МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

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

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №4

по дисциплине «Имитационное моделирование робототехнических систем»

Выполнил:
студент групп № R4134c
Фомин В.М.

Проверил:
преподаватель
Ракшин Е.А.

Санкт-Петербург, 2025

Задание

1. Дополнить предыдущую лабораторную работу: к созданной модели в предыдущем задании необходимо добавить приводы (актюаторы). Для сухожильного механизма — два привода (q1 и q2).

2. Изменить файл .xml, добавив контейнеры <actuator> и <sensor>.

3. Определить управляющее воздействие с помощью ПД-регулятора. Управление задается как q^{des} = AMP * sin(FREQ * t) + BIAS. Параметры синусоиды приведены в таблице. Если управляющий сигнал выходит за пределы рабочей зоны механизма, уменьшите амплитуду и, только если это необходимо, скорректируйте смещение (bias).

Ход работы

1. Возьмём параметры нашего желаемого сигнала из таблицы 1.

Таблица 1: Параметры желаемого сигнала

q_i	AMP, deg	FREQ, Hz	BIAS, deg
q_1	52.89	3.92	19.4
q_2	50.54	1.08	19

2. Приведём выражение для нашего воздействия, заданного в виде:

$$\tau = K_p \cdot e + K_d \cdot \dot{e},$$

где $\dot{e} = \dot{q}^{des} - \dot{q}$, $K_p = [150\ 260]$, $K_d = [3\ 4]$.

3. Результаты программной реализации представлены на рисунках 1-3:

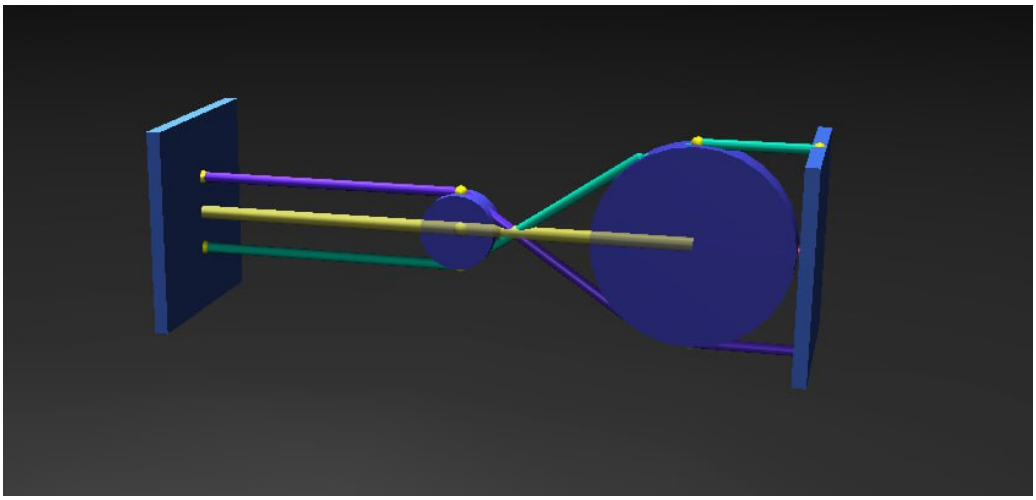

Рис. 1: Конфигурация RR-механизма

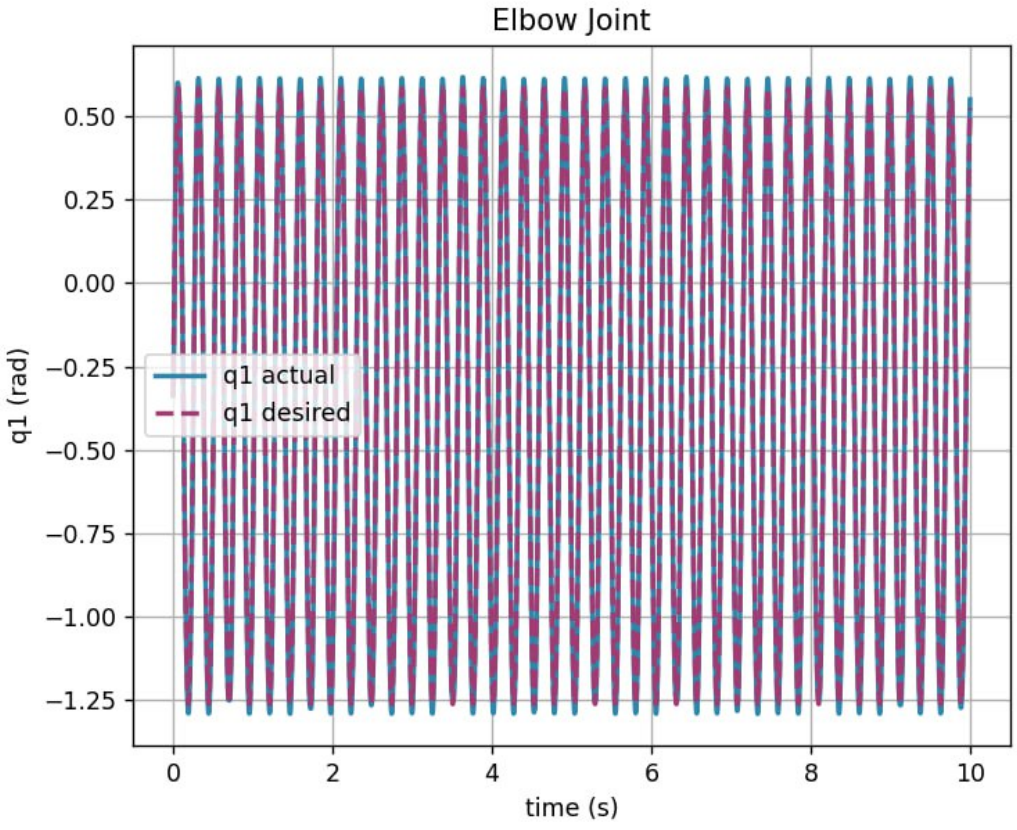

Рис. 2: Результат по первому приводу

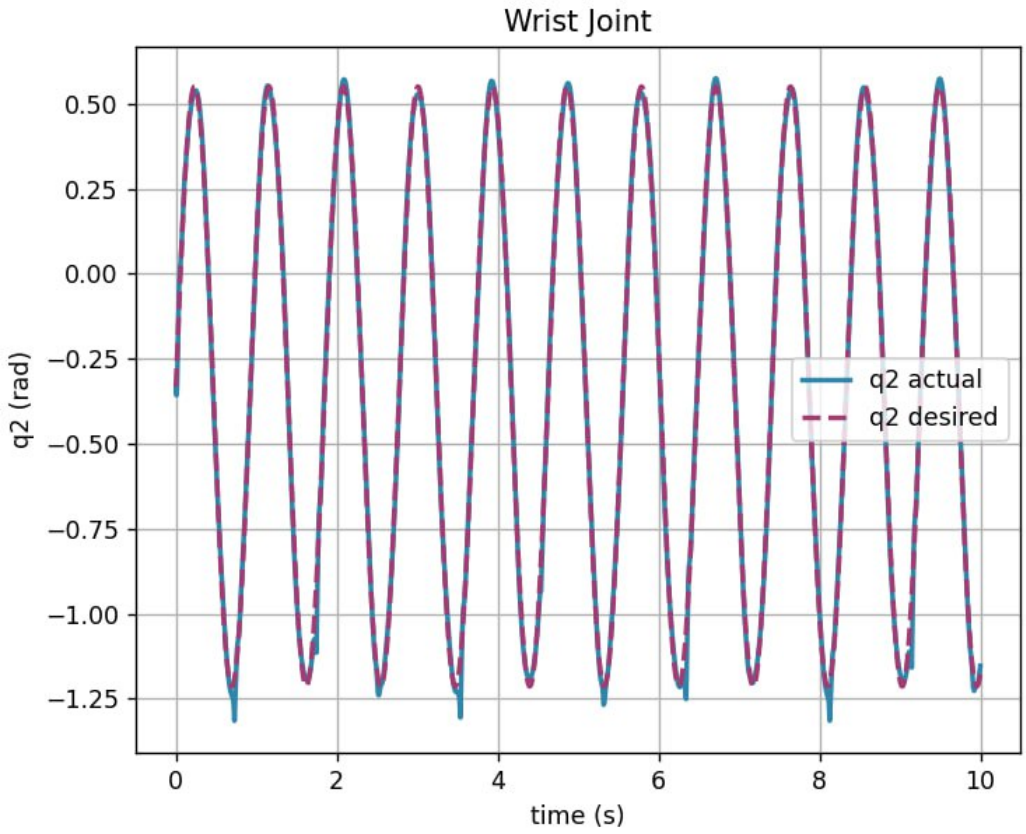

Рис. 3: Результат по второму приводу

Для получения данных было добавлено несколько строк в xml файл:

```
<actuator>
        <motor name="q1_actuator" tendon="tendon1" gear="1"
           ctrllimited="true" ctrlrange="-50 50"/>
        <motor name="q2_actuator" tendon="tendon2" gear="1"
           ctrllimited="true" ctrlrange="-50 50"/>
    </actuator>

    <sensor>
        <tendonpos name="q1_sensor" tendon="tendon1"/>
        <tendonpos name="q2_sensor" tendon="tendon2"/>
        <tendonvel name="q1_vel_sensor" tendon="tendon1"/>
        <tendonvel name="q2_vel_sensor" tendon="tendon2"/>
    </sensor>
```

А также был написан следующий код для реализации PD-регулятора и вывода графиков:

```python
import mujoco
import mujoco.viewer
import numpy as np
import matplotlib.pyplot as plt
import time

model_path = r"tendon_test.xml"

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

elbow_id = model.joint('elbow').id
wrist_id = model.joint('wrist').id

elbow_qpos = model.jnt_qposadr[elbow_id]
wrist_qpos = model.jnt_qposadr[wrist_id]

elbow_qvel = model.jnt_dofadr[elbow_id]
wrist_qvel = model.jnt_dofadr[wrist_id]

act1 = model.actuator('m_elbow').id
act2 = model.actuator('m_wrist').id

DEG_TO_RAD = np.pi / 180.0

AMP1  = 52.89 * DEG_TO_RAD
FREQ1 = 3.92
BIAS1 = 19.4 * DEG_TO_RAD

AMP2  = 50.54 * DEG_TO_RAD
FREQ2 = 1.08
BIAS2 = 19.0 * DEG_TO_RAD

Kp = np.array([50.0, 40.0])
Kd = np.array([0.2, 0.2])

ctrl_min = model.actuator_ctrlrange[:, 0]
ctrl_max = model.actuator_ctrlrange[:, 1]

dt = model.opt.timestep
sim_time = 10.0
steps = int(sim_time / dt)
t0 = 0.0

q1_start = AMP1 * np.sin(2*np.pi*FREQ1 * t0) + BIAS1
q2_start = AMP2 * np.sin(2*np.pi*FREQ2 * t0) + BIAS2

data.qpos[elbow_qpos]  = -q1_start
```

```
data.qpos[wrist_qpos]  = -q2_start
data.qvel[elbow_qvel]  = 0
data.qvel[wrist_qvel]  = 0

mujoco.mj_forward(model, data)

t_log  = []
q1_log = []
q2_log = []
q1d_log = []
q2d_log = []

start_time = time.time()

with mujoco.viewer.launch_passive(model, data) as viewer:
    viewer.cam.distance = 0.5
    viewer.cam.azimuth = 45
    viewer.cam.elevation = -20

    for step in range(steps):
        if not viewer.is_running():
            break

        t = step*dt

        q1_des = AMP1 * np.sin(2*np.pi*FREQ1 * t) - BIAS1
        q2_des = AMP2 * np.sin(2*np.pi*FREQ2 * t) - BIAS2

        dq1_des = AMP1 * 2*np.pi*FREQ1 * np.cos(2*np.pi*FREQ1 * t)
        dq2_des = AMP2 * 2*np.pi*FREQ2 * np.cos(2*np.pi*FREQ2 * t)

        q1  = data.qpos[elbow_qpos]
        q2  = data.qpos[wrist_qpos]
        dq1 = data.qvel[elbow_qvel]
        dq2 = data.qvel[wrist_qvel]

        u1 = Kp[0] * (q1_des - q1) + Kd[0] * (dq1_des - dq1)
        u2 = Kp[1] * (q2_des - q2) + Kd[1] * (dq2_des - dq2)

        u1 = float(np.clip(u1, ctrl_min[act1], ctrl_max[act1]))
        u2 = float(np.clip(u2, ctrl_min[act2], ctrl_max[act2]))

        data.ctrl[act1] = u1
        data.ctrl[act2] = u2

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

```

```
            t_log.append(t)
            q1_log.append(q1)
            q2_log.append(q2)
            q1d_log.append(q1_des)
            q2d_log.append(q2_des)

        print("Simulation finished. Close the viewer window to continue..."
            )
        while viewer.is_running():
            time.sleep(0.1)


plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.plot(t_log, q1_log, label="q1 actual", color='#2E86AB', linewidth
    =2)
plt.plot(t_log, q1d_log, '--', label="q1 desired", color='#A23B72',
    linewidth=2)
plt.legend()
plt.grid()
plt.xlabel("time (s)")
plt.ylabel("q1 (rad)")
plt.title("Elbow Joint")

plt.subplot(1, 2, 2)
plt.plot(t_log, q2_log, label="q2 actual", color='#2E86AB', linewidth
    =2)
plt.plot(t_log, q2d_log, '--', label="q2 desired", color='#A23B72',
    linewidth=2)
plt.legend()
plt.grid()
plt.xlabel("time (s)")
plt.ylabel("q2 (rad)")
plt.title("Wrist Joint")

plt.tight_layout()
plt.show()

#
q1_error = np.array(q1d_log) - np.array(q1_log)
q2_error = np.array(q2d_log) - np.array(q2_log)

print(f"Simulation completed: {sim_time} seconds")
print(f"Max q1 error: {np.max(np.abs(q1_error)):.4f} rad")
print(f"Max q2 error: {np.max(np.abs(q2_error)):.4f} rad")
print(f"RMS q1 error: {np.sqrt(np.mean(q1_error**2)):.4f} rad")
print(f"RMS q2 error: {np.sqrt(np.mean(q2_error**2)):.4f} rad")
```

4. В таблице 2 приведены данные об значениии ошибки:

Таблица 2: Ошибки позиционирования суставов

Параметр	**Сустав 1 (q1)**	**Сустав 2 (q2)**
Макс. ошибка, рад	0.1356	0.2136
СКО ошибки, рад	0.0237	0.0248

Вывод

В ходе выполнения практической работы была успешно реализована система управления двухзвенным роботом-манипулятором с использованием ПД-регулятора в среде MuJoCo. Основные достижения включают:

1. **Синтез системы управления** — разработан ПД-регулятор для траекторного управления, обеспечивающий отслеживание заданных синусоидальных траекторий для обоих суставов. Подобранные коэффициенты усиления Kp = array([50.0, 40.0]) и Kd = array([0.2, 0.2]) продемонстрировали эффективное подавление колебаний и обеспечение устойчивости системы.

2. **Верификация модели** — Результаты подтвердили адекватность математической модели и корректность реализации алгоритма управления.