

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

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

Факультет систем управления и робототехники

Отчёт по практической работе №4

по дисциплине
«Имитационное моделирование робототехнических систем»

Студент:
Группа № R4134c

Д.С. Обухова

Преподаватель:
ассистент ФСУиР

Е.А. Ракшин

Санкт Петербург 2025

Начальные данные (#35, ису 336878)

q1			q2		
AMP, deg	FREQ, Hz	BIAS, deg	AMP, deg	FREQ, Hz	BIAS, deg
47.95	3.33	14.9	51.92	3.17	27.3

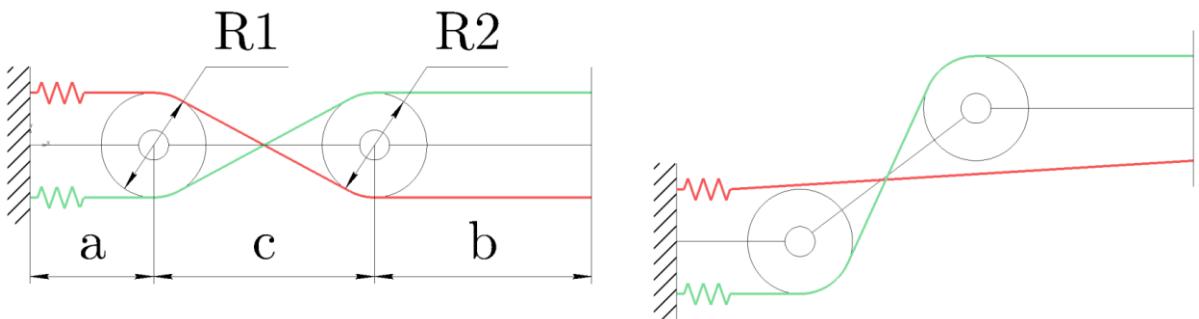


Рисунок 1 - Плоский механизм, соединенный сухожилием 2R.

XML-модель

XML описание включает в себя: звенья, сайты, 2 троса (сухожилия) через точки. Чтобы тросы работали корректно нужно предусмотреть вспомогательные тела и ограничители, фиксирующие положения точек. Ориентация эффектора скреплена жёсткой связью.

В соответствии с заданием в XML-модель были добавлены два мотор-привода (tendon1_motor, tendon2_motor), управляющие длинами тросов. Также добавлены набор сенсоров для измерения текущих длин и скоростей изменения длин тяг, положения и скорости подвижного шкива.

```

<actuator>
  <motor name="act_q1" joint="A" gear="1" ctrlrange="-5 5"/>
  <motor name="act_q2" joint="B" gear="1" ctrlrange="-5 5"/>
</actuator>

<sensor>
  <jointpos name="sens_q1_pos" joint="A"/>
  <jointvel name="sens_q1_vel" joint="A"/>
  <jointpos name="sens_q2_pos" joint="B"/>
  <jointvel name="sens_q2_vel" joint="B"/>
  <framepos objtype="site" objname="effector"/>
</sensor>
```

Рисунок 2 – Дополнительный код для actuator и sensor.

Python-файл

На основе этой модели создан Python-скрипт, выполняющий загрузку модели, инициализацию данных, вычисление управляющего воздействия с помощью ПД-регулятора .

Во время моделирования управляющий сигнал подаётся на оба троса, а положение эфектора считывается с сенсора и используется для построения его траектории.

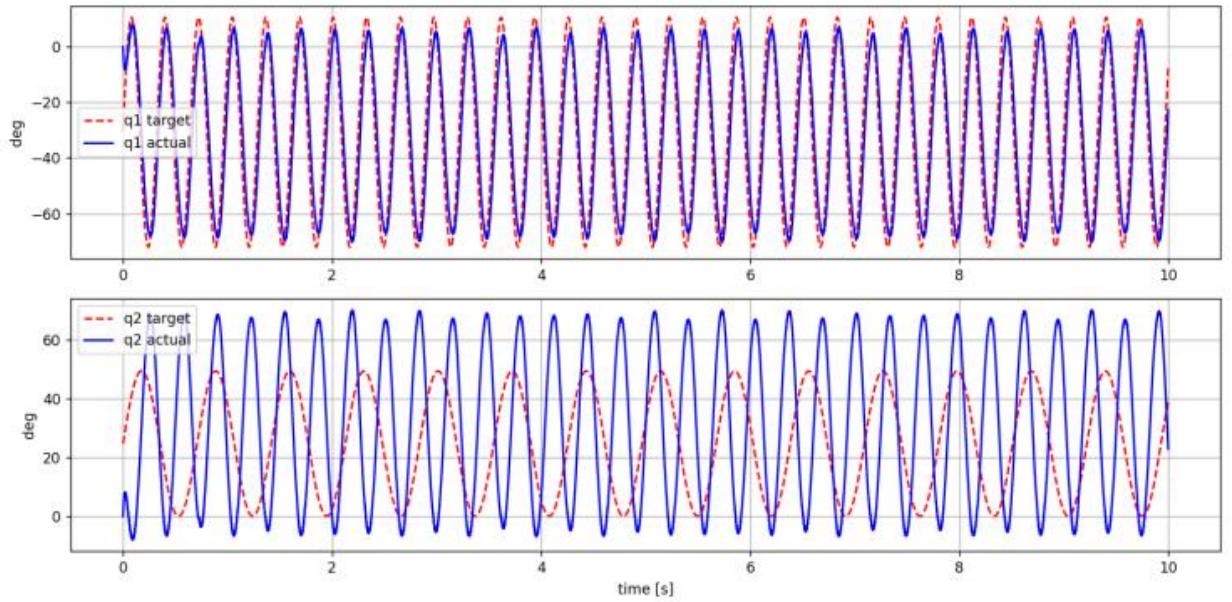


Рисунок 3- Результат моделирования.

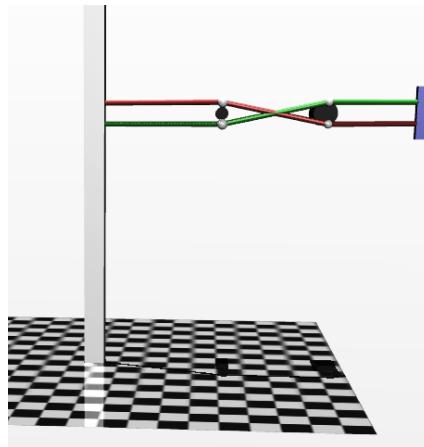


Рисунок 4 – Модель механизма.

Заключение

В ходе выполнения данной практической работы в среде MuJoCo был смоделирован плоский механизм 2R с сухожильным соединением. Управление длинами тяг осуществлялось с помощью ПД-регулятора, формирующего управляющее усилие на основе отклонения текущей длины тяги от заданной синусоидальной траектории. Полученные результаты подтверждают корректность xml файла и его соответствие заданной структуре системы.

Листинги

```
import mujoco
import matplotlib.pyplot as plt
import numpy as np
import os
import mujoco.viewer
import time
from tendon import generate_tendon_xml

class PDController:
    def __init__(self, kp, kd, dt, output_limit=None):
        self.kp=kp
        self.kd=kd
        self.dt=dt
        self.prev_error=0
        self.limit=output_limit

    def calculate(self, target, current):
        error = target - current
        derivative = (error - self.prev_error) / self.dt
        output = self.kp * error + self.kd * derivative
        self.prev_error = error

        if self.limit:
            output = np.clip(output, -self.limit, self.limit)

    return output

params_q1 = {'amp': np.deg2rad(47.95), 'freq':3.33, 'bias':
np.deg2rad(14.9)}
params_q2 = {'amp': np.deg2rad(51.92), 'freq':3.17, 'bias':
np.deg2rad(27.3)}

def get_target(t, params):
    return params['amp']* np.sin(2* np.pi *
params['freq']*t)+params['bias']

R1, R2, a, b, c = 0.011, 0.012, 0.088, 0.047
xml = generate_tendon_xml (R1, R2, a, b, c)
model = mujoco.MjModel.from_xml_string(xml.encode("utf-8"))
data = mujoco.MjData(model)

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

dt = 1e-4
sim_time = 10
steps = int(sim_time/dt)
```

```

pd_q1 = PDController(kp=4.0, kd=0.1, dt=dt, output_limit=8.0)
pd_q2 = PDController(kp=4.0, kd=0.1, dt=dt, output_limit=8.0)

EE_position_x = []
EE_position_z = []

viewer = mujoco.viewer.launch_passive (model, data)

for i in range (STEP_NUM):
    if viewer.is_running:
        t = data.timeseries
        q1_des = get_target(t, params_q1)
        q2_des = get_target (t, params_q2)

        q1_curr = data.sencordata[0]
        q2_curr = data.sencordata[2]

        data.ctrl[0]=pd_q1.calculate(q1_des, q1_curr)
        data.ctrl[1]=pd_q2.calculate(q2_des, q2_curr)

        position_EE = data.sensordata[:3]
        EE_position_x.append(position_EE[0])
        EE_position_z.append(position_EE[2])

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

    else:
        break
viewer.close()

```