Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Факультет Систем Управления и Робототехники

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №4

Вариант №30

ПО ДИСЦИПЛИНЕ: «ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ
РОБОТОТЕХНИЧЕСКИХ СИСТЕМ»

Выполнил:
Мищенко И. А.,
336835, гр. R4150
Проверил:
Ракшин Е. А.,
ассистент ФСУиР

Санкт-Петербург,
2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

1 Цель работы

Цель данной работы — изучить принципы добавления актуаторов и сенсоров в модели MuJoCo, а также освоить реализацию управления механизмом с использованием PD-регулятора и синусоидальной траектории.

2 Задачи, решаемые при выполнении работы

1) Внести изменения в XML-файл модели, добавив контейнеры <actuator> и <sensor> для получения данных о положении и скорости суставов.
2) Определить желаемую траекторию движения привода, используя гармонический закон вида $q^{des}(t) = AMP \cdot \sin(FREQ \cdot t) + BIAS$ и подобрать параметры амплитуды, частоты и смещения согласно индивидуальному варианту.
3) Реализовать PD-управление приводом, определив управляющее воздействие по закону $u = K_p(q^{des} - q) + K_d(\dot{q}^{des} - \dot{q})$.
4) Провести моделирование движения механизма, оценить корректность работы привода и влияние выбранных параметров управления.
5) При необходимости скорректировать амплитуду или смещение, чтобы обеспечить движение механизма в пределах рабочей области.
6) Построить графики и визуализировать результаты, продемонстрировать поведение системы под действием PD-регулятора.
7) Сформулировать выводы о влиянии параметров траектории и регулятора на устойчивость и качество движения механизма.

ИСХОДНЫЕ ДАННЫЕ

Вариант №1: Tendon connected 2R planar mechanism

q_i	AMP, deg	$FREQ, Hz$	$BIAS, deg$
q_1	24.1	3.57	40
q_2	28.61	1.48	7

Таблица 1: Параметры согласно варианту

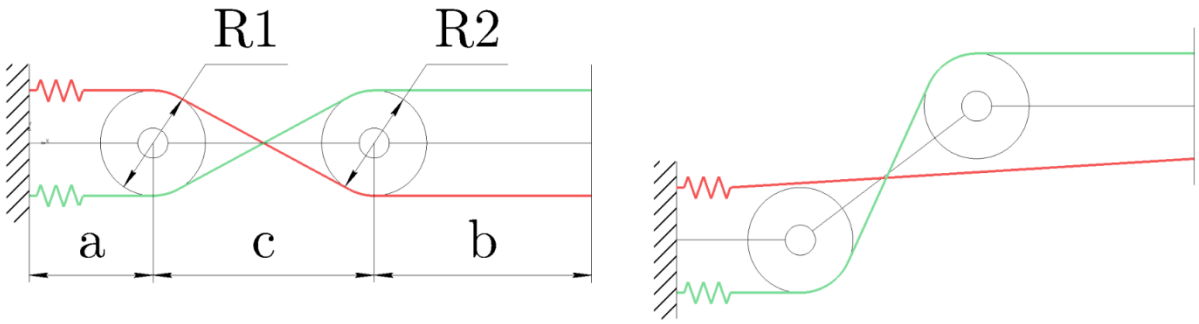

Рисунок 1: Модель двухзвенного 2R-планарного механизма

ХОД РАБОТЫ

1 Модификация XML-файла модели механизма

Для реализации управления добавим актуаторы для обоих шарниров (q_1 и q_2), а также сенсоры. Фрагмент описания актуаторов и сенсоров приведен в листинге 1:

```
<actuator>
    <motor name="motor_A" joint="A" ctrlrange="-50 50"/>
    <motor name="motor_B" joint="B" ctrlrange="-50 50"/>
</actuator>

<sensor>
    <jointpos joint="A"/>
    <jointvel joint="A"/>
    <jointpos joint="B"/>
    <jointvel joint="B"/>
</sensor>
```

Листинг 1: Фрагмент описания актуаторов и сенсоров в модели

2 Реализация алгоритма управления

Ниже приведен листинг программного кода на Python для запуска и моделирования механизма по заданному алгоритму управления в среде MuJoCo.

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

model = mujoco.MjModel.from_xml_path(

"SRS\\practice_4\\submissions\\336835_MischenkoIvan_Task4\\task4_model.xml")
data = mujoco.MjData(model)

motor1_id = model.actuator('motor_A').id # q1
motor2_id = model.actuator('motor_B').id # q2

joint_q1 = model.joint('A').id
joint_q2 = model.joint('B').id

joint_q1_pos = model.jnt_qposadr[joint_q1]
joint_q2_pos = model.jnt_qposadr[joint_q2]
```

```
joint_q1_vel = model.jnt_dofadr[joint_q1]
joint_q2_vel = model.jnt_dofadr[joint_q2]

# Расчет значений параметров
deg2rad = np.pi / 180

AMP1 = 24.1 * deg2rad
FREQ1 = 3.57
BIAS1 = 40 * deg2rad

AMP2 = 28.61 * deg2rad
FREQ2 = 1.48
BIAS2 = 7 * deg2rad

Kp1 = 8.5
Kd1 = 0.025
Kp2 = 4.5
Kd2 = 0.025

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

dt = model.opt.timestep
sim_time = 5.0
steps = int(sim_time / dt)
time = np.arange(steps) * 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[joint_q1_pos] = -q1_start
data.qpos[joint_q2_pos] = -q2_start
data.qvel[joint_q1_vel] = 0
data.qvel[joint_q2_vel] = 0

mujoco.mj_forward(model, data)

q1_log = []
q2_log = []
q1_des_log = []
q2_des_log = []

with mujoco.viewer.launch_passive(model, data) as viewer:
    viewer.cam.distance = 0.5
    viewer.cam.azimuth = 135
    viewer.cam.elevation = -20

    for i in range(steps):
        t = i * dt

        q1_pos_des = AMP1 * np.sin(2 * np.pi * FREQ1 * t) + BIAS1
        q2_pos_des = AMP2 * np.sin(2 * np.pi * FREQ2 * t) + BIAS2
        q1_vel_des = AMP1 * 2 * np.pi * FREQ1 * np.cos(2 * np.pi * FREQ1 *
t)
        q2_vel_des = AMP2 * 2 * np.pi * FREQ2 * np.cos(2 * np.pi * FREQ2 *
t)

        q1_pos = data.qpos[joint_q1_pos]
        q2_pos = data.qpos[joint_q2_pos]
        q1_vel = data.qvel[joint_q1_vel]
        q2_vel = data.qvel[joint_q2_vel]
```

```
        u1 = Kp1 * (q1_pos_des - q1_pos) + Kd1 * (q1_vel_des - q1_vel)
        u2 = Kp2 * (q2_pos_des - q2_pos) + Kd2 * (q2_vel_des - q2_vel)

        u1 = float(np.clip(u1, ctrl_min[motor1_id], ctrl_max[motor1_id]))
        u2 = float(np.clip(u2, ctrl_min[motor2_id], ctrl_max[motor2_id]))

        data.ctrl[motor1_id] = u1
        data.ctrl[motor2_id] = u2

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

        q1_log.append(q1_pos)
        q2_log.append(q2_pos)
        q1_des_log.append(q1_pos_des)
        q2_des_log.append(q2_pos_des)

plt.figure(figsize=(10, 6))
plt.plot(time, q1_log, label="q1")
plt.plot(time, q1_des_log, label="q1_des")
plt.title("Joint 1 trajectory")
plt.legend()
plt.grid(True)
plt.xlabel("Time [s]")
plt.ylabel("Angle [rad]")
plt.show()

plt.figure(figsize=(10, 6))
plt.plot(time, q2_log, label="q2")
plt.plot(time, q2_des_log, label="q2_des")
plt.title("Joint 2 trajectory")
plt.legend()
plt.grid(True)
plt.xlabel("Time [s]")
plt.ylabel("Angle [rad]")
plt.show()
```

Листинг 2: Программный код для моделирования

После загрузки MJCF-модели и инициализации структуры данных MuJoCo выполняется пошаговое моделирование динамики механизма. На каждом шаге симуляции вычисляется текущее состояние системы (положение и скорость суставов), формируется желаемое состояние, а затем по закону управления PD-регулятора рассчитывается управляющее воздействие для каждого актуатора.

Желаемая траектория для каждого сустава задаётся в виде синусоидальной функции: $q^{des}(t) = AMP \cdot \sin(2\pi \cdot FREQ \cdot t) + BIAS$, что позволяет моделировать периодические движения с заранее заданной амплитудой, частотой и смещением. Производная желаемой траектории вычисляется аналитически и используется в дифференциальной части регулятора.

Управляющее воздействие формируется по классическому закону PD-регулятора:

$$u = K_p(q^{des} - q) + K_d(\dot{q}^{des} - \dot{q}),$$

где:

K_p – коэффициент пропорциональной составляющей,

K_d – коэффициент дифференциальной составляющей,

$q, \dot{q}$ – текущее положение и скорость сустава.

Сформированная величина управления ограничивается диапазоном, заданным в XML-файле для каждого актуатора ($ctrlrange$), что имитирует реальные ограничения мощности приводов. Полученный сигнал записывается в $data.ctrl$, после чего вызывается функция $mujoco.mj\_step$, осуществляющая один шаг физического моделирования.

В процессе симуляции значения фактического положения суставов и желаемых траекторий сохраняются в массивы q_log и q_des_log, которые затем используются для построения временных графиков. Это позволяет сравнить работу регулятора и оценить точность следования заданной траектории.

3 Результаты моделирования

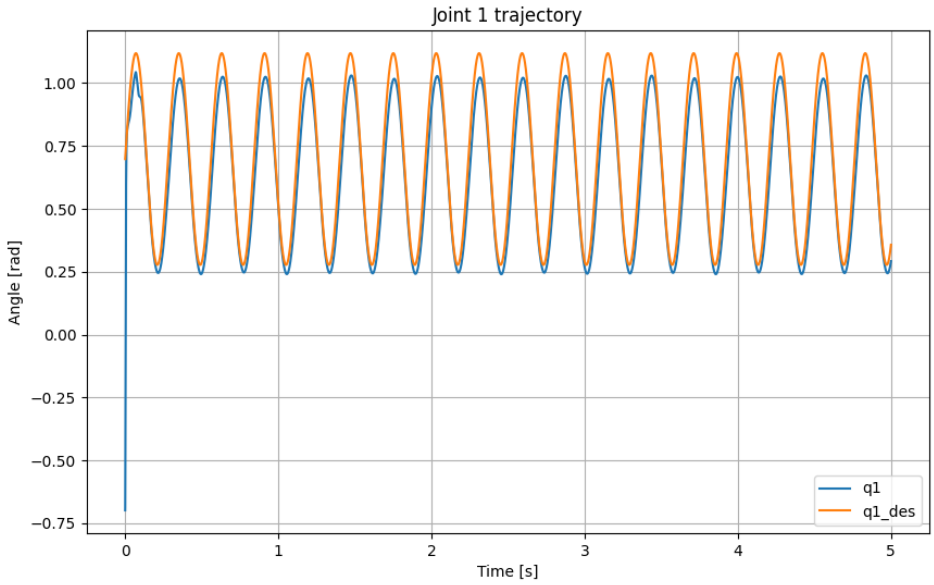

Рисунок 2: Результат моделирования для q_1

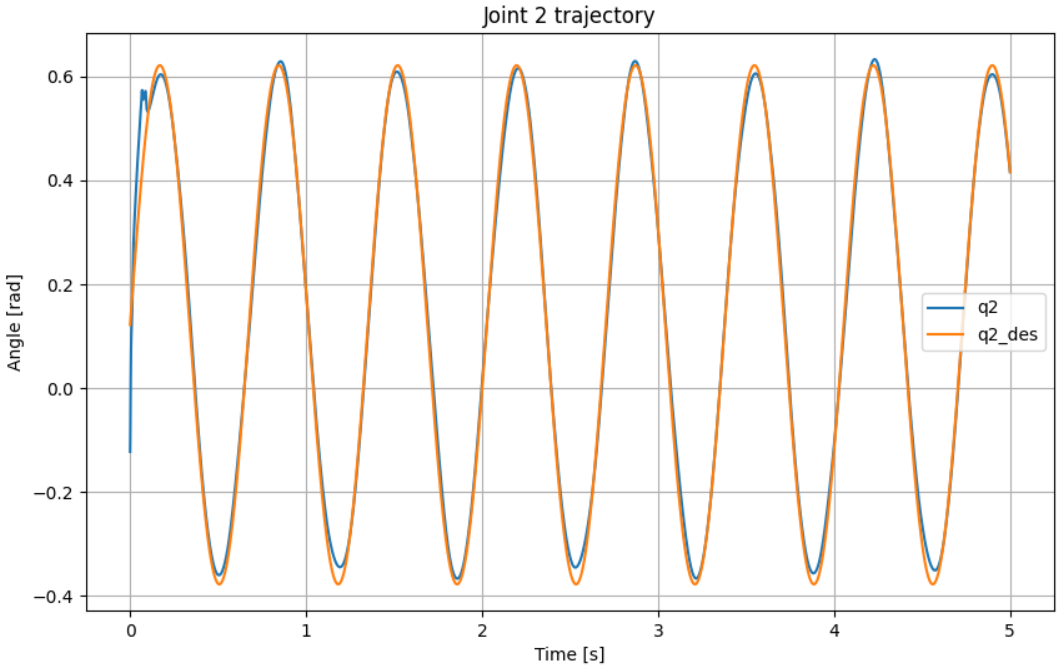

Рисунок 3: Результат моделирования для q_2

Анализ графиков, представленных на рисунках 2 и 3, показывает, что фактические траектории суставов несущественно отличаются от желаемых. В

обоих случаях прослеживается небольшое запаздывание, что свидетельствует о сложной динамике механизма и высокой чувствительности.

Так как управление осуществляется не напрямую через вращение звеньев, а через изменение натяжения пространственных сухожилий, которые проходят через систему роликов и соединительных точек. Такая конструкция приводит к выраженной нелинейности и снижает эффективность классического PD-регулятора.

Малое значение дифференциального коэффициента приводит к недостаточному демпфированию, в то время как избыточно большой пропорциональный коэффициент способен вызывать резонансные явления. Подбор параметров PD-регулятора для сухожильных механизмов является нетривиальной задачей, поскольку оптимальные значения существенно зависят от текущей конфигурации звеньев и натяжения тросов.

Несмотря на указанные сложности, результаты моделирования позволяют сделать вывод, что механизм корректно реагирует на управляющие воздействия: движение имеет выраженный периодический характер, а изменение формы траектории согласуется с изменениями направления требуемого момента. Это подтверждает корректность интеграции актуаторов и принципиальную работоспособность системы управления.

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы была создана и модифицирована MJCF-модель двухзвенного сухожильного 2R-планарного механизма. В модель были добавлены актуаторы и сенсоры, обеспечивающие возможность реализации алгоритмов управления. На основе заданных в индивидуальном варианте параметров была сформирована желаемая синусоидальная траектория движения суставов. Для слежения за этой траекторией был разработан и реализован PD-регулятор.

Результаты моделирования показали, что фактическое движение механизма практически полностью соответствует желаемому. Несущественные отклонения объясняются особенностями конструкции сухожильной передачи, наличием геометрических ограничений и выраженной нелинейностью динамики. Несмотря на это, механизм демонстрирует корректную реакцию на управляющее воздействие и способность воспроизводить периодический характер движения.

Работа позволяет сделать вывод о том, что управление сухожильными механизмами требует более сложных методов, чем классический PD-регулятор, и чувствительно к параметрам траектории и ограничению управляющего воздействия. Полученные результаты подтвердили важность корректного задания структуры модели и грамотного подбора параметров регулятора, а также продемонстрировали взаимодействие между архитектурой механизма и качеством слежения за траекторией.