Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский университет ИТМО»
(Университет ИТМО)

Отчет по практической работе № 3
по дисциплине
“Имитационное моделирование робототехнических систем”

Студент: Хомяков Д.А. (506873)
Группа: R4133c
Преподаватель: Ракшин Е.А (373529)

Санкт-Петербург
2025

1. Цель работы

В этой работе требуется воссоздать модель по заданному варианту в среде MuJoCo с использованием Python, получить данные о движении (положении) суставов и реализовать управление моделью.

2. Исходные данные

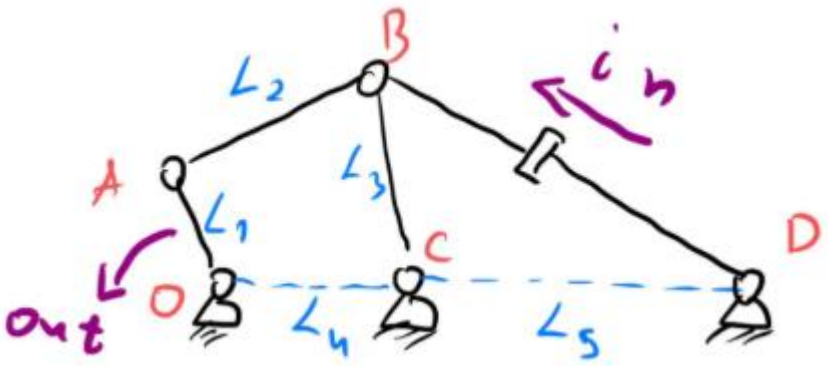

Рисунок 1. Вид схемы из технического задания

В таблице 1 представлены исходные данные

Таблица 1. Исходные данные

Вид	L1, м	L2, м	L2, м	L3, м	L4, м
OPTIMUS	0.043	0.0559	0.0645	0.043	0.215

3. Описание технического задания и поставленные задачи

На рисунке 1 изображена кинематическая схема, состоящая из 4-ёх звеньев и так называемого “слайдера”. Работа будет производится в симуляторе MuJoCo. Нужно смоделировать кинематическую схему и построить график траектории.

4. Ход решения

Представленный XML-код реализует динамическую модель четырёхзвенника типа *OPTIMUS* в среде физического моделирования MuJoCo. Конструкция включает в себя три вращательных звена, сходящихся в общей точке B, и дополнительное звено со скользящим движением (слайдер), реализованное через призматический (slide) шарнир (см приложение 1). На рисунке 2 показана кинематическая модель в MuJoCo.

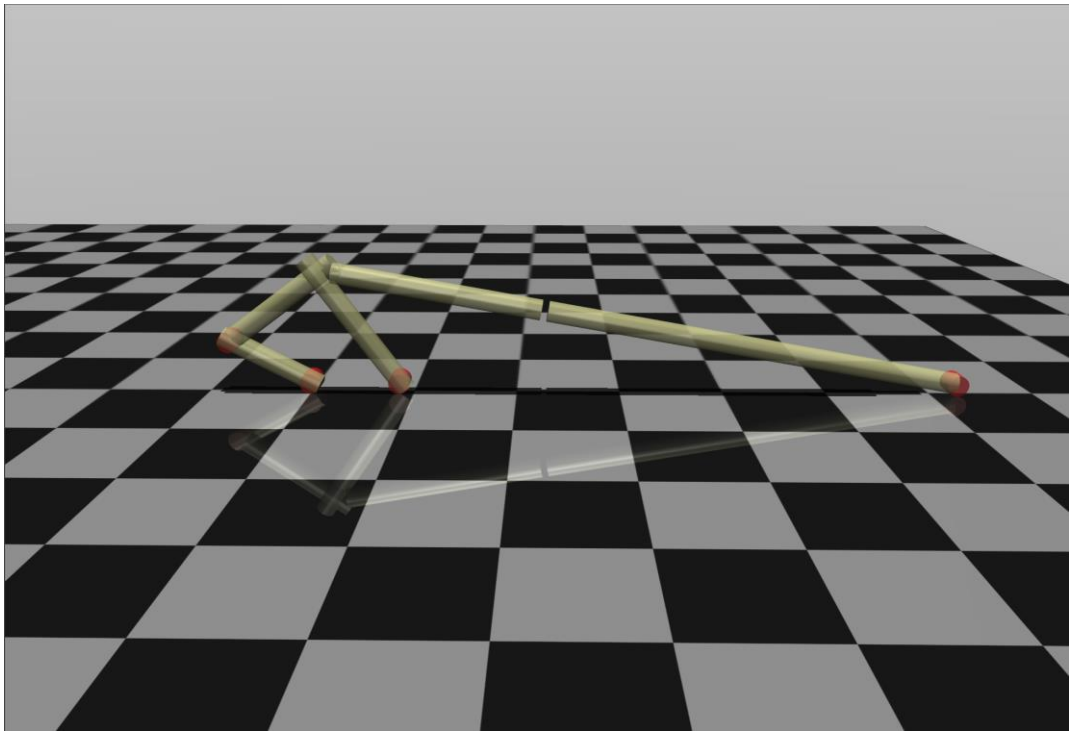

Рисунок 2. Кинематическая модель в MuJoCo.

На изображении представлена многозвенная механическая система типа OPTIMUS, состоящая из трёх вращательных звеньев и одного поступательного (слайдера). Совокупность жёстких цилиндрических звеньев, соединённых кинематическими парами — шарнирами.

Все элементы показаны в виде длинных светло-жёлтых цилиндров, а шарнирные соединения выделены красноватыми торцами.

5. Результаты

На рисунке 3 построена траектория движения точки-эффектора механизма (конечная точка звена B) с помощью Python кода (см. приложение 2).

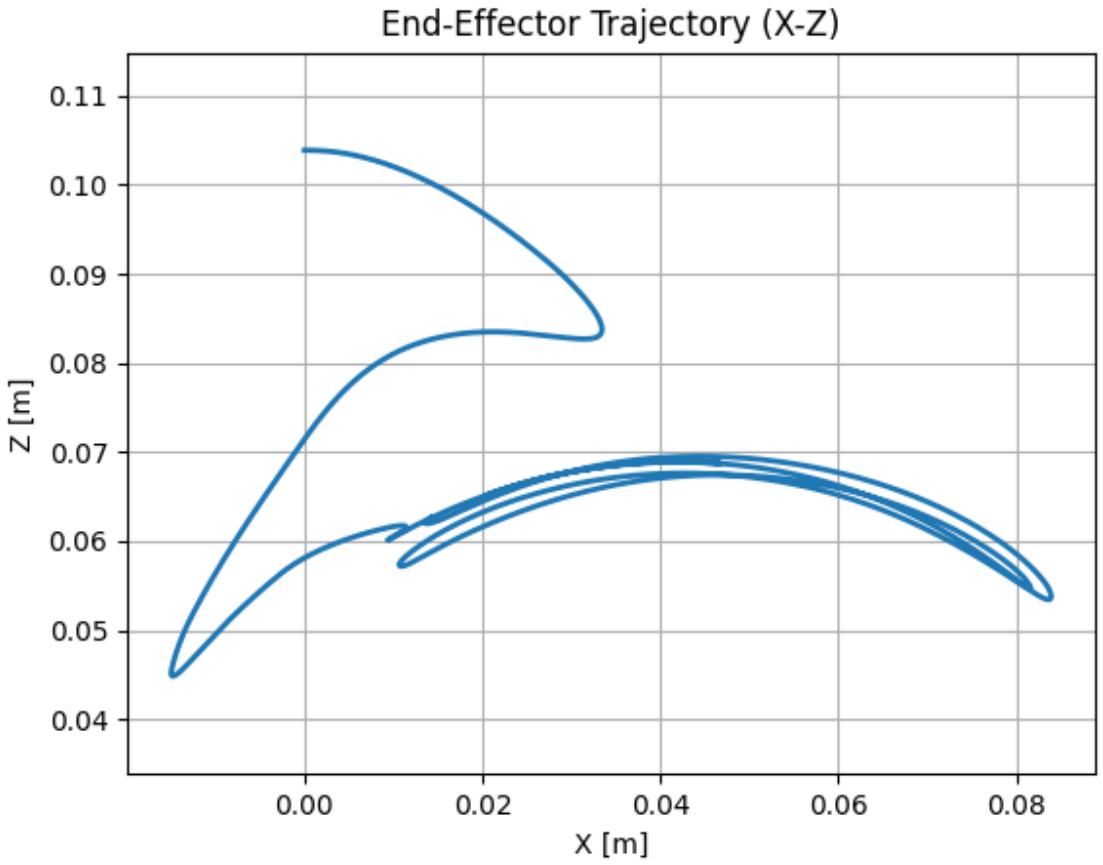

Рисунок 3. График траектории

6. Выводы

В ходе выполнения практической работы была разработана и исследована динамическая модель четырёхзвенной механической системы типа OPTIMUS в среде физического моделирования MuJoCo, построенная согласно исходным данным задания. На основе предоставленных геометрических параметров были сформированы звенья, шарнирные соединения, а также реализовано поступательное звено — слайдер, обеспечивающий изменяемую длину одной из кинематических цепей.

В процессе работы были выполнены следующие задачи:

1. Создана XML-модель механизма, включающая три вращательных шарнира и один поступательный.
   При моделировании учтены фактические длины звеньев, представленные в техническом задании, а также реализовано соединение трёх кинематических ветвей в общей точке B с помощью ограничений MuJoCo.
2. Проведена симуляция движения, в ходе которой механизм отображался в интерактивном визуализаторе MuJoCo. Были проверены корректность кинематических зависимостей и работоспособность модели под действием гравитации.
3. Разработан Python-скрипт для управления симуляцией, записи координат конечного звена (end-effector) и построения траектории его движения. Данные собирались с выбранного сайта модели, что позволило получить фактическую динамику точки B.
4. Построена траектория движения эффектора в координатной плоскости X–Z. График показывает сложное нелинейное движение, обусловленное совместной работой нескольких звеньев и наличием поступательного слайдера.

Анализ полученной траектории позволяет сделать вывод о том, что механизм генерирует характерные криволинейные движения, и его кинематика соответствует структуре исходной схемы.

7. Приложение

Приложение № 1 xml-модель:

```
<?xml version="1.0" encoding="utf-8"?>
<mujoco>

    <option timestep="1e-4"/>
    <option gravity="0 0 -9.8"/>

    <asset>
        <texture type="skybox" builtin="gradient" rgb1="1 1 1" rgb2="0.5 0.5
0.5" width="265" height="256"/>
        <texture name="grid" type="2d" builtin="checker" rgb1="0.1 0.1 0.1"
rgb2="0.6 0.6 0.6" width="300" height="300"/>
        <material name="grid" texture="grid" texrepeat="10 10"
reflectance="0.2"/>
    </asset>

    <worldbody>

        <light pos="0 0 10"/>
        <geom type="plane" size="0.5 0.5 0.1" material="grid"/>

        <camera name="side view" pos="0.1 -1.5 1.0" euler="90 0 0"
fovy="60"/>
        <camera name="upper view" pos="0 0 1.5" euler="0 0 0"/>

        <!-- O — A — B -->
        <body name="OAB" pos="0 0 0.005" euler="90 0 0">

            <joint name="O" type="hinge" axis="0 0 1" stiffness="0"
damping="0"/>

            <geom name="point O" type="cylinder" pos="0 0 0" size="0.005
0.005" rgba="0.89 0.14 0.16 0.5"/>
            <geom name="link OA" type="cylinder" pos="0 0.0215 0" size="0.005
0.026" rgba="0.8 0.8 0.56 0.6" euler="90 0 0"/>

            <body name="AB" pos="0 0.043 0">
                <joint name="A" type="hinge" axis="0 0 1" stiffness="0"
damping="0"/>

                <geom name="pointA" type="cylinder" pos="0 0 0" size="0.005
0.005" rgba="0.89 0.14 0.16 0.5"/>
                <geom name="linkAB" type="cylinder" pos="0 0.0338 0"
size="0.005 0.0310" rgba="0.8 0.8 0.56 0.6" euler="90 0 0"/>

                <site name="sB1" pos="0 0.0559 0" size="0.004"/>
            </body>
        </body>

        <!-- C — B -->
        <body name="CB" pos="0.043 0 0.005" euler="90 0 0">
            <joint name="C" type="hinge" axis="0 0 1"/>

            <geom name="pointC" type="cylinder" pos="0 0 0" size="0.005
0.005" rgba="0.89 0.14 0.16 0.5"/>
            <geom name="linkCB" type="cylinder" pos="0 0.03445 0" size="0.005
0.039" rgba="0.8 0.8 0.56 0.6" euler="90 0 0"/>

            <site name="sB2" pos="0 0.0645 0" size="0.004"/>
```

```
        </body>

        <!--ДОБАВЛЕННЫЙ СЛАЙДЕР-->
        <body name="DSB" pos="0.312 0 0.005" euler="90 0 0">

            <joint name="D" type="hinge" axis="0 0 1"/>

            <geom name="pointD" type="cylinder" pos="0 0 0" size="0.005
0.005" rgba="0.89 0.14 0.16 0.5"/>
            <geom name="linkDS" type="cylinder" pos="0 0.1 0" size="0.005
0.1" rgba="0.8 0.8 0.56 0.6" euler="90 0 0"/>

            <body name="SB" pos="0 0.17 0">
                <joint name="S" type="slide" axis="0 1 0" range="-0.03
0.03"/>

                <geom name="linkSB" type="cylinder" pos="0 0.05 0"
size="0.005 0.05" rgba="0.8 0.8 0.56 0.6" euler="90 0 0"/>
                <site name="sB3" pos="0 0.1 0" size="0.004"/>
            </body>

        </body>

    </worldbody>

    <!-- Все звенья соединяются в одну точку B -->
    <equality>
        <connect site1="sB1" site2="sB2"/>
        <connect site1="sB1" site2="sB3"/>
    </equality>

    <!-- Актуатор слайдера -->
    <actuator>
        <position name="S" joint="S"/>
    </actuator>

</mujoco>
```

Приложение № 2 Python код:

```
import time
import mujoco
import mujoco.viewer
import matplotlib.pyplot as plt

paused = False

# ---------------------------------------------
#  SPACE – pause/unpause
# ---------------------------------------------
def key_callback(keycode):
    global paused
    if keycode == 32:     # SPACE
        paused = not paused
        print("Paused =", paused)


# ---------------------------------------------
#  MAIN
# ---------------------------------------------
def main():
```

```
    XML_FILE = "optimus.xml"    # ← поменяй на имя, которое ты используешь

    # загрузка модели
    m = mujoco.MjModel.from_xml_path(XML_FILE)
    d = mujoco.MjData(m)

    # сайт конечного звена
    EE_SITE = "sB1"
    ee_x, ee_y, ee_z = [], [], []

    print("\nЗапуск simulation... нажми SPACE для паузы.\n")

    # запуск viewer
    with mujoco.viewer.launch_passive(m, d, key_callback=key_callback) as
viewer:

        while viewer.is_running():

            step_start = time.time()

            if not paused:
                mujoco.mj_step(m, d)
                viewer.sync()

                # координаты end-effector
                sid = m.site(EE_SITE).id
                pos = d.site_xpos[sid]

                ee_x.append(pos[0])
                ee_y.append(pos[1])
                ee_z.append(pos[2])

            # синхронизация под реальное время
            dt = m.opt.timestep - (time.time() - step_start)
            if dt > 0:
                time.sleep(dt)

    # ================================================
    #     ПОСТРОЕНИЕ ГРАФИКА ПОСЛЕ ЗАКРЫТИЯ VIEWER
    # ================================================

    print("\nViewer закрыт — строю траекторию...\n")

    fig = plt.figure()
    ax = fig.add_subplot(111)

    ax.plot(ee_x, ee_z, linewidth=2)
    ax.set_title("End-Effector Trajectory (X-Z)")
    ax.set_xlabel("X [m]")
    ax.set_ylabel("Z [m]")
    ax.grid(True)
    ax.axis("equal")

    # включаем панораму / зум мышкой
    plt.ion()
    plt.show()
    input("\nНажми ENTER для выхода...\n")


if __name__ == "__main__":
    main()
```