Цель работы

1. Добавить приводы для модели, созданной в предыдущей лабораторной работе. Для данного тензорного механизма (рис. 1) необходимо два привода (q_1 и q_2);
2. Изменить файл *.xml*, добавив контейнеры *actuator* и *sensor*;
3. Определить усилие управление с помощью *PID-регулятора*, используя параметры синусоидальной волны из таблицы (табл. 1). В случае выхода за пределы рабочего пространства механизма, уменьшить амплитуду и при необходимости отрегулировать смещение;
4. Запустить симуляцию работы механизма, используя физический движок *MuJoCo (Multi-Joint dynamics with Contact)*.

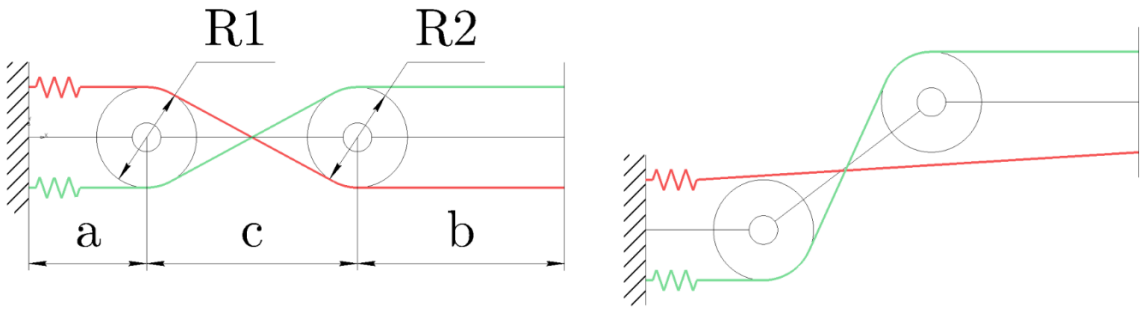

Рисунок 1. плоский механизм с сухожильным соединением, где R_1 – радиус первого шкифа, R_2 – радиус второго шкифа, a – длина первого звена, b – длина второго звена, c – смещение точки крепления сухожилия второго звена

Таблица 1 – Данные геометрических параметров

	AMP, deg	$FREQ$, Hz	$BIAS$, deg
q_1	34.45	3.85	42.1
q_2	11.77	1.61	-31.6

Определение усилия управления с помощью PID-регулятора

Определить усилие PID-регулятора можно по формуле:

$$q^{des} = AMP \cdot \sin(FREQ \cdot t) + BIAS \quad (1)$$

При этом для использования в математических функциях MuJoCo необходимо преобразовывать углы из градусов в радианы. Реализация представлена на рисунке 2.

```
SIMEND = 30
TIMESTEP = 0.001
STEP_NUM = int(SIMEND / TIMESTEP)
timeseries = np.linspace(0, SIMEND, STEP_NUM)

AMP1 = np.deg2rad(34.45)
FREQ1 = 3.85
BIAS1 = np.deg2rad(42.1)
theta1 = AMP1 * np.sin(2 * np.pi * FREQ1 * timeseries) + BIAS1

AMP2 = np.deg2rad(11.77)
FREQ2 = 1.61
BIAS2 = np.deg2rad(-31.6)
theta2 = AMP2 * np.sin(2 * np.pi * FREQ2 * timeseries) + BIAS2
```

Рисунок 2. Фрагмент кода с реализацией функции *deg2rad*

Функционал PID-регулятора для каждого привода реализован и представлен на рисунке 2.

```
11  def set_torque_R1(mj_data, KP, KV, desired_pos):
12      current_pos = mj_data.qpos[0]
13      current_vel = mj_data.qvel[0]
14      data.ctrl[0] = KP * (desired_pos - current_pos) + KV * (0 - current_vel)
15
16  def set_torque_R2(mj_data, KP, KV, desired_pos):
17      current_pos = mj_data.qpos[1]
18      current_vel = mj_data.qvel[1]
19      data.ctrl[1] = KP * (desired_pos - current_pos) + KV * (0 - current_vel)
20
```

Рисунок 3. Функции ПИД-регулятора для каждого привода

График теоретических углов приводов представлен на рисунке 4.

График ошибки энд-эффектора по оси X представлен на рисунке 5.

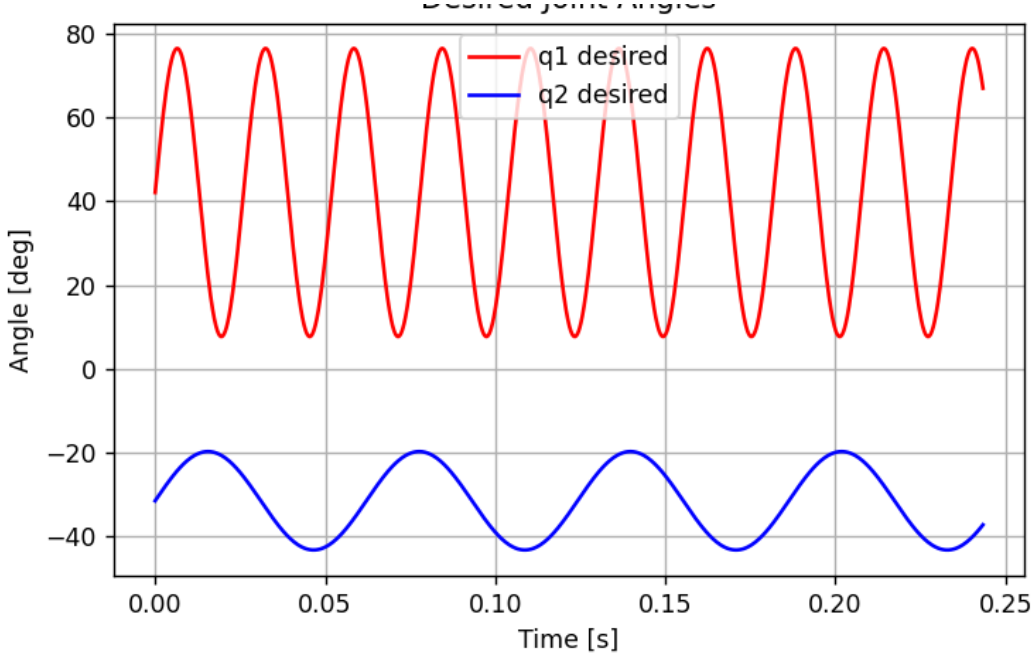

Рисунок 4. Теоретические углы приводов

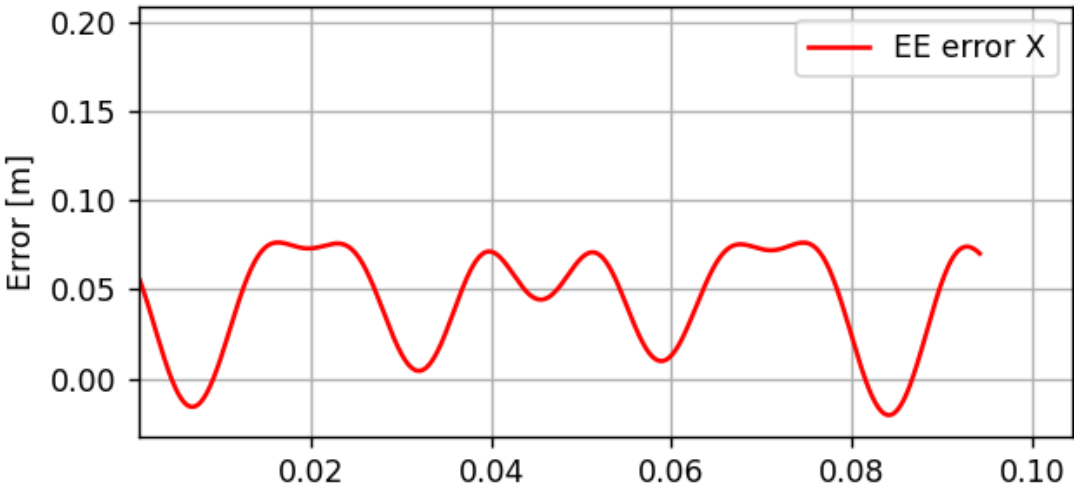

Рисунок 5. Ошибка энд-эффектора

Результаты симуляции

Механизм демонстрирует плавное движение благодаря сухожильным соединениям и синусоидальному управлению (рис. 6, 7).

Фрагменты файла XML и Python кода представлены на рисунках 8, 9 соответственно.

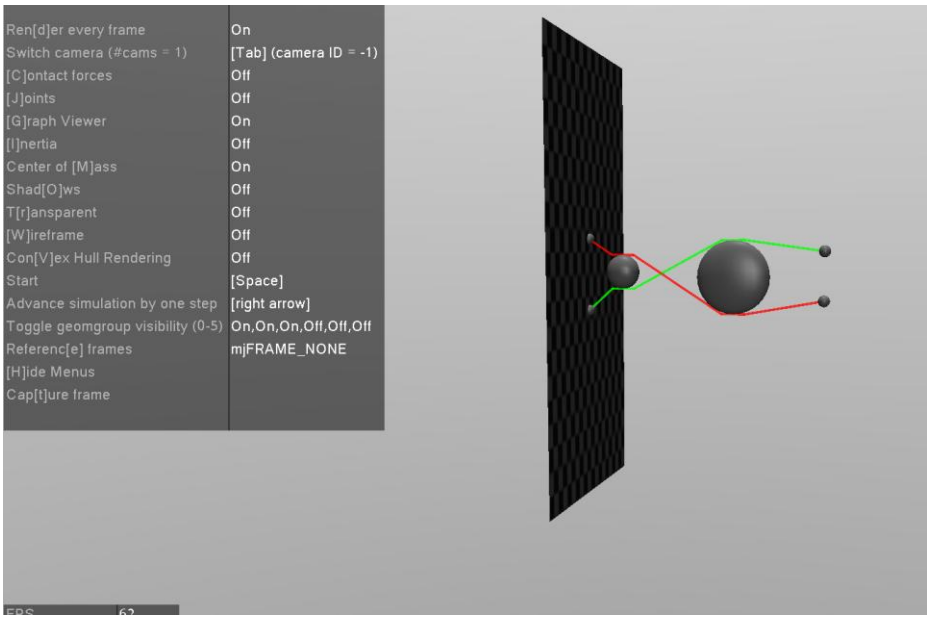

Рисунок 6. Механизм в начальном положении

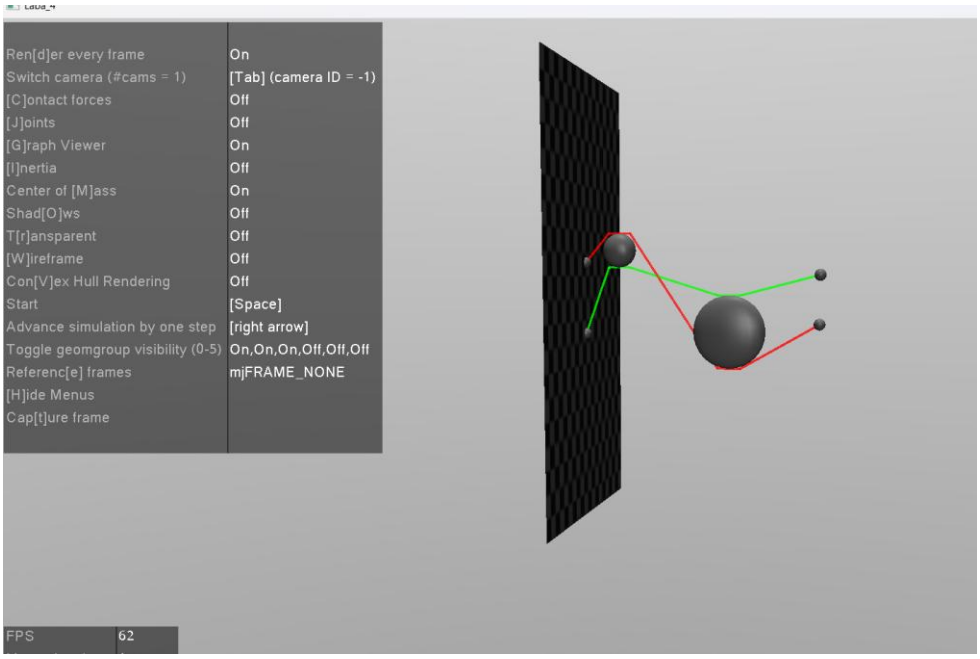

Рисунок 7. Момент работы механизма

```
    <worldbody>
        <body name="R1" pos="0.032 0 1">
            <site name="corner1downleft" pos="-0.01 0 -0.015" size="0.0001"/>
            <site name="corner1downright" pos="0.01 0 -0.015" size="0.0001"/>
        </body>


        <body name="R2" pos="0.132 0 1">
            <joint name="jointR2" type="slide" axis="0 0 1"/>
            <geom type="ellipsoid" size="0.032 0.01 0.032"/>
            <site name="R2_site" pos="0 0 0"/>


            <site name="corner2up" pos="0 0 0.032" size="0.0001"/>
            <site name="corner2upleft" pos="-0.01 0 0.032" size="0.0001"/>
            <site name="corner2upright" pos="0.01 0 0.032" size="0.0001"/>

            <site name="corner2down" pos="0 0 -0.032" size="0.0001"/>
            <site name="corner2downleft" pos="-0.01 0 -0.032" size="0.0001"/>
            <site name="corner2downright" pos="0.01 0 -0.032" size="0.0001"/>
        </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>
```

Рисунок 8. Фрагмент XML-модели

```python
for i in range(STEP_NUM):
    if viewer.is_alive:

        set_torque_R1(data, 100, 10, theta1[i])
        set_torque_R2(data, 100, 10, theta2[i])


        current_time = data.time
        position_time.append(current_time)


        position_R1 = data.site_xpos[model.site('R1_site').id]
        R1_position_x.append(position_R1[0])
        R1_position_z.append(position_R1[2])


        position_R2 = data.site_xpos[model.site('R2_site').id]
        R2_position_x.append(position_R2[0])
        R2_position_z.append(position_R2[2])

        theta1_trajectory.append(theta1[i])
        theta2_trajectory.append(theta2[i])



        mujoco.mj_step(model, data)
        viewer.render()
    else:
        break

viewer.close()
```

Рисунок 9. Фрагмент кода на языке Python

Вывод

В ходе выполнения лабораторной работы была модифицирована XML-модель механизма с сухожильными соединениями, реализованы два привода q_1 и q_2, также был использован PID-регулятор для управления механизмом. Визуализирована работа механизма посредством MuJoCo.