Министерство науки и высшего образования Российской Федерации

Государственное Автономное Образовательное Учреждение Высшего Образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Имитационное моделирование робототехнических систем

Отчет по третьей лабораторной работе

Автор:	Терещенко Евгений Константинович
Номер ИСУ:	507632
Группа:	R4150
Преподаватель:	Ракшин Егор Александрович
Дата сдачи отчета:	18 ноября 2025 г.

Оглавление

1 Цель Работы

Целью данной работы является моделлирование динамики пассивного коленного механизма типа Optimus knee в среде MuJoCo и осуществить подобор его геометрических параметров таким образом, чтобы траектория конечной точки была максимально близка к горизонтальной линии на заданной высоте h_z. Исходная система представленна на изображении 1.1

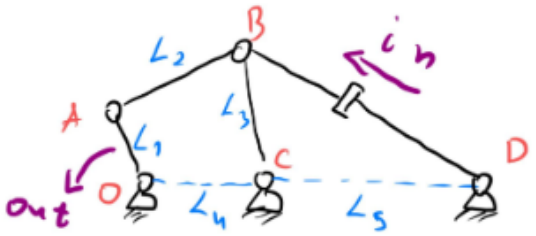

Рис. 1.1: Заданная система

Результатом работы должна стать оптимизация геометрии за счет работы оптимизатора и анализ ошибки между действительным и "желаемым"значением.

2 Ход работы

Прежде всего сама система должна быть инициализирована пользователем. В среде MuJoCo это достигается путем создания файла на разметочном языке XML.

Поэтому сначала будет разобрана архитектура XML файла, а затем обсуждена работа кода и в частности оптимизатора, а так же проанализированы ошибки.

2.1 Структура XML-модели

Модель механизма задаётся в файле `mech.xml` в формате MuJoCo XML. Ниже кратко рассмотрены ключевые элементы.

Во-первых, задаются глобальные параметры моделирования: шаг интегрирования и гравитация (ускорение свободного падения направлено вдоль оси z):

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

Далее в секции `asset` описывается оформление сцены (текстура пола и т.п.), а в `worldbody` создаются источник света, опорная плоскость и камеры обзора:

```
<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"/>
    <camera name="upper view" pos="0 0 1.5"      euler="0 0 0"/>
```

Сама механическая система реализована набором вложенных тел (`body`) со шарнирами `joint` и твёрдыми телами `geom`. Левый опорный шарнир A:

```
<body name="ABC1P" pos="0 0 1.5" euler="90 0 0">
    <joint name="A" type="hinge" axis="0 0 1"
            stiffness ="0" springref="0" damping="0"/>
    <geom name="point A" type="cylinder" pos="0 0 0" size="0.02 0.02"/>
    <geom name="link AB" type="cylinder" pos="0 −0.075 0" size="0.015 0.075"/>
    ...
```

Внутри этого тела вложено тело с шарниром B и звеном BC:

```
<body name="BC1P" pos="0 −0.15 0">
    <joint name="B" type="hinge" axis="0 0 1"
            stiffness ="0" springref="0" damping="0.1"/>
    <geom name="point B" type="cylinder" pos="0 0 0" size="0.02 0.02"/>
    <geom name="link BC" type="cylinder" pos="0 −0.1875 0" size="0.015 0.1875"/>
    <site name="sC1" size="0.01" pos="0 −0.375 0"/>
    ...
```

Аналогично, правое опорное звено с шарниром D описывается отдельным телом:

```
<body name="DC2" pos="0.3 0 1.5" euler="90 0 0">
    <joint name="D" type="hinge" axis="0 0 1"
            stiffness ="0" springref="0" damping="0.1"/>
    <geom name="point D" type="cylinder" pos="0 0 0" size="0.02 0.02"/>
    <geom name="link DC" type="cylinder" pos="0 −0.1875 0" size="0.015 0.1875"/>
    <site name="sC2" size="0.01" pos="0 −0.375 0"/>
</body>
```

Важно отметить, что в шарнирах `B` и `D` задан коэффициент вязкого демпфирования `damping=0.1`. Именно он, а не явно заданный демпфер, моделирует пассивное демпфирование в системе.

Точка сочленения звеньев в точке C реализована не отдельным шарниром, а парой кинетически связываемых сайтов `sC1` и `sC2`. В секции `equality` эти сайты жёстко соединяются:

```
<equality>
    <connect site1="sC1" site2="sC2"/>
</equality>
```

Таким образом формируется замкнутая кинематическая цепь четырёхзвенника.

Управление осуществляется через позиционный привод в суставе A:

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

Наконец, для регистрации траектории конечной точки (энд-эффектора) используется датчик положения рамки, привязанный к сайту `sP`:

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

Именно эта сенсорная информация далее считывается в Python-скрипте и используется для анализа траектории и вычисления ошибки.

Получившаяся система представлена на изображении 2.1:

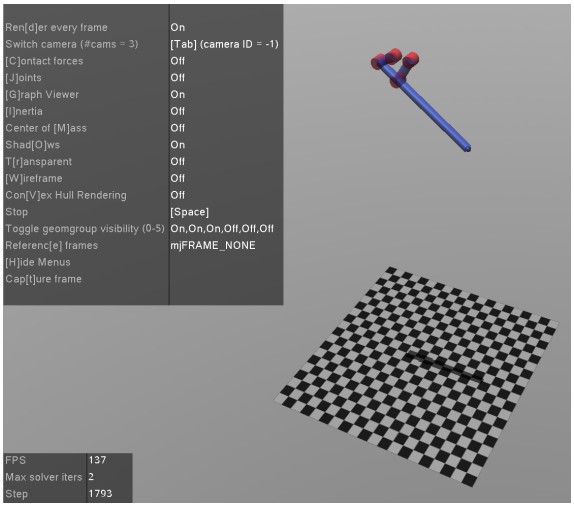

Рис. 2.1: Получившаяся система в среде MuJoCo

2.2 Программная реализация динамики и оптимизации

Для моделлирования динамики был разработан скрипт, ключевыми библиотеками которого стали следующие библиотеки: `mujoco`, `mujoco_viewer`, `scipy`.

Для параметрического изменения геометрии звеньев используется вспомогательная функция `swap_par`, которая находит элемент по типу и имени и перезаписывает указанный атрибут. Это позволяет из одного XML-шаблона получать разные конфигурации механизма:

```
def swap_par(tree, element_type, element_name, attribute_name, new_value):
    root = tree.getroot()
    element = None
    for elem in root.iter():
        if etree.QName(elem).localname == element_type \
           and elem.get("name") == element_name:
            element = elem
            break
    if element is None:
        raise ValueError(f'Element <{element_type} name="{element_name}"> not found')
    element.set(attribute_name, new_value)
```

Заданные длины звеньев (L_1–L_5) подставляются в модель изменением позиций и размеров соответствующих `geom` и `body`, после чего XML перезаписывается и загружается в MuJoCo:

```
L_AB = 0.064; L_BC = 0.0832; L_DC = 0.096
L_AD = 0.064; L_CP = 0.32

tree = etree.parse(f)
swap_par(tree, 'geom', 'link_AB', 'pos',  f"0 -{L_AB/2} 0")
swap_par(tree, 'geom', 'link_AB', 'size', f"0.015 {L_AB/2}")

tree.write(f, pretty_print=True, xml_declaration=True, encoding='UTF-8')

model = mujoco.MjModel.from_xml_path(f)
data  = mujoco.MjData(model)
```

Управляющее воздействие подаётся в сустав `A` через привод `<position joint="A»`. В скрипте реализован ПД-закон управления, который формирует значение элемента `data.ctrl[0]` на основе текущего угла и скорости шарнира и желаемой траектории $\theta_{\text{des}}(t)$:

```
def set_torque(mj_data, KP, KV, theta):
    data.ctrl[0] = KP * (theta - mj_data.qpos[0]) \
                 + KV * (0.0   - mj_data.qvel[0])

SIMEND   = 20.0
TIMESTEP = 0.001
STEP_NUM = int(SIMEND / TIMESTEP)
timeseries = np.linspace(0, SIMEND, STEP_NUM)

T    = 2.0
FREQ = 1.0 / T
AMP  = np.deg2rad(-90.0)
BIAS = np.deg2rad(-90.0)
theta_des = AMP * np.sin(FREQ * timeseries) + BIAS
```

Основной цикл симуляции на каждом шаге задаёт управляющее воздействие, выполняет интегрирование динамики и записывает координаты энд-эффектора (сайта `sP`):

```
EE_position_x = []
EE_position_z = []

viewer = mujoco_viewer.MujocoViewer(model, data,
                                    title="4bar",
                                    width=1920, height=1080)

for i in range(STEP_NUM):
    if not viewer.is_alive:
        break
    set_torque(data, 20.0, 5.0, theta_des[i])

    position_EE = data.site_xpos[1]
    EE_position_x.append(position_EE[0])
    EE_position_z.append(position_EE[2])
```

```
16
17      mujoco.mj_step(model, data)
18      viewer.render()
19  viewer.close()
```

Качество механизма оценивается по среднеквадратичному отклонению высоты точки P от заданного уровня h_z во второй половине траектории:

```
1  h_z = 0.98  # [m]
2  EEz = np.array(EE_position_z)
3  midlength = len(EEz) // 2
4  error  = np.mean((EEz[midlength:] - h_z)**2)
5  print(error)
```

Для автоматического подбора длин звеньев используется оптимизатор `scipy.optimize.minim`
Целевая функция `fourbar` по вектору параметров $X = [L_{AB}, L_{BC}, L_{DC}, L_{AD}, L_{CP}]$ перестраивает XML-модель, выполняет симуляцию и возвращает описанную выше ошибку траектории:

```
1  def fourbar(X):
2      L_AB, L_BC, L_DC, L_AD, L_CP = X
3      tree  = etree.parse(f)
4
5      swap_par(tree, 'geom', 'link  AB', 'pos',    f"0 -{L_AB/2} 0")
6      swap_par(tree, 'geom', 'link  AB', 'size',  f"0.015 {L_AB/2}")
7
8      tree.write(f,  pretty_print=True, xml_declaration=True, encoding='UTF-8')
9
10     model = mujoco.MjModel.from_xml_path(f)
11     data  = mujoco.MjData(model)
12
13     EE_z = []
14     for  i  in  range(STEP_NUM):
15         set_torque(data, 20.0,  5.0,  theta_des[i])
16         EE_z.append(data.site_xpos[1][2])
17         mujoco.mj_step(model, data)
18
19     EE_z = np.array(EE_z)
20     mid = len(EE_z) // 2
```

```
    return np.mean((EE_z[mid:] - h_z)**2)
```

Начальное приближение берётся из табличных длин, а также задаются ограничения на диапазон параметров и их соотношения:

```
X_init = np.array([0.064, 0.0832, 0.096, 0.064, 0.32])
bnds   = [(0.02, 0.4)] * 5
constraints = [
    {'type': 'eq', 'fun': lambda v: 2.5 * v[0] - v[1]},
    {'type': 'eq', 'fun': lambda v: 2.5 * v[0] - v[2]},
    {'type': 'eq', 'fun': lambda v: 2.0 * v[0] - v[3]},
    {'type': 'eq', 'fun': lambda v: 2.5 * v[0] - v[4]},
]
opts = {'ftol': 1e-7, 'maxiter': 100}

result = minimize(fourbar, X_init, method='SLSQP',
                  bounds=bnds, options=opts,
                  constraints=constraints)

print(f"X = {result.x}")
print(f"error = {result.fun}")
```

Таким образом, Python-скрипт реализует полный цикл: параметризацию XML-модели, возбуждение механизма заданным управляющим воздействием в суставе A, съём траектории энд-эффектора и численную оптимизацию геометрии звеньев по критерию минимизации вертикального отклонения точки P от заданной высоты.

2.3 Результаты

В случае системы с длинами звеньев заданными в условии лабораторной работы и представленными в таблице: ошибка составляет 0.0113 м, что вцелом не является большим отклонением (но и не является принебрежимым), но в контексте некоторых задач может стать критическим, поэтому проанализируем результат работы оптимизатора.

Таблица 2.1: Длины звеньев механизма по заданию

L_1, м	L_2, м	L_3, м	L_4, м	L_5, м
0,064	0,0832	0,096	0,064	0,32

После отработки оптимизатора с ограничениями указанными в предыдущем параграфе ошибка становится равной $3.3877 * 10^{-7}$м, однако при вышеупомянутых ограничениях слишком сильно меняется геометрия системы, а именно длины звеньев становятся равными:

Таблица 2.2: Длины звеньев механизма после оптимизации

L_1, м	L_2, м	L_3, м	L_4, м	L_5, м
0,13	0,3244	0,324	0,26	0,32

Траектория Энд-Эффектора представлена на изображении 2.2:

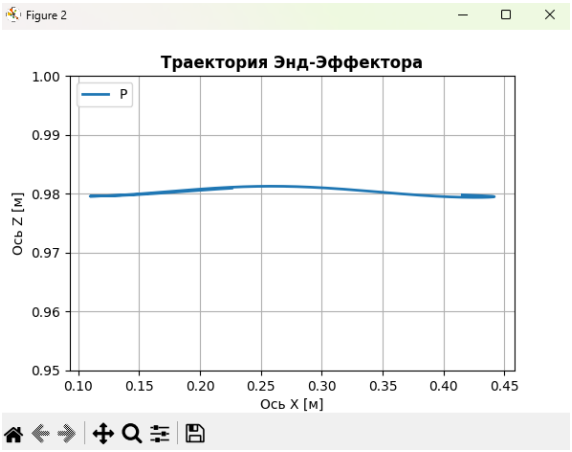

Рис. 2.2: Траектория Энд-Эффектора

3 Вывод

В данной работе была смоделирована динамика пассивного коленного механизма типа Optimus knee в среде MuJoCo на основе XML-описания и реализован Python-скрипт, позволяющий возбудить движение системы и регистрировать траекторию энд-эффектора. На основании исходных табличных значений длин звеньев была собрана базовая модель, для которой вычислено среднеквадратичное отклонение вертикальной координаты энд-эффектора от заданной высоты h_z. Для исходной геометрии эта ошибка составила порядка 0.0113 м, что демонстрирует заметное, но не критическое отклонение траектории от идеальной горизонтальной линии.

Далее был реализован численный поиск геометрических параметров звеньев с использованием оптимизатора `scipy.optimize.minimize`. В качестве целевой функции использовалась ошибка по высоте точки P, а в виде ограничений были заданы жёсткие соотношения между длинами звеньев. В результате работы оптимизатора ошибка была уменьшена до величины порядка $3.4 \cdot 10^{-7}$ м, то есть практически до нуля, что соответствует почти идеально горизонтальной траектории энд-эффектора на заданной высоте. Однако за столь малую ошибку пришлось заплатить существенным изменением исходной геометрии: оптимальные длины звеньев заметно отличаются от значений, заданных в условии лабораторной работы.

Таким образом, в ходе работы были показаны как возможности среды MuJoCo для моделирования многозвенных механизмов с демпфированием в шарнирах и сенсорикой, так и эффективность численной оптимизации для формирования требуемых траекторий. Вместе с тем стало очевидно, что при постановке задачи оптимизации необходимо аккуратно выбирать ограничения на параметры: слишком жёсткие соотношения могут приводить к нереалистичным изменениям конструкции, даже если формально достигается почти нулевая ошибка по выбранному критерию.

Список иллюстраций