Министерство образования и науки Российской Федерации ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧЕРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Отчёт по практической работе №3

по дисциплине

«Имитационное моделирование робототехнических систем»

Отчёт выполнила: Румянцева А.В. (340890)

Преподаватель: Ракшин Е.А. (373529)

Дата выполнения: 17.11.2025

Санкт-Петербург, 2025

1. Постановка задачи и цель работы

В данной работе предложено воссоздать модель по варианту в среде MuJoCo с использованием Python, снять данные о положении суставов, а также задать управление.

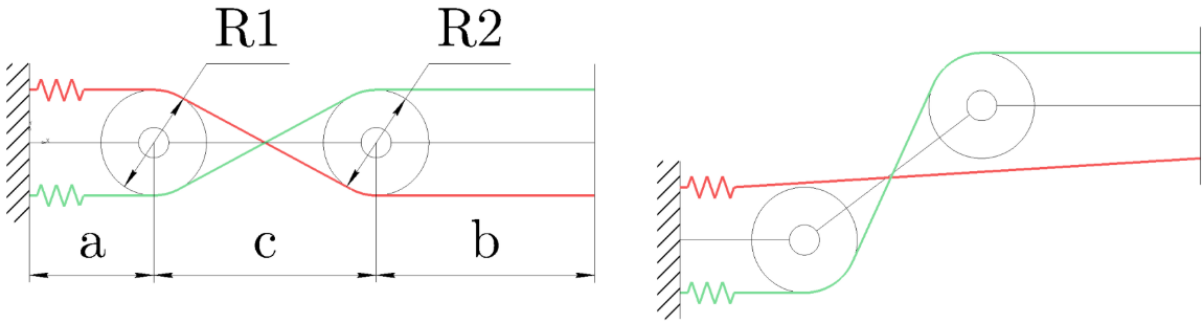

Рисунок 1. Два вида схем из технического задания

2. Создание модели XML

a. Выполнение работы

Сразу стоит отметить, что выполнение работы не считается полностью корректным из-за способа синтеза «сухожилий» (далее tendons). На рисунке 1 видно наличие коллизии с суставами, однако из-за особенностей tendons, они сами по себе не имеют коллизию с другими элементами и geom'ами. В ходе выполнения работы было две идеи по реализации модели в MuJoCo: привязать к tendons множество сфер, имитируя коллизию или отступить от технического задания и насильно привязать tendons по касательной. В виду ограниченного времени выполнения практического задания, отсутствия опыта в программном обеспечении и вынужденной смены среды разработки с Google Colab на Visual Studio, было решено использовать второй вариант выполнения.

Перед началом работы следует разобрать схему в соответствии с вариантом:

$$\mathrm{R}_1 = 0.031, \qquad R_2 = 0.021, \qquad a = 0.085, \qquad b = 0.082, \qquad c = 0.09$$

На схеме на рисунке 1 имеется стенка, откуда начинаются tendons. Далее необходимо два эллипсоида (в ходе работы были использованы именно они из предположения, что между tendons – диски, а не сферы) на расстоянии a для R_1 и $a + c$ для R_2. Окончание tendons, соответственно, в конце суммарного отрезка $a + c + b$. На схеме также видно, что центры дисков расположены на одном отрезке. Сами диски являются суставами типа slide, то есть они способны двигаться только в одной оси. Исходя из правого изображения на рисунке 1, диски двигаются по оси z, но при этом статичны по x. Информации по движению относительно y нет, поэтому было сделано предположение, что диски по этой оси тоже статичны. Также видно, что сами tendons имеют демпферы.

b. Результат

Аналитический анализ схемы окончен, на котором и основана модель XML (приложение 1) для симуляции. В исходном положении схема выглядит следующим образом, показанном на рисунке №2.

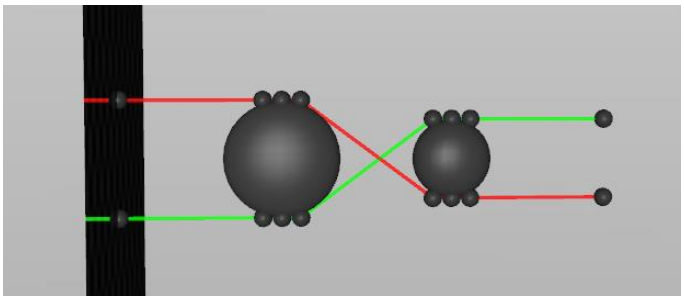

Рисунок 2. Схема в MuJoCo

3. Снятие данных

a. Выполнение работы

После успешного (частично, о причинах написано выше) создания схемы, было решено приступить к написанию кода, которое позволило бы снять данные о положении дисков. Изначально, согласно материалу данному на практических занятиях, планировалось использовать специальную библиотеку matplotlib.pyplot, но по неизвестным причинам эта библиотека не работала ни в Colab, ни в Visual Studio. Более того, Colab отказывался запускаться и заканчивал работу ошибкой из-за бага в одной из библиотек (GLWF) без возможности её обновления. Из-за совокупности этих проблем, было решено использовать Visual Studio и вместо построения графиков в Python, был написан способ экспорта данных в csv на манер предыдущей работы. Весь код описан в приложении №2. Способ снятия данных прост: использовался встроенный внутри дисков site, с которого считывались данные о положении по оси x, по оси z, а также дополнительно снимались данные о времени. В виду особенности типа суставов, никакого изменения по оси x не будет, ведь их положение в этой оси статично.

b. Результат

На рисунке №3 представлен график положения по оси z в течение времени при ручном изменении положения дисков.

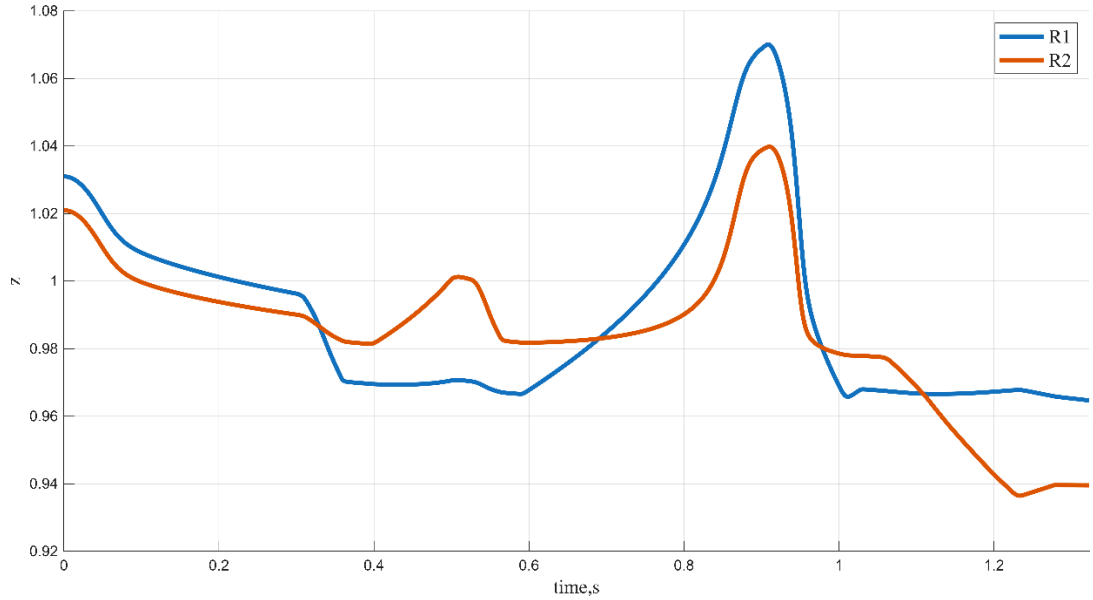

Рисунок 3. Положение дисков с течением времени

Как уже упоминалось ранее, tendons имеют демпфирование. Было решено проверить, как оно влияет на изменение положения дисков и их зависимость друг от друга.

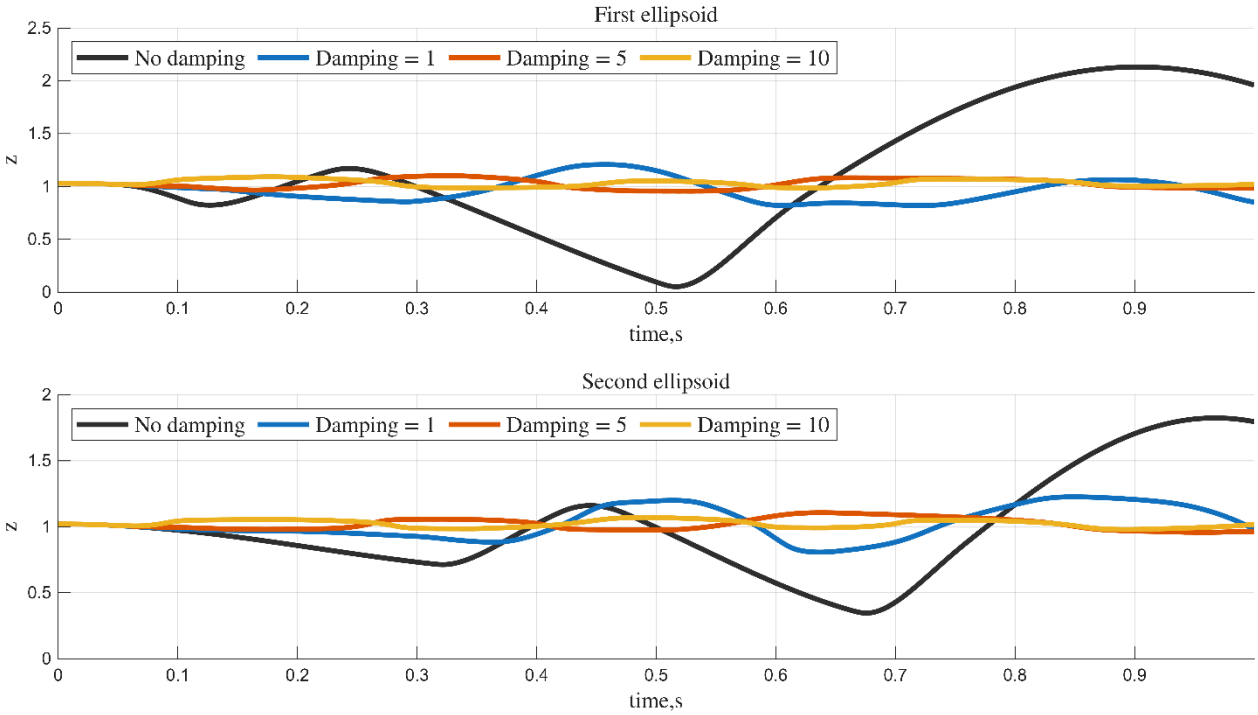

Рисунок 4. Сравнение положения при различном демпфировании

Как видно из рисунка №4, при отсутствии демпфирования разброс положения гораздо больше, чем с ним. И чем больше значения, тем более жёсткими получаются tendons.

4. Управление и актуаторы

a. Выполнение работы

Управление задаётся с помощью «актуаторов» и подаются на суставы. В ходе работы было принято решение повторить пример с заданной синусоидой и повторением положения за ней. Сами по себе актуаторы прописаны в модели XML (приложение №1), а синусоида в коде Python (приложение №2). В качестве исследования было принято решение подать управление на каждый из суставов отдельно и посмотреть, как оно влияет на положение каждого из дисков и проверить выводы о том, какой диск влияет на другой сильнее. Значение демпфирования было решено оставить равным десяти.

b. Результат

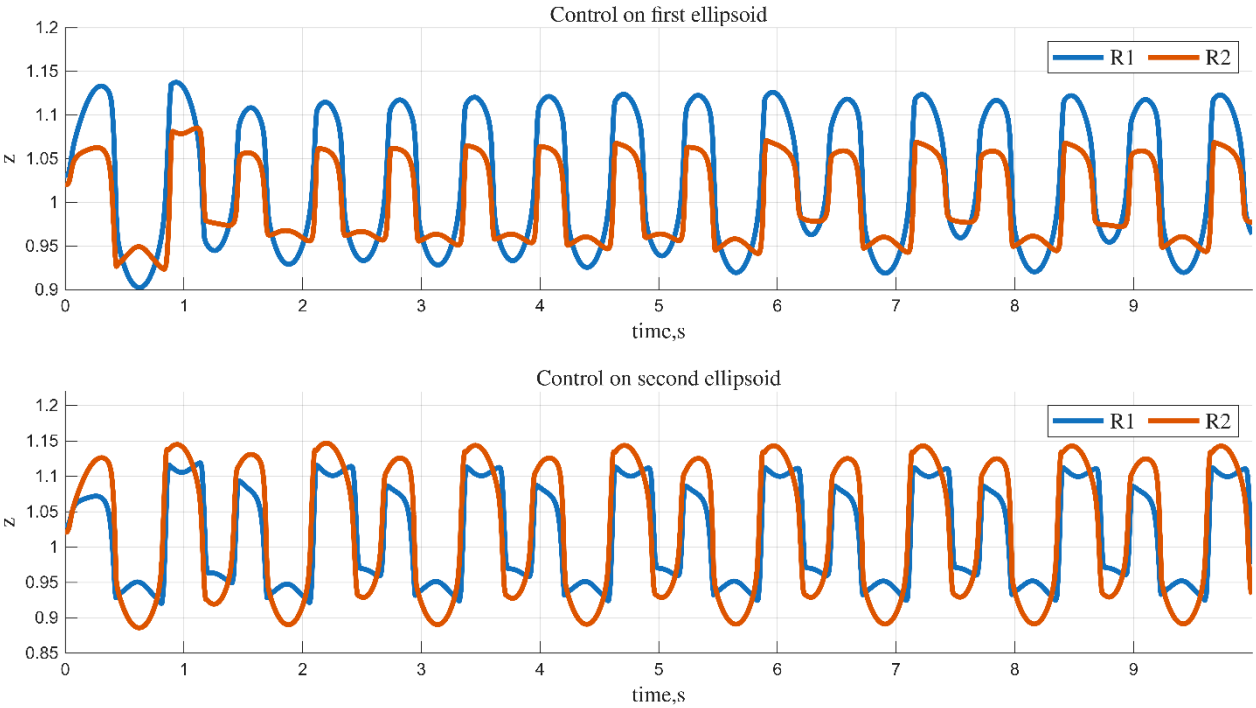

Рисунок 5. Управление на разных дисках

5. Выводы

В ходе работы были исследованы методы создания имитационных моделей в среде MuJoCo с подкреплением в виде кода на языке Python. По итогу работы следует отметить, что несмотря на отсутствие теоретического разбора понятия «сухожилий» (tendons), собственного изучения хватило для реализации модели, хоть и не так идеально, как хотелось изначально. Сами по себе tendons действительно играют роль некой «резинки» с ожидаемыми эффектами. Единственное, что не удалось выполнить в работе, как уже говорилось ранее – полное соответствие техническому заданию и назначение коллизии с дисками.

6. Приложения

Приложение №1. Схема XML

```
<mujoco>

    <option timestep="1e-4"/>
```

```
    <option gravity="0 0 -9.8"/>

    <asset>
        <texture type="skybox" builtin="gradient" rgb1="1 1 1" rgb2="0.5 0.5
0.5" width="265" height="256"/>
        <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>

        <camera name="upper view" pos="0 0 1" euler="90 0 0"/>

        <body name = "wall" pos = "0 0 1" euler = "0 90 0">
            <geom type="plane" size="0.2 0.1 0.2" material="grid"/>
            <site name = "beginning1" pos = "0.031 0 0"/>
            <site name = "beginning2" pos = "-0.031 0 0"/>
            <site name = "end1" pos = "0.021 0 0.257"/>
            <site name = "end2" pos = "-0.021 0 0.257"/>
        </body>

        <body name = "R1" pos="0.085 0 1">
            <joint name = "jointR1"  type="slide" axis = "0 0 1"/>
            <geom type = "ellipsoid" size = "0.031 0.01 0.031"/>
            <site name="R1_site" pos="0 0 0"/>

            <site name = "corner1up" pos = "0 0 0.031" size = "0.005"/>
            <site name = "corner1upleft" pos = "-0.01 0 0.031" size = "0.005"/>
            <site name = "corner1upright" pos = "0.01 0 0.031" size = "0.005"/>

            <site name = "corner1down" pos = "0 0 -0.031" size = "0.005"/>
            <site name = "corner1downleft" pos = "-0.01 0 -0.031" size =
"0.005"/>
            <site name = "corner1downright" pos = "0.01 0 -0.031" size =
"0.005"/>
        </body>

        <body name = "R2" pos="0.175 0 1">
            <joint name = "jointR2" type="slide" axis = "0 0 1"/>
            <geom type = "ellipsoid" size = "0.021 0.01 0.021"/>
            <site name="R2_site" pos="0 0 0"/>

            <site name = "corner2up" pos = "0 0 0.021" size = "0.005"/>
            <site name = "corner2upleft" pos = "-0.01 0 0.021" size = "0.005"/>
            <site name = "corner2upright" pos = "0.01 0 0.021" size = "0.005"/>

            <site name = "corner2down" pos = "0 0 -0.021" size = "0.005"/>
            <site name = "corner2downleft" pos = "-0.01 0 -0.021" size =
"0.005"/>
```

```
            <site name = "corner2downright" pos = "0.01 0 -0.021" size =
"0.005"/>
        </body>
    </worldbody>

    <tendon>
        <spatial name="tendon1" width="0.001" springlength="0.1" damping="10"
rgba = "255 0 0 0.55">
            <site site = "beginning2"/>

            <site site = "corner1upleft"/>
            <site site = "corner1up"/>
            <site site = "corner1upright"/>

            <site site = "corner2downleft"/>
            <site site = "corner2down"/>
            <site site = "corner2downright"/>

            <site site = "end1"/>
        </spatial>
    </tendon>

    <tendon>
        <spatial name="tendon2" width="0.001" springlength="0.1" damping="10"
rgba = "0 255 0 0.55">>
            <site site = "beginning1"/>

            <site site = "corner1downleft"/>
            <site site = "corner1down"/>
            <site site = "corner1downright"/>

            <site site = "corner2upleft"/>
            <site site = "corner2up"/>
            <site site = "corner2upright"/>

            <site site = "end2"/>
        </spatial>
    </tendon>

    <actuator>
        <position name="For R1" joint="jointR1"/>
        <position name="For R2" joint="jointR2"/>
    </actuator>

    <sensor>
        <framepos objtype="body" objname="R1"/>
        <framepos objtype="body" objname="R2"/>
    </sensor>


</mujoco>
```

Приложение №2. Код Python

```python
import mujoco
import mujoco_viewer
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import os
import mujoco.viewer
import time

f1 = "MujocoLab3.xml"

model = mujoco.MjModel.from_xml_path(f1)
data = mujoco.MjData(model)

def set_torque(mj_data, KP, KV, theta):
    data.ctrl[1] = KP * (-mj_data.qpos[0] + theta) + KV * (0 - mj_data.qvel[0])

SIMEND = 100
TIMESTEP = 0.001
STEP_NUM = int(SIMEND / TIMESTEP)
timeseries = np.linspace(0, SIMEND, STEP_NUM)

T = 1 # [s]
FREQ = 1/T # [Hz]
AMP = 2 # [rad]
BIAS = 0 # [rad]

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

position_time = []

R1_position_x = []
R1_position_z = []

R2_position_x = []
R2_position_z = []

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

for i in range(STEP_NUM):
    if viewer.is_alive:
        set_torque(data, 5, 1, theta_des[i])
```

```
        current_time = data.time
        position_time.append(current_time)

        position_R1 = data.site_xpos[5]
        R1_position_x.append(position_R1[0])
        R1_position_z.append(position_R1[2])

        position_R2 = data.site_xpos[12]
        R2_position_x.append(position_R2[0])
        R2_position_z.append(position_R2[2])

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

    else:
        break
viewer.close()

midlength = int(STEP_NUM/2)

df = pd.DataFrame({
    'time': position_time,
    'R1_x': R1_position_x,
    'R1_z': R1_position_z,
    'R2_x': R2_position_x,
    'R2_z': R2_position_z
})

csv_filename = "Task3_positions_data.csv"
df.to_csv(csv_filename, index=False)
```