Национальный исследовательский университет ИТМО
Факультет систем управления и робототехники

Отчет
о выполнении практического задания №4
по дисциплине «Имитационное моделирование робототехнических систем»

Выполнил
студент гр. R4134c
ИСУ 505887 А. С. Абраменко

Преподаватель Е. А. Ракшин

«___» _____ 2025 г.

Санкт-Петербург
2025

Задание

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).

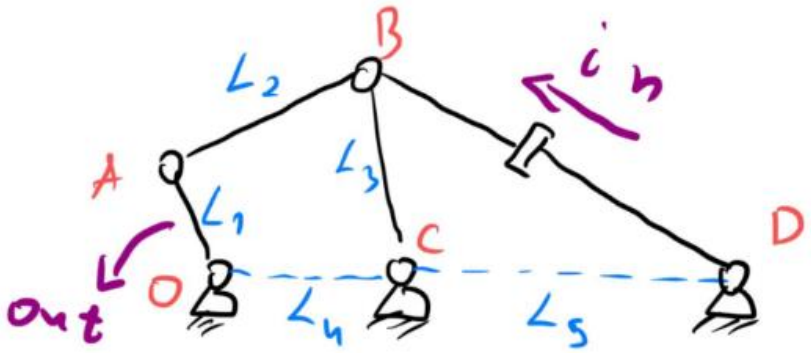
2. Modify the .xml file by adding <actuator> and <sensor> containers (look at the examples in the previous task).

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 control sequence goes beyond workspace of the mechanism, decrease the amplitude and tune bias only if needed.

Variant 2 - Optimus' knee closed-chain mechanism:



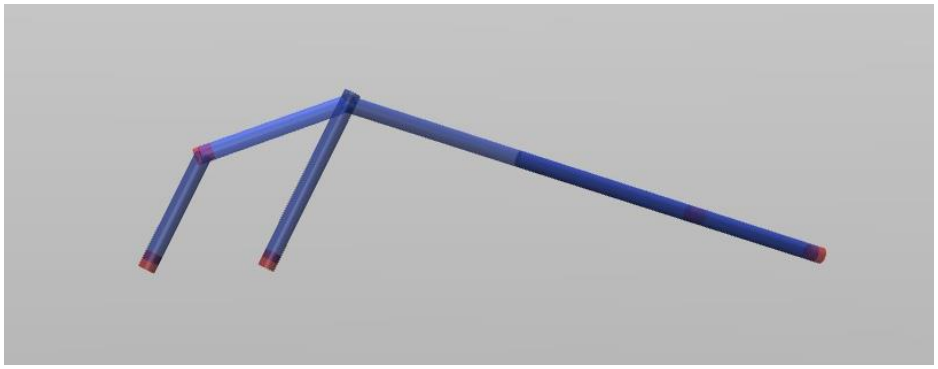
Sine wave parameters:

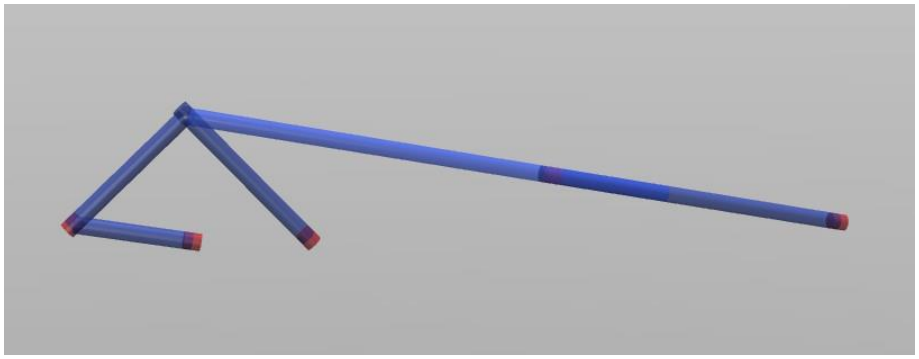
AMP, deg	FREQ, Hz	BIAS, deg
10.51	3.47	24

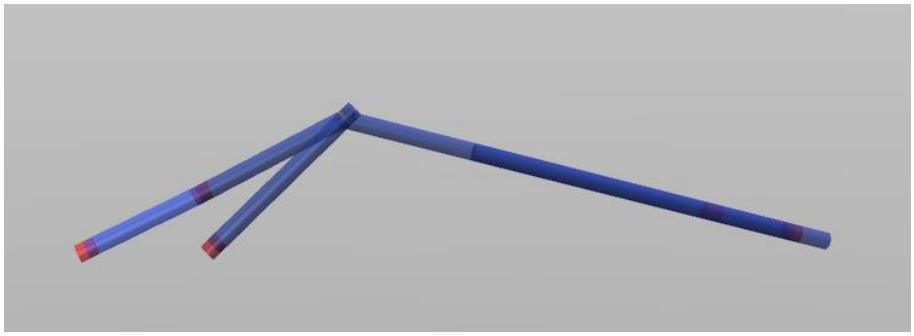
Ход работы

Программная реализация тел в соответствии с заданными исходными размерами посредством mujoco представлена в лабораторной работе 3.

Результаты работы программы, разработанной в лабораторной работе 3, представлены на рисунках ниже:







В соответствии с заданным вариантом, выходные значения необходимо снимать с положения звена OA (сочленение A):

```
<sensor>
      <framepos objtype="site" objname="sA"/>
</sensor>
```

Входное воздействие, необходимое для работы меанизма в соответствии с заданным законом, подается на слайдер:

```
<actuator>
      <position name="slider" joint="slider"/>
</actuator>
```

Программа для задания входного воздействия с использованием заданных параметров представлена ниже. Функция для входного воздействия подразумевает управление положением посредством ПД-регулятора.

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

```
SIMEND = 50
TIMESTEP = 0.001
STEP_NUM = int(SIMEND / TIMESTEP)
timeseries = np.linspace(0, SIMEND, STEP_NUM)

FREQ = 3.47 # [Hz]
AMP =  np.deg2rad(10.51)
BIAS = np.deg2rad(24)

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

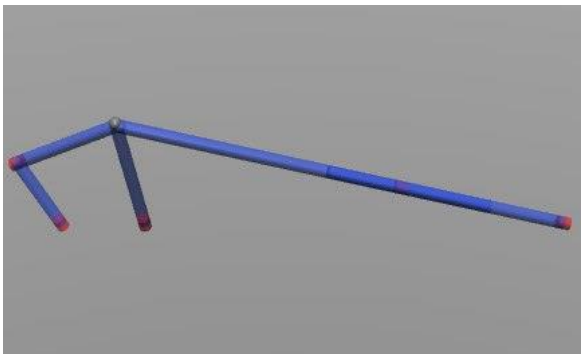
Для вывода выходных значений также реализовано:

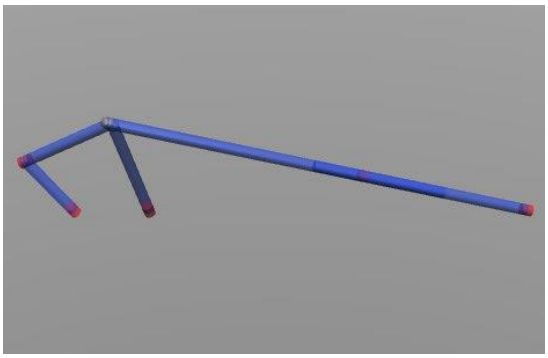
```
EE_pos_x = []
EE_pos_z = []
...
pos_EE = data.site_xpos[0]
EE_pos_x.append(pos_EE[0])
EE_pos_z.append(pos_EE[2])
...
plt.clf()
plt.plot(EE_pos_x[5000:], EE_pos_z[5000:], '-', linewidth=2, label='P')
plt.title('End-effector trajectory', fontsize=12, fontweight='bold')
plt.legend(loc='upper left')
plt.xlabel('X-Axis [m]')
plt.ylabel('Z-Axis [m]')
plt.axis('equal')
plt.grid()
plt.draw()
```

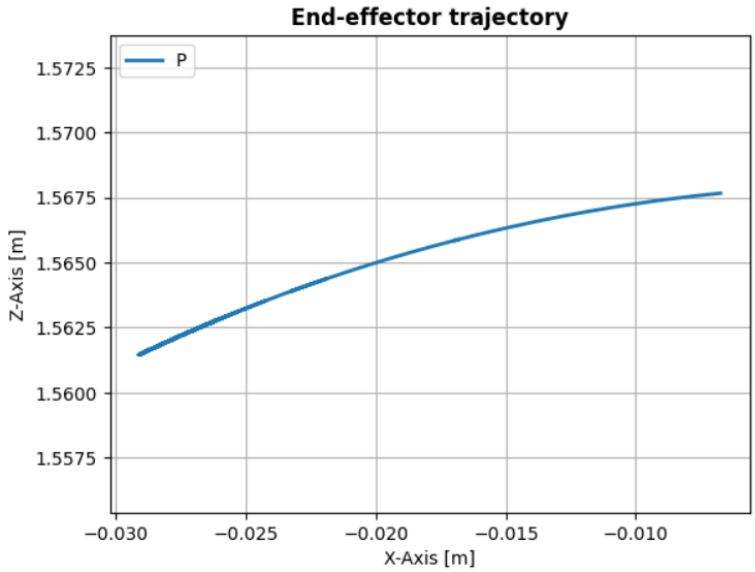
Далее был также настроен ПД-регулятор: KP = 50, KV = 10

Результаты работы программы

Промежуточные положения механизма, а также график выходных значений представлены на рисунках ниже.







Выводы

В процессе выполнения данной работы был реализован актуатор (посредством контейнера actuator), позволяющий выполнять движение механизма по заданному закону, далее был настроен ПД-регулятор положения выходного звена. Также был реализован вывод выходных значений симуляции посредством контейнера sensor.

ПРИЛОЖЕНИЕ

1. opmimus.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>

    <light pos="0 0 10"/>
    <geom type="plane" size="0.5 0.5 0.1" material="grid"/>

    <camera name="side view" pos="0.1 -1.5 1.0" euler="90 0 0" fovy="60"/>
    <camera name="upper view" pos="0 0 1.5" euler="0 0 0"/>

    <body name="OAB1" pos="0 0 1.5" euler="0 0 0">

            <joint name="O" type="hinge" axis="0 -1 0" stiffness="0"
springref="0" damping="0"/>
            <geom name="point O" type="cylinder" pos="0 0 0" size="0.005 0.005"
rgba="0.89 0.14 0.16 0.5" euler="0 0 0" contype="0"/>
            <geom name="link OA" type="cylinder" pos="0 0 0.034" size="0.005
0.034" rgba="0.21 0.32 0.82 0.5" euler="0 0 0" contype="0"/>
            <site name="sA" size="0.005" pos="0 0 0.068"/>

            <body name="AB1" pos="0 0 0.068" euler="0 0 0">

                <joint name="A" type="hinge" axis="0 -1 0" stiffness="0"
springref="0" damping="0.1"/>
                <geom name="point B" type="cylinder" pos="0 0 0" size="0.005
0.005" rgba="0.89 0.14 0.16 0.5" euler="0 0 0" contype="0"/>
                <geom name="link AB1" type="cylinder" pos="0 0 0.0422"
size="0.005 0.0442" rgba="0.21 0.32 0.82 0.5" euler="0 0 0" contype="0"/>
                <site name="sC1" size="0.005" pos="0 0 0.0884"/>

            </body>

        </body>
```

```
    <body name="CB2" pos="0.068 0 1.5" euler="0 0 0">

        <joint name="C" type="hinge" axis="0 -1 0" stiffness="0" springref="0"
damping="0.1"/>
        <geom name="point C" type="cylinder" pos="0 0 0" size="0.005 0.005"
rgba="0.89 0.14 0.16 0.5" euler="0 0 0" contype="0"/>
        <geom name="link CB2" type="cylinder" pos="0 0 0.051" size="0.005 0.051"
rgba="0.21 0.32 0.82 0.5" euler="0 0 0" contype="0"/>
        <site name="sC2" size="0.005" pos="0 0 0.102"/>

    </body>

    <body name="DFB3" pos="0.408 0 1.5" euler="0 0 0">

        <joint name="D" type="hinge" axis="0 -1 0" stiffness="0" springref="0"
damping="0"/>
        <geom name="point D" type="cylinder" pos="0 0 0" size="0.005 0.005"
rgba="0.89 0.14 0.16 0.5" euler="0 0 0" contype="0"/>
        <geom name="link DB3" type="cylinder" pos="0 0 0.1" size="0.005 0.1"
rgba="0.21 0.32 0.82 0.5" euler="0 0 0" contype="0"/>

        <body name="FB3" pos="0 0 0.075" euler="0 0 0">

            <joint name="slider" type="slide" axis="0 0 1" limited="true"
range="-0.2 0.2" stiffness="0" springref="0" damping="0"/>
            <geom name="point B3" type="cylinder" pos="0 0 0" size="0.005 0.005"
rgba="0.89 0.14 0.16 0.5" euler="0 0 0" contype="0"/>
            <geom name="link FB3" type="cylinder" pos="0 0 0.075" size="0.005
0.15" rgba="0.21 0.32 0.82 0.5" euler="0 0 0" contype="0"/>
            <site name="sC3" size="0.005" pos="0 0 0.225"/>

        </body>

    </body>

    </worldbody>

    <equality>
        <connect site1="sC1" site2="sC3"/>
        <connect site1="sC1" site2="sC2"/>
    </equality>

    <sensor>
        <framepos objtype="site" objname="sA"/>
    </sensor>

    <actuator>
        <position name="slider" joint="slider"/>
    </actuator>
```

```
</mujoco>
```

2. optimus.ipynb

```
import mujoco
import mujoco.viewer
import matplotlib.pyplot as plt
import numpy as np
import os
from lxml import etree
import time
f1 = "optimus4.xml"

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

print(f"Number of actuators: {model.nu}")

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

SIMEND = 50
TIMESTEP = 0.001
STEP_NUM = int(SIMEND / TIMESTEP)
timeseries = np.linspace(0, SIMEND, STEP_NUM)

FREQ = 3.47 # [Hz]
AMP =  np.deg2rad(10.51)
BIAS = np.deg2rad(24)

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

EE_pos_x = []
EE_pos_z = []
with mujoco.viewer.launch_passive(model, data) as viewer:
    for i in range(STEP_NUM):
        set_torque(data, 50, 10, theta_des[i])

        pos_EE = data.site_xpos[0]
        EE_pos_x.append(pos_EE[0])
        EE_pos_z.append(pos_EE[2])

        mujoco.mj_step(model, data)
        viewer.sync()
        time.sleep(0.001)
viewer.close()

plt.clf()
plt.plot(EE_pos_x[5000:], EE_pos_z[5000:], '-', linewidth=2, label='P')
```

```
plt.title('End-effector trajectory', fontsize=12, fontweight='bold')
plt.legend(loc='upper left')
plt.xlabel('X-Axis [m]')
plt.ylabel('Z-Axis [m]')
plt.axis('equal')
plt.grid()
plt.draw()
```