Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Лабораторная работа №4

По дисциплине

«Имитационное моделирование робототехнических систем»

Выполнил: студент группы R4134c Москвин Д.А.

Проверил: ассистент Ракшин Е.А.

Санкт-Петербург 2025

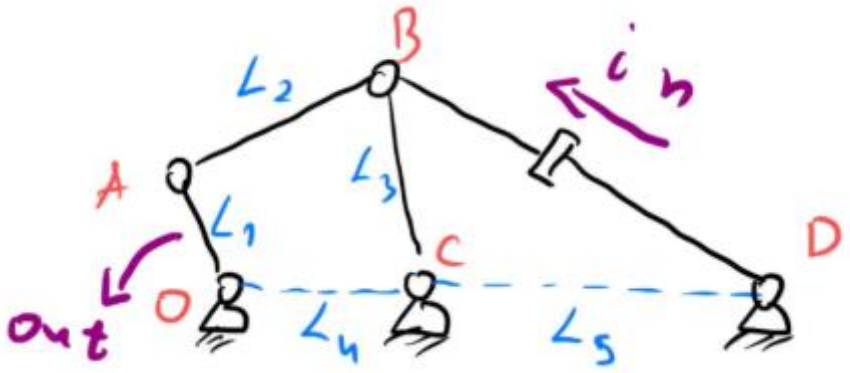
Оглавление

1. Введение 3
2. Исходные данные 3
3. Ход работы 3
4. Результаты и анализ 5
4.1. Полученная траектория 5
4.2. Анализ результатов 5
5. Выводы 5

1. Введение

Цель работы: Модификация существующей модели механизма Optimus путем добавления приводов и датчиков, реализация системы управления на основе PD-регулятора и анализ траектории движения исполнительного органа.

2. Исходные данные



L1, m = 0.077

L2, m = 0.1001

L3, m = 0.1155

L4, m = 0.077

L5, m = 0.385

Для q1:

AMP, deg = 54.96

FREQ, Hz = 1.89

BIAS, deg = 0.8

3. Ход работы

На основе модели из предыдущей лабораторной работы были внесены следующие изменения в файл optimus.xml:
Добавлены контейнеры:
<actuator> - для управления приводом ползуна
<sensor> - для отслеживания позиции точки A

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

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

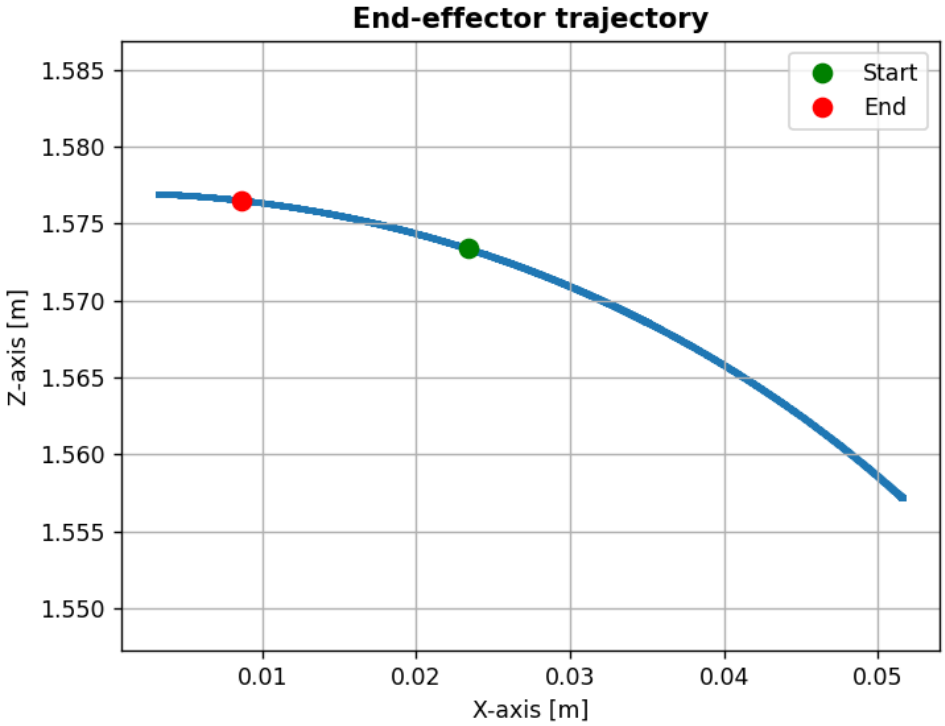
Разработан код реализующий PD-регулятор с параметрами:
KP = 50
KV = 10

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


SIMEND = 12
TIMESTEP = 0.001
STEP_NUM = int(SIMEND / TIMESTEP)
timeseries = np.linspace(0, SIMEND, STEP_NUM)

FREQ = 1.89  # [Hz]
AMP = np.deg2rad(54.96)
BIAS = np.deg2rad(0.8)

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()
EE_pos_x = np.array(EE_pos_x)
EE_pos_z = np.array(EE_pos_z)
mask = timeseries >= 5.0
x = EE_pos_x[mask]
z = EE_pos_z[mask]
t = timeseries[mask]
t_norm = (t - t.min()) / (t.max() - t.min())
plt.scatter(x, z, s=3)
plt.plot(x[0], z[0], 'go', markersize=8, label='Start')
plt.plot(x[-1], z[-1], 'ro', markersize=8, label='End')
plt.title('End-effector trajectory', fontsize=12, fontweight='bold')
plt.xlabel('X-axis [m]')
plt.ylabel('Z-axis [m]')
plt.axis('equal')
plt.grid(True)
plt.legend()
plt.show()
```

4. Результаты и анализ

4.1. Полученная траектория

На рисунке представлена траектория движения исполнительного органа механизма в плоскости X-Z



4.2. Анализ результатов

Траектория представляет собой плавную кривую, характерную для шарнирно-рычажных механизмов с ограниченной подвижностью

Движение происходит в компактной рабочей зоне, где координаты меняются в пределах нескольких сантиметров

Начальная точка и конечная точка расположены очень близко друг к другу, что говорит о периодичности и повторяемости движения

Изменение по оси Z менее выражено, что соответствует геометрии механизма и его реальной кинематике

Применяемый PD-регулятор обеспечивает устойчивое отслеживание входного сигнала

Коэффициенты KP = 50 и KV = 10 дают:

стабильное движение

отсутствие заметных колебаний

5. Выводы

Успешно модифицирована модель механизма Optimus путем добавления приводов

Реализована система управления исполнительным звеном на основе PD-регулятора, обеспечивающая стабильное отслеживание управляющего сигнала

Получена траектория движения конца звена, отражающая реальную рабочую область механизма

Система демонстрирует устойчивое поведение и точное отслеживание заданной траектории

Все параметры управляющего сигнала находятся в рабочем диапазоне механизма