg * deg2rad
BIAS2 = BIAS2_deg * deg2rad

def make_xml():
    return f"
```

```
<mujoco model="DualActuated_PassiveTendons">
  <option timestep="0.002" gravity="0 0 0"/>

  <worldbody>
    <geom name="floor" type="plane" size="2 2 0.01" rgba="0.6 0.8 0.6 1"/>
    <camera name="cam" pos="0.3 0.8 0.3" xyaxes="-1 0 0 0 0 1"/>

    <!-- Sites на полу -->
    <site name="floor_red"   pos="-{R_R1} 0 0"     rgba="1 0 0 1"
size="0.003"/>
    <site name="floor_green" pos=" {R_R1} 0 0"     rgba="0 1 0 1"
size="0.003"/>

    <body name="link_a" pos="0 0 0">
      <geom type="capsule" size="0.005 {LEN_A/2:.3f}" fromto="0 0 0 0 0
{LEN_A}"/>

      <!-- R1 -->
      <body name="R1" pos="0 0 {LEN_A}">
        <joint name="R1_joint" type="hinge" axis="0 1 0" damping="0.01"/>
        <geom type="cylinder" size="{R_R1} 0.01" rgba="0.8 0.6 0.3 1"
euler="90 0 0"/>
        <site name="R1_top"  pos=" {R_R1} 0 0" rgba="0 1 0 1"
size="0.003"/>
        <site name="R1_bot"  pos="-{R_R1} 0 0" rgba="1 0 0 1"
size="0.003"/>

        <!-- link_b -->
        <geom type="capsule" size="0.005 {LEN_B/2:.3f}" fromto="0 0 0 0 0
{LEN_B}" rgba="0.7 0.7 0.9 1"/>

        <!-- R2 -->
        <body name="R2" pos="0 0 {LEN_B}">
          <joint name="R2_joint" type="hinge" axis="0 1 0" damping="0.01"/>
          <geom type="cylinder" size="{R_R2} 0.01" rgba="0.6 0.8 0.3 1"
euler="90 0 0"/>
          <site name="R2_top"  pos=" {R_R2} 0 0" rgba="1 0 0 1"
size="0.003"/>
          <site name="R2_bot"  pos="-{R_R2} 0 0" rgba="0 1 0 1"
size="0.003"/>

          <!-- link_c -->
          <body name="link_c" pos="0 0 0">
            <geom type="capsule" size="0.004 {LEN_C/2:.3f}" fromto="0 0 0 0
0 {LEN_C}" rgba="0.5 0.8 0.8 1"/>
```

```
                <!-- right_wall -->
                <body name="right_wall" pos="0 0 {LEN_C}">
                  <geom type="box" size="0.1 0.05 0.005" rgba="0.5 0.5 0.8 1"/>
                  <site name="wall_red"   pos="-0.03 0 0.0025" rgba="0 1 0 1"
size="0.003"/>
                  <site name="wall_green" pos=" 0.03 0 0.0025" rgba="1 0 0 1"
size="0.003"/>
                </body>
              </body>
            </body>
          </body>
        </body>
      </worldbody>

      <tendon>
        <spatial name="tendon_red" width="0.0025" rgba="1 0 0 0.8"
limited="false">
          <site site="floor_red"/>
          <site site="R1_bot"/>
          <site site="R2_top"/>
          <site site="wall_green"/>
        </spatial>
        <spatial name="tendon_green" width="0.0025" rgba="0 1 0 0.8"
limited="false">
          <site site="floor_green"/>
          <site site="R1_top"/>
          <site site="R2_bot"/>
          <site site="wall_red"/>
        </spatial>
      </tendon>

      <actuator>
        <motor name="motor_R1" joint="R1_joint" ctrllimited="true" ctrlrange="-
10 10"/>
        <motor name="motor_R2" joint="R2_joint" ctrllimited="true" ctrlrange="-
10 10"/>
      </actuator>

      <sensor>
        <jointpos name="q1" joint="R1_joint"/>
        <jointpos name="q2" joint="R2_joint"/>
      </sensor>
    </mujoco>
    '''.encode()
```

```
        # Запуск
        model = mujoco.MjModel.from_xml_string(make_xml())
        data = mujoco.MjData(model)
        viewer = mujoco_viewer.MujocoViewer(model, data, width=900,
height=700)

        # Индексы
        jnt_R1 = mujoco.mj_name2id(model, mujoco.mjtObj.mjOBJ_JOINT,
"R1_joint")
        jnt_R2 = mujoco.mj_name2id(model, mujoco.mjtObj.mjOBJ_JOINT,
"R2_joint")
        sen_q1 = mujoco.mj_name2id(model, mujoco.mjtObj.mjOBJ_SENSOR,
"q1")
        sen_q2 = mujoco.mj_name2id(model, mujoco.mjtObj.mjOBJ_SENSOR,
"q2")

        addr_q1 = model.jnt_qposadr[jnt_R1]
        addr_q2 = model.jnt_qposadr[jnt_R2]
        addr_dq1 = model.jnt_dofadr[jnt_R1]
        addr_dq2 = model.jnt_dofadr[jnt_R2]

        # PID + ограничение момента
        kp1, kd1 = 800.0, 40.0
        kp2, kd2 = 1000.0, 50.0
        tau_max = 8.0

        data.qpos[addr_q1] = BIAS1
        data.qpos[addr_q2] = BIAS2
        mujoco.mj_forward(model, data)

        t = 0.0
        dt = model.opt.timestep

        print("Запуск")
        print(f"   q₁(t) = {AMP1_deg:.2f}°·sin({FREQ1:.2f}·2π·t) +
{BIAS1_deg:.1f}°")
        print(f"   q₂(t) = {AMP2_deg:.2f}°·sin({FREQ2:.2f}·2π·t) +
{BIAS2_deg:.1f}°")

        try:
            while viewer.is_alive:
                # Чтение
                q1 = data.sensor(sen_q1).data[0]
                q2 = data.sensor(sen_q2).data[0]
                dq1 = data.qvel[addr_dq1]
                dq2 = data.qvel[addr_dq2]
```

```
        # Желаемые траектории
        q1_des = AMP1 * np.sin(2 * np.pi * FREQ1 * t) + BIAS1
        q2_des = AMP2 * np.sin(2 * np.pi * FREQ2 * t) + BIAS2

        # Управление с ограничением
        tau1 = kp1 * (q1_des - q1) - kd1 * dq1
        tau2 = kp2 * (q2_des - q2) - kd2 * dq2
        if abs(tau1) > tau_max: tau1 = np.sign(tau1) * tau_max
        if abs(tau2) > tau_max: tau2 = np.sign(tau2) * tau_max

        data.ctrl[0] = tau1
        data.ctrl[1] = tau2

        mujoco.mj_step(model, data)
        t += dt

        # Лог
        if int(t * 2) % 2 == 0 and abs(t - round(t, 1)) < dt * 1.5:
            print(
                f"t={t:5.2f}s | "
                f"q₁={np.rad2deg(q1):6.2f}° ({np.rad2deg(q1_des):6.2f}°) | "
                f"q₂={np.rad2deg(q2):6.2f}° ({np.rad2deg(q2_des):6.2f}°)",
                end='\r'
            )

        viewer.render()

except KeyboardInterrupt:
    pass
finally:
    viewer.close()
    print("\ Симуляция завершена.")
```

На рисунке 1 представлен результат работы данного кода.

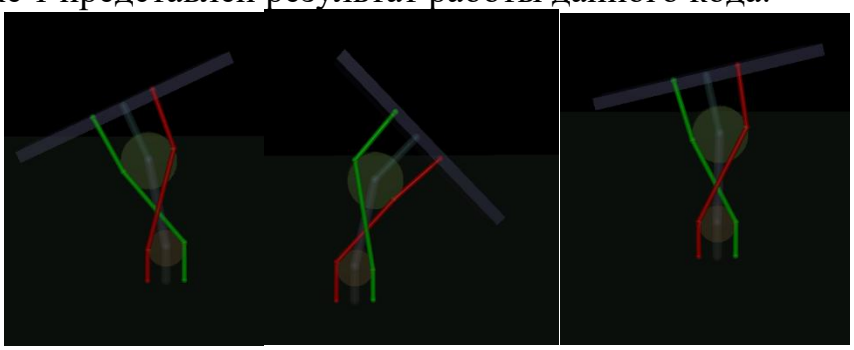

Рисунок 1 – Результат работы кода в трех положениях

Вывод

В ходе работы был разработан код симуляции плоского сухожильного механизма в программной среде Mujoco с актуаторами и сенсорами