Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет по практической работе №3

По дисциплине: Имитационное моделирование робототехнических систем

На тему: «Моделирование в среде Mujoko»

Студент:

Рязанцев Д.Л.

Группа:

R4134c

Преподаватель:

Ракшин Е. А.

Санкт-Петербург, 2025

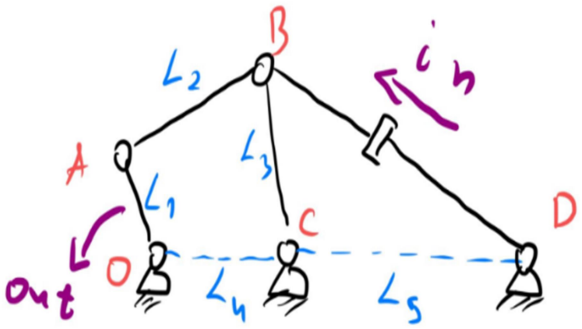
СОДЕРЖАНИЕ

Цель работы: промоделировать систему в Mujoko

Постановка задачи

- Написать код на python;
- Запустить симуляцию

Исходные данные берутся в соответствии с номером ИСУ 507578. Вариант 2.




```
# Параметры длин
L1 = 0.079
L2 = 0.1027
L3 = 0.1185
L4 = 0.079
L5 = 0.395
```

Рисунок 1. Схема системы

МОДЕЛИРОВАНИЕ

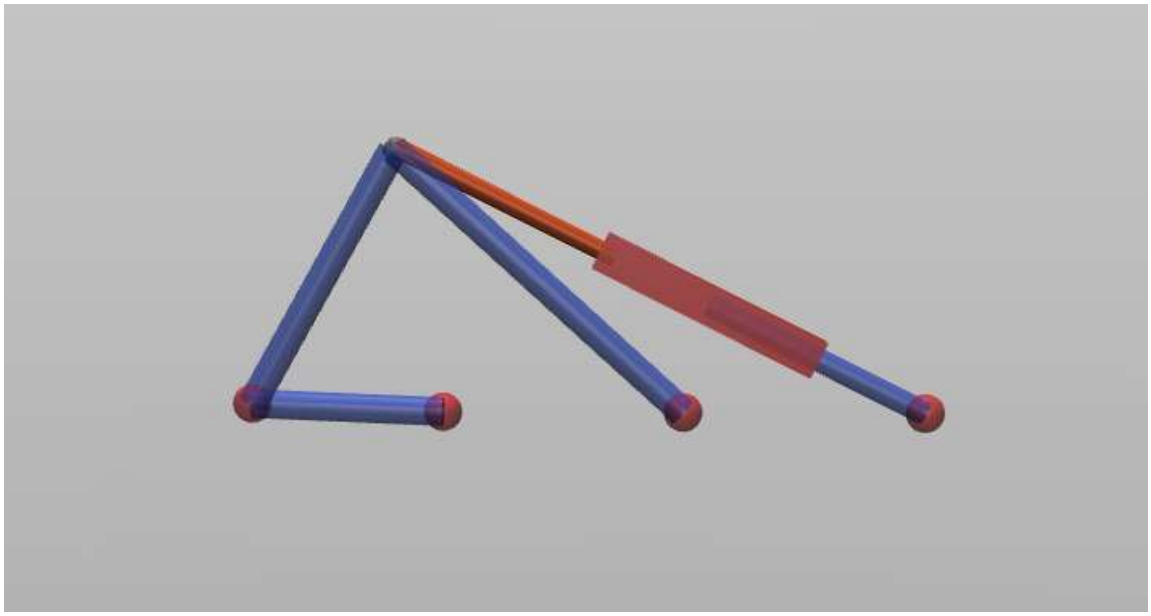


Рисунок 2. Модель коленного механизма с замкнутой цепью Оптимуса

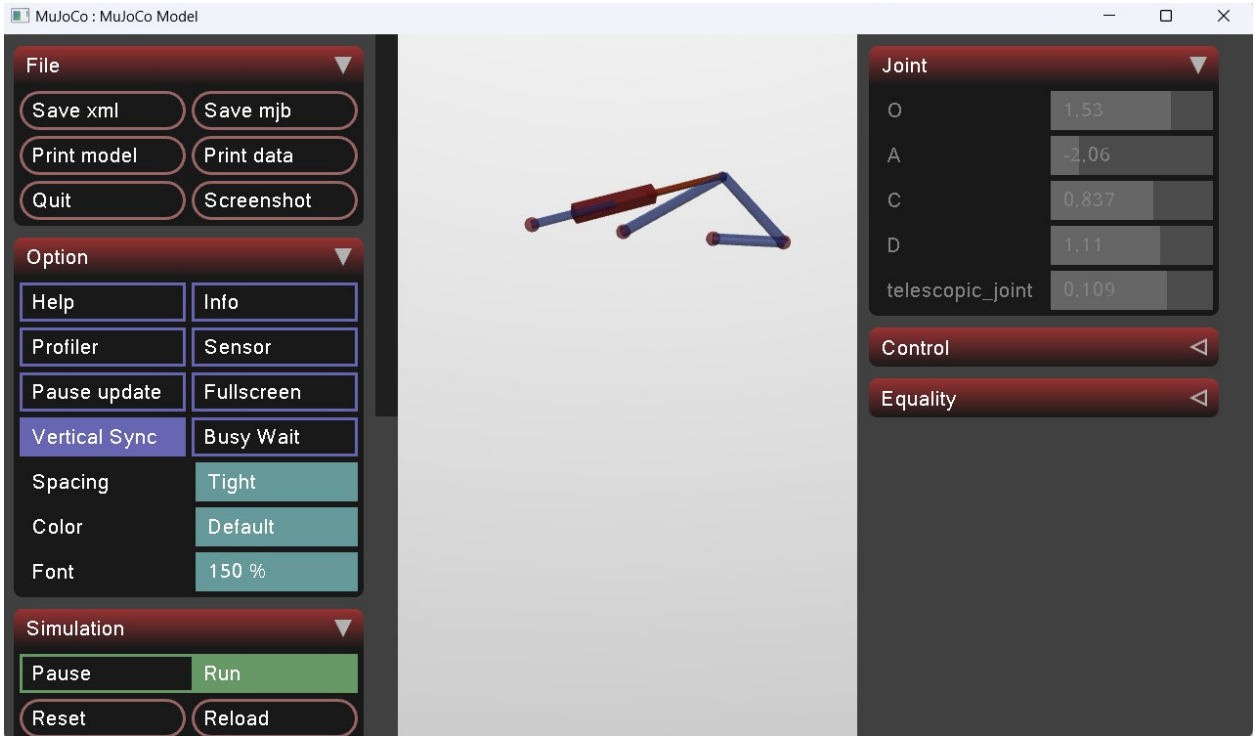


Рисунок 3. Схема системы

КОД НА python

```
import mujoco
import mujoco.viewer
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
import numpy as np
import os
from lxml import etree
import time
f1 = "4bar.xml"
f2 = "4bar_telescopic_connector.xml"
def swap_par(tree, element_type, element_name, attribute_name, new_value):
    element = tree.find(f'.//{element_type}[@name="{element_name}"]')
    if element is not None:
        element.set(attribute_name, new_value)
    else:
        print(f"Предупреждение: Элемент {element_type}[@name='{element_name}'] не
найден")
# Параметры длин
L1 = 0.079
L2 = 0.1027
L3 = 0.1185
L4 = 0.079
L5 = 0.395
L_BD_fixed = 0.20  # Фиксированная часть BD
L_BD_telescopic = 0.12  # Телескопическая часть BD
connector_length = 0.10  # Начальная длина соединителя
```

```
h = 1.5
tree = etree.parse(f1)
# БАЛАНСИРОВАННЫЕ параметры
joints = tree.findall('.//joint')
for joint in joints:
    joint.set('damping', '0.1')  # Умеренное демпфирование
    if joint.get('stiffness') is not None:
        joint.set('stiffness', '10')  # Небольшая жесткость
# Умеренные коннекторы
equality = tree.find('.//equality')
if equality is not None:
    connect_elements = equality.findall('.//connect')
    for connect in connect_elements:
        connect.set('solimp', '0.8 0.9 0.01')
        connect.set('solref', '0.01 0.5')
swap_par(tree, 'geom', 'link OA', 'pos', f"0 -{L1/2} 0")
swap_par(tree, 'geom', 'link OA', 'size', f"0.015 {L1/2}")
swap_par(tree, 'body', 'AB', 'pos', f"0 -{L1} 0")
swap_par(tree, 'geom', 'link AB', 'pos', f"0 -{L2/2} 0")
swap_par(tree, 'geom', 'link AB', 'size', f"0.015 {L2/2}")
swap_par(tree, 'site', 'sB1', 'pos', f"0 -{L2} 0")
swap_par(tree, 'body', 'CB2', 'pos', f"{L4} 0 {h}")
swap_par(tree, 'geom', 'link CB', 'pos', f"0 -{L3/2} 0")
swap_par(tree, 'geom', 'link CB', 'size', f"0.015 {L3/2}")
swap_par(tree, 'site', 'sB2', 'pos', f"0 -{L3} 0")
swap_par(tree, 'body', 'DB3', 'pos', f"{L4 + L5} 0 {h}")
swap_par(tree, 'geom', 'link DB_fixed', 'pos', f"0 -{L_BD_fixed/2} 0")
swap_par(tree, 'geom', 'link DB_fixed', 'size', f"0.012 {L_BD_fixed/2}")
# Обновляем соединительный элемент
swap_par(tree, 'geom', 'connector', 'pos', f"0 -{L_BD_fixed + connector_length/2}
0")
swap_par(tree, 'geom', 'connector', 'size', f"0.02 {connector_length/2} 0.02")
# Обновляем позиции для телескопической части
telescopic_body = tree.find('.//body[@name="BD_telescopic"]')
if telescopic_body is not None:
    telescopic_body.set('pos', f"0 -{L_BD_fixed + connector_length} 0")
telescopic_geom = tree.find('.//geom[@name="link DB_telescopic"]')
if telescopic_geom is not None:
    telescopic_geom.set('pos', f"0 -{L_BD_telescopic/2} 0")
    telescopic_geom.set('size', f"0.01 {L_BD_telescopic/2}")
telescopic_site = tree.find('.//site[@name="sB3"]')
if telescopic_site is not None:
    telescopic_site.set('pos', f"0 -{L_BD_telescopic} 0")
# Обновляем joint range для телескопического звена
telescopic_joint = tree.find('.//joint[@name="telescopic_joint"]')
if telescopic_joint is not None:
    telescopic_joint.set('range', f"{-L_BD_telescopic/2} {L_BD_telescopic/2}")
    telescopic_joint.set('damping', '0.02')
# Обновляем актуатор - сбалансированная мощность
telescopic_actuator = tree.find('.//position[@name="telescopic_control"]')
if telescopic_actuator is not None:
```

```
    telescopic_actuator.set('ctrlrange', f"{-1.0} {1.0}")
    telescopic_actuator.set('kp', '800')
    telescopic_actuator.set('kv', '15')
tree.write(f2, pretty_print=True, xml_declaration=True, encoding='UTF-8')
model = mujoco.MjModel.from_xml_path(f2)
data = mujoco.MjData(model)
class BalancedPIDController:
    def __init__(self, kp, ki, kd, output_limits=(-np.inf, np.inf)):
        self.kp = kp
        self.ki = ki
        self.kd = kd
        self.output_limits = output_limits
        self.reset()
    def reset(self):
        self.integral = 0.0
        self.previous_error = 0.0
    def compute(self, error, dt):
        # Ограничиваем интегральную составляющую
        self.integral += error * dt
        self.integral = np.clip(self.integral, -2, 2)
        if dt > 0:
            derivative = (error - self.previous_error) / dt
        else:
            derivative = 0.0
        output = self.kp * error + self.ki * self.integral + self.kd * derivative
        output = np.clip(output, self.output_limits[0], self.output_limits[1])
        self.previous_error = error
        return output
# Получаем индексы
telescopic_actuator_idx = mujoco.mj_name2id(model, mujoco.mjtObj.mjOBJ_ACTUATOR,
"telescopic_control")
telescopic_joint_idx = mujoco.mj_name2id(model, mujoco.mjtObj.mjOBJ_JOINT,
"telescopic_joint")
joint_A_idx = mujoco.mj_name2id(model, mujoco.mjtObj.mjOBJ_JOINT, "A")
connector_geom_idx = mujoco.mj_name2id(model, mujoco.mjtObj.mjOBJ_GEOM,
"connector")

# Функция для обновления размера и позиции соединительного элемента
def update_connector(telescopic_position):
    if connector_geom_idx != -1:
        # Убедимся, что соединитель правильно отражает движение телескопа
        connector_pos_y = - (L_BD_fixed + connector_length/2 +
telescopic_position/2)
        connector_size_y = connector_length/2 + telescopic_position/2
        model.geom_size[connector_geom_idx][1] = max(0.01, connector_size_y)
        model.geom_pos[connector_geom_idx][1] = connector_pos_y
pid_controller = BalancedPIDController(
    kp=500.0,
    ki=15.0,
    kd=10.0,
    output_limits=(-50.0, 50.0)
```

```
with mujoco.viewer.launch_passive(model, data) as viewer:
    viewer.cam.distance = 2.25
    viewer.cam.azimuth = 90
    viewer.cam.elevation = -10
    viewer.cam.lookat[:] = [0.15, -0.2, 1.25]
    print("=== ФАЗА 1: СБАЛАНСИРОВАННАЯ СТАБИЛИЗАЦИЯ ===")
    # Устанавливаем начальные положения для ЕСТЕСТВЕННОЙ конфигурации
    if joint_A_idx != -1:
        # Среднее положение между верхним и нижним
        data.qpos[model.jnt_qposadr[joint_A_idx]] = np.deg2rad(-90)
    if telescopic_joint_idx != -1:
        data.qpos[model.jnt_qposadr[telescopic_joint_idx]] = 0.0
    # Умеренная стабилизация
    stabilization_steps = 800
    for i in range(stabilization_steps):
        if telescopic_joint_idx != -1:
            current_pos = data.qpos[model.jnt_qposadr[telescopic_joint_idx]]
            update_connector(current_pos)
        # Плавное уменьшение воздействия
        if i < 200 and telescopic_actuator_idx != -1:
            data.ctrl[telescopic_actuator_idx] = 0.0
        mujoco.mj_step(model, data)
        viewer.sync()
        if i % 160 == 0 and telescopic_joint_idx != -1:
            current_pos = data.qpos[model.jnt_qposadr[telescopic_joint_idx]]
            angle_A = data.qpos[model.jnt_qposadr[joint_A_idx]] if joint_A_idx !=
-1 else 0
            #print(f"Шаг {i}: телескоп = {current_pos:.3f} м, угол A =
{np.rad2deg(angle_A):.1f}°")
        if i < 400:
            time.sleep(0.002)
    print("Стабилизация завершена")
    if telescopic_actuator_idx == -1 or telescopic_joint_idx == -1:
        print("ОШИБКА: Не могу продолжить - необходимые элементы не найдены")
    else:
        print("=== ФАЗА 2: ПЛАВНОЕ УПРАВЛЕНИЕ ТЕЛЕСКОПОМ ===")
        angle_A_history = []
        telescopic_pos_history = []
        connector_length_history = []
        time_history = []
        control_signal_history = []
        positions = [-0.15, -0.1, -0.25, -0.05, -0.3]
        start_time = time.time()
        pid_controller.reset()
        for pos_idx, target_pos in enumerate(positions):
            current_angle = data.qpos[model.jnt_qposadr[joint_A_idx]] if
joint_A_idx != -1 else 0
            print(f"\n--- Позиция {pos_idx+1}/{len(positions)}: телескоп ->
{target_pos:.2f} м ---")
            # НАЧАЛЬНЫЙ ТОЛЧОК для преодоления трения
            initial_push_steps = 20
```

```
            for push_step in range(initial_push_steps):
                error = target_pos -
data.qpos[model.jnt_qposadr[telescopic_joint_idx]]
                control_signal = np.sign(error) * 1.0
                data.ctrl[telescopic_actuator_idx] = control_signal
                mujoco.mj_step(model, data)
                viewer.sync()
                time.sleep(0.001)
            move_steps = 2000
            for step in range(move_steps):
                previous_pos = current_pos
                dt = 0.01
                current_pos = data.qpos[model.jnt_qposadr[telescopic_joint_idx]]
                angle_A = data.qpos[model.jnt_qposadr[joint_A_idx]] if
joint_A_idx != -1 else 0
                # Обновляем соединительный элемент
                update_connector(current_pos)
                error = target_pos - current_pos
                control_signal = pid_controller.compute(error, dt)
                data.ctrl[telescopic_actuator_idx] = control_signal
                mujoco.mj_step(model, data)
                viewer.sync()
                angle_A_history.append(angle_A)
                telescopic_pos_history.append(current_pos)
                connector_length_history.append(model.geom_size[connector_geom_id
x][1] * 2)
                time_history.append(time.time() - start_time)
                control_signal_history.append(control_signal)
                # Вывод информации (реже для читаемости)
                if step % 70 == 0:
                    connector_len = model.geom_size[connector_geom_idx][1] * 2
                    print(f"  Шаг {step}: телескоп={current_pos:.3f}м,
цель={target_pos:.3f}м")
                    print(f"         ошибка={error:.3f},
управление={control_signal:.3f}")
                    print(f"         угол A={np.rad2deg(angle_A):.1f}°")
                time.sleep(0.008)
            # Результат позиции
            final_angle = data.qpos[model.jnt_qposadr[joint_A_idx]] if
joint_A_idx != -1 else 0
            final_pos = data.qpos[model.jnt_qposadr[telescopic_joint_idx]]
            final_connector_len = model.geom_size[connector_geom_idx][1] * 2
            final_error = target_pos - final_pos
            print(f"  Достигнуто: телескоп={final_pos:.3f}м (цель:
{target_pos:.3f}м)")
            print(f"     ошибка: {final_error:.3f}м, угол
A={np.rad2deg(final_angle):.1f}°")
        print("\n=== АНАЛИЗ РЕЗУЛЬТАТОВ ===")
        # Построение графиков
        if angle_A_history and telescopic_pos_history:
            plt.figure(figsize=(15, 12))
```

```
            plt.subplot(4, 1, 1)
            plt.plot(time_history, telescopic_pos_history, 'b-', linewidth=2,
label='Фактическая позиция')
            target_for_plot = []
            for i, pos in enumerate(positions):
                target_for_plot.extend([pos] * move_steps)
            target_for_plot = target_for_plot[:len(time_history)]
            plt.plot(time_history, target_for_plot, 'r--', alpha=0.7,
linewidth=1, label='Целевая позиция')
            plt.ylabel('Позиция [м]')
            plt.title('Сбалансированное управление телескопическим звеном')
            plt.legend()
            plt.grid(True)
            plt.subplot(4, 1, 2)
            plt.plot(time_history, control_signal_history, 'orange', linewidth=1,
alpha=0.7, label='Сигнал управления')
            plt.ylabel('Управление')
            plt.title('Сигнал управления ПИД-регулятора')
            plt.legend()
            plt.grid(True)
            plt.subplot(4, 1, 3)
            plt.plot(time_history, connector_length_history, 'r-', linewidth=2,
label='Длина соединителя')
            plt.ylabel('Длина [м]')
            plt.title('Длина соединительного элемента')
            plt.legend()
            plt.grid(True)
            plt.subplot(4, 1, 4)
            plt.plot(time_history, np.rad2deg(angle_A_history), 'g-',
linewidth=2, label='Угол A')
            plt.plot(telescopic_pos_history, np.rad2deg(angle_A_history),
'purple', linewidth=1, alpha=0.5, label='Зависимость')
            plt.ylabel('Угол A [°]')
            plt.xlabel('Время [с] / Позиция телескопа [м]')
            plt.title('Угол в точке A и его зависимость от позиции телескопа')
            plt.legend()
            plt.grid(True)
            plt.tight_layout()
            plt.savefig('balanced_telescopic_control.png', dpi=300,
bbox_inches='tight')
            #print("✓ Графики сохранены в 'balanced_telescopic_control.png'")
            # Статистика
            """print(f"\nСТАТИСТИКА:")
            print(f"Диапазон угла A: {np.min(np.rad2deg(angle_A_history)):.1f}°
до {np.max(np.rad2deg(angle_A_history)):.1f}°")
            print(f"Диапазон позиции телескопа:
{np.min(telescopic_pos_history):.3f} до {np.max(telescopic_pos_history):.3f} м")
            print(f"Средняя ошибка позиции:
{np.mean(np.abs(np.array(telescopic_pos_history) -
np.array(target_for_plot))):.3f} м")"""
print("=== СИМУЛЯЦИЯ ЗАВЕРШЕНА ===")
```

МОДЕЛЬ В XML

```
<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"/>
<material name="connector_red" rgba="0.9 0.2 0.2 0.8"/>
</asset>
<worldbody>
<light pos="0 0 10"/>
<geom type="plane" size="0.5 0.5 0.1" material="grid"/>
<body name="OAB" pos="0 0 1.5" euler="90 0 180">
<joint name="O" type="hinge" axis="0 0 1" stiffness="0" damping="0.05"/>
<geom name="point O" type="sphere" pos="0 0 0" size="0.02" rgba="0.89 0.14 0.16
0.5"/>
<geom name="link OA" type="cylinder" pos="0 -0.1 0" size="0.015 0.1" rgba="0.21
0.32 0.82 0.5" euler="90 0 0"/>
<body name="AB" pos="0 -0.2 0" euler="0 0 0">
<joint name="A" type="hinge" axis="0 0 1" stiffness="0" damping="0.05"/>
<geom name="point A" type="sphere" pos="0 0 0" size="0.02" rgba="0.89 0.14 0.16
0.5"/>
<geom name="link AB" type="cylinder" pos="0 -0.15 0" size="0.015 0.15" rgba="0.21
0.32 0.82 0.5" euler="90 0 0"/>
<site name="sB1" size="0.01" pos="0 -0.3 0"/>
</body>
</body>
<body name="CB2" pos="0.25 0 1.5" euler="90 0 180">
<joint name="C" type="hinge" axis="0 0 1" stiffness="0" damping="0.05"/>
<geom name="point C" type="sphere" pos="0 0 0" size="0.02" rgba="0.89 0.14 0.16
0.5"/>
<geom name="link CB" type="cylinder" pos="0 -0.2 0" size="0.015 0.2" rgba="0.21
0.32 0.82 0.5" euler="90 0 0"/>
<site name="sB2" size="0.01" pos="0 -0.4 0"/>
</body>
<body name="DB3" pos="0.5 0 1.5" euler="90 0 180">
<joint name="D" type="hinge" axis="0 0 1" stiffness="0" damping="0.05"/>
<geom name="point D" type="sphere" pos="0 0 0" size="0.02" rgba="0.89 0.14 0.16
0.5"/>
<geom name="link DB_fixed" type="cylinder" pos="0 -0.125 0" size="0.012 0.125"
rgba="0.21 0.32 0.82 0.5" euler="90 0 0"/>
<geom name="connector" type="box" size="0.02 0.125 0.02" rgba="0.9 0.2 0.2 0.8"
pos="0 -0.25 0"/>
<body name="BD_telescopic" pos="0 -0.25 0">
<joint name="telescopic_joint" type="slide" axis="0 -1 0" limited="true" range="-
0.25 0.25" stiffness="0" damping="0.1"/>
```

```
<geom name="link DB_telescopic" type="cylinder" pos="0 -0.125 0" size="0.01
0.125" rgba="0.9 0.3 0.1 0.8" euler="90 0 0"/>
<site name="sB3" size="0.01" pos="0 -0.25 0"/>
<site name="connector_start" size="0.005" pos="0 0.125 0" rgba="1 0 0 1"/>
</body>
</body>
</worldbody>
<equality>
<connect site1="sB1" site2="sB2" solimp="0.8 0.9 0.01" solref="0.01 0.3"/>
<connect site1="sB2" site2="sB3" solimp="0.8 0.9 0.01" solref="0.01 0.3"/>
</equality>
<actuator>
<position name="telescopic_control" joint="telescopic_joint" ctrllimited="true"
ctrlrange="-0.25 0.25" kp="200" kv="30"/>
</actuator>
<sensor>
<framepos objtype="site" objname="sB1"/>
<jointpos joint="O"/>
<jointpos joint="A"/>
<jointpos joint="telescopic_joint"/>
<framepos objtype="site" objname="sB3"/>
<jointpos joint="A" name="joint_A_angle"/>
<framepos objtype="site" objname="connector_start"/>
</sensor>
</mujoco>
```

Заключение

Данная система представляет собой комплекс, реализующий управление четырехзвенным механизмом с телескопическим звеном через физический движок MuJoCo. По своей сути, это виртуальный прототип реального механического устройства, где математическая модель точно воспроизводит физические свойства и динамические процессы.

Основная идея системы заключается в создании адаптивного механизма, способного изменять свою геометрию в реальном времени за счет телескопического элемента. Это не просто статическая конструкция, а динамическая система, где изменение длины одного звена вызывает перераспределение усилий и перемещений во всей кинематической цепи. Особенность подхода заключается в балансировке параметров управления — демпфирования, жесткости и управляющих воздействий, что обеспечивает плавное и предсказуемое поведение системы.