МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Университет ИТМО

дисциплина

«Имитационное моделирование робототехнических систем»

Отчет по лабораторной работе № 4

Выполнил:
Смирнов И. Д. R4136c

Проверил:
Ракшин Е. А.

Санкт-Петербург

2025

Задание

1. Модифицировать модель из предыдущего задания, добавить привод<actuaor> и датчик <sensor>
2. Определите усилие управления с помощью PD-регулятора.

желаемая $q^{des} = AMP \cdot \sin(FREQ \cdot t) + BIAS$

с параметрами:

AMP, deg	FREQ, Hz	BIAS, deg
28.04	1.24	26.1

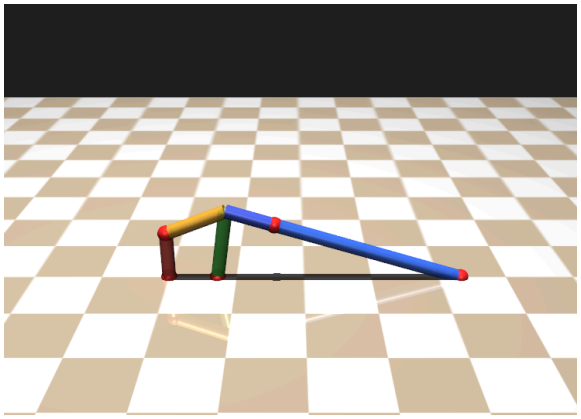
Ход работы

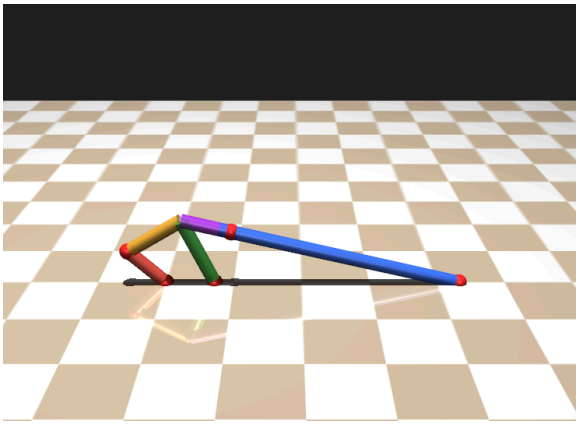
Доработаем конфигурацию механизма MuJoCo, добавив сенсор и актуатор:

```
<?xml version='1.0' encoding='UTF-8'?>
<mujoco model="Optimus Mechanism">
    <option timestep="1e-3" gravity="0 0 -9.8"/>

    <asset>
        <texture type="skybox" builtin="gradient" rgb1="0.08 0.08 0.08" rgb2="0.15 0.15 0.15"
width="256" height="256"/>
        <texture name="grid" type="2d" builtin="checker" rgb1="0.88 0.88 0.88" rgb2="0.72 0.65 0.55"
width="300" height="300"/>
        <material name="grid" texture="grid" texrepeat="10 10" reflectance="0.2"/>
        <material name="link_mat" reflectance="0.3"/>
    </asset>

    <worldbody>
        <light directional="false" pos="0 0 2.5" diffuse="0.9 0.9 0.9" specular="0.3 0.3 0.3"/>
        <geom type="plane" size="0.5 0.5 0.01" material="grid"/>

        <body name="OAB1" pos="0 0 0">
            <joint name="O" type="hinge" axis="0 -1 0" stiffness="0" springref="0" damping="0"/>

            <geom name="point_O" type="sphere" pos="0 0 0" size="0.006" rgba="0.95 0.15 0.15 1.0"
contype="0"/>
            <geom name="link_OA" type="cylinder" pos="0 0 0.018" size="0.005 0.018"
                  rgba="0.90 0.30 0.25 0.8" material="link_mat" contype="0"/>

            <body name="AB1" pos="0 0 0.036">
                <joint name="A" type="hinge" axis="0 -1 0" stiffness="0" springref="0"
damping="0.1"/>
                <geom name="point_A" type="sphere" pos="0 0 0" size="0.006" rgba="0.95 0.15 0.15
1.0" contype="0"/>
```

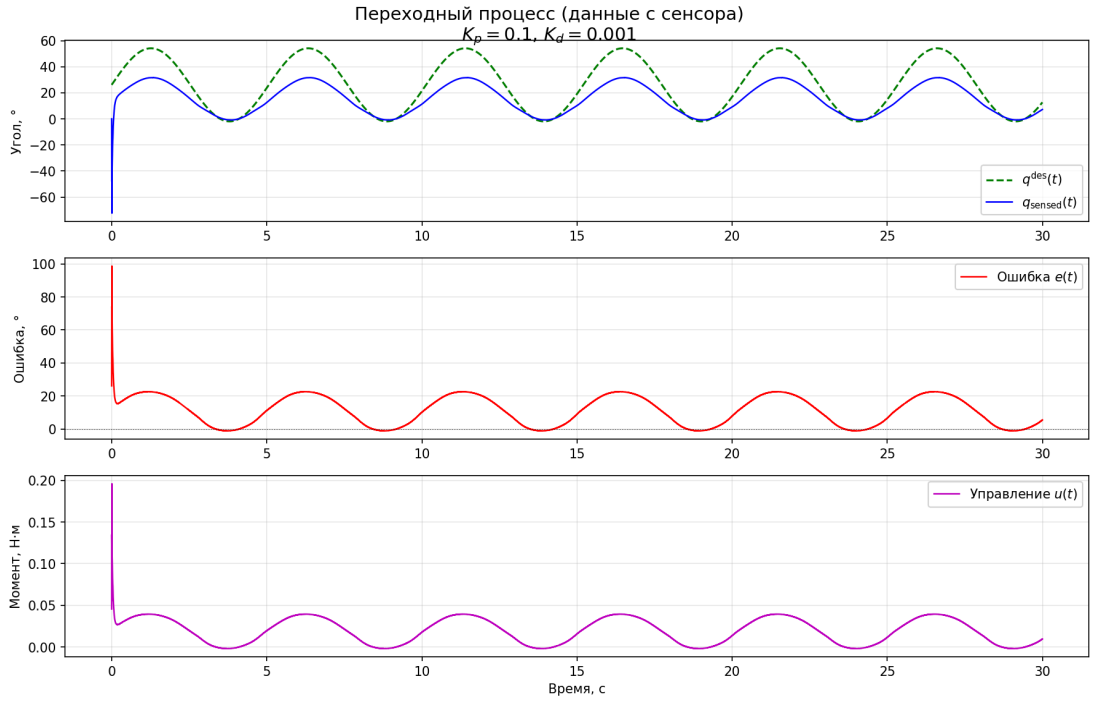
```
                <geom name="link_AB1" type="cylinder" pos="0 0 0.0234" size="0.005 0.0234"
                      rgba="0.90 0.65 0.20 0.8" material="link_mat" contype="0"/>
                <site name="sC1" pos="0 0 0.0468" size="0.003"/>
            </body>
        </body>

        <body name="CB2" pos="0.036 0 0">
            <joint name="C" type="hinge" axis="0 -1 0" stiffness="0" springref="0" damping="0.1"/>
            <geom name="point_C" type="sphere" pos="0 0 0" size="0.006" rgba="0.95 0.15 0.15 1.0"
contype="0"/>
            <geom name="link_CB2" type="cylinder" pos="0 0 0.027" size="0.005 0.027"
                  rgba="0.25 0.70 0.25 0.8" material="link_mat" contype="0"/>
            <site name="sC2" pos="0 0 0.054" size="0.003"/>
        </body>

        <body name="DFB3" pos="0.216 0 0">
            <joint name="D" type="hinge" axis="0 -1 0" stiffness="0" springref="0" damping="0"/>
            <geom name="point_D" type="sphere" pos="0 0 0" size="0.006" rgba="0.95 0.15 0.15 1.0"
contype="0"/>
            <geom name="link_DB3" type="cylinder" pos="0 0 0.09" size="0.005 0.09"
                  rgba="0.20 0.40 0.90 0.8" material="link_mat" contype="0"/>

            <body name="FB3" pos="0 0 0.18">
                <joint name="slider" type="slide" axis="0 0 1" limited="true"
                       range="-0.2 0.2" stiffness="0" springref="0" damping="0"/>
                <geom name="point_B3" type="sphere" pos="0 0 0" size="0.006" rgba="0.95 0.15 0.15
1.0" contype="0"/>
                <geom name="link_FB3" type="cylinder" pos="0 0 0.018" size="0.005 0.018"
                      rgba="0.60 0.20 0.80 0.8" material="link_mat" contype="0"/>
                <site name="sC3" pos="0 0 0.036" size="0.003"/>
            </body>
        </body>
    </worldbody>

    <actuator>
        <motor name="motor_O" joint="O" ctrlrange="-10 10" gear="1"/>
    </actuator>

    <sensor>
        <jointpos name="sensor_O_pos" joint="O"/>
    </sensor>
    <equality>
        <connect name="eq_C1_C2" site1="sC1" site2="sC2" solimp="0.995 0.99 0.001" solref="0.01 1"/>
        <connect name="eq_C1_C3" site1="sC1" site2="sC3" solimp="0.995 0.99 0.001" solref="0.01 1"/>
    </equality>
</mujoco>
```

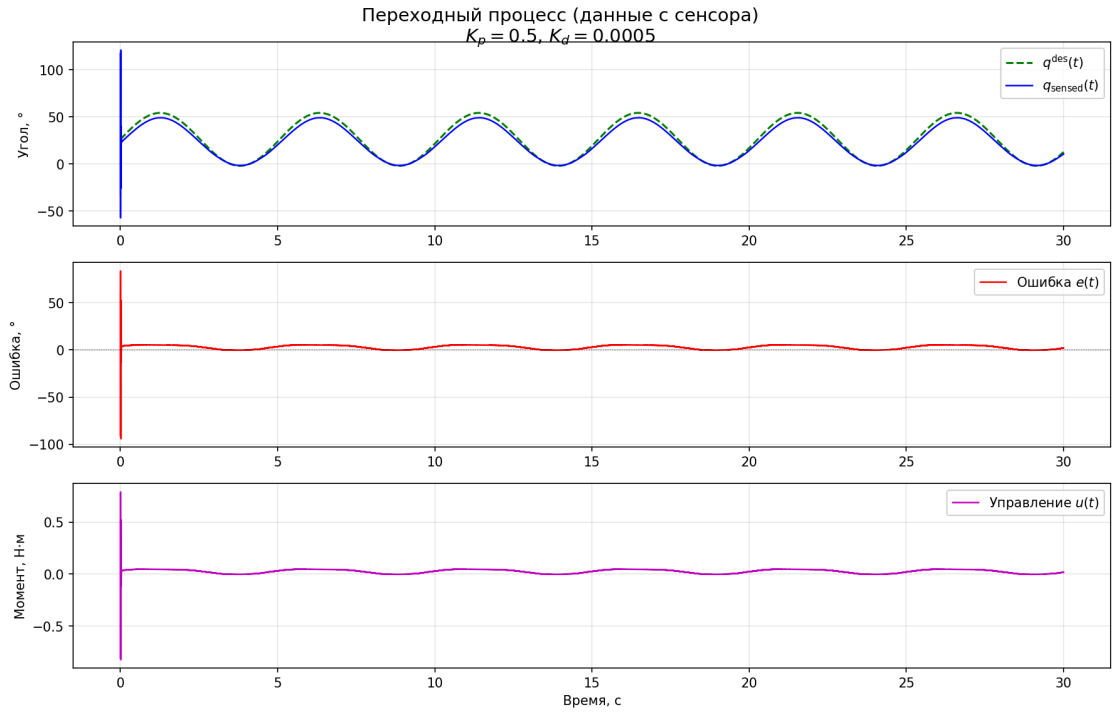
Получим подвижный механизм:

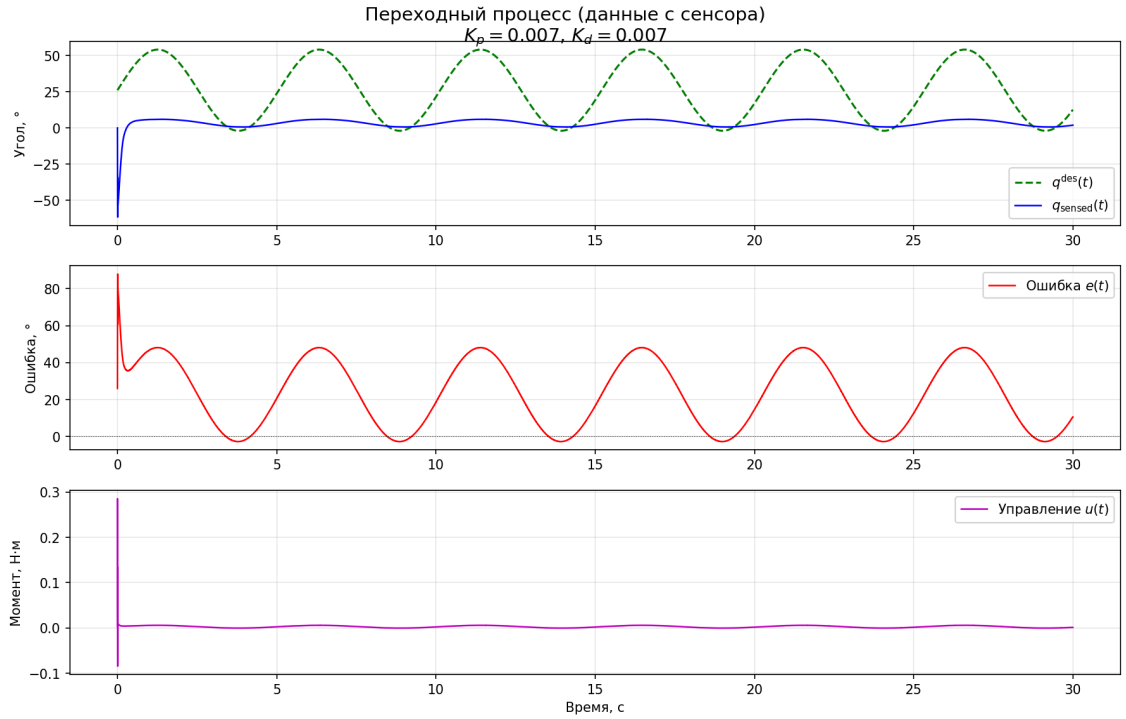




Снимем с него графики переходного процесса с различными коэффициентами ПД регулятора:




Переходный процесс (данные с сенсора)
$K_p = 0.5$, $K_d = 0.0005$
Угол, °
$q^{des}(t)$
$q_{sensed}(t)$
Ошибка, °
Ошибка $e(t)$
Момент, Н·м
Управление $u(t)$
Время, с


Переходный процесс (данные с сенсора)
$K_p = 0.007$, $K_d = 0.007$
Угол, °
$q^{des}(t)$
$q_{sensed}(t)$
Ошибка, °
Ошибка $e(t)$
Момент, Н·м
Управление $u(t)$
Время, с

Вывод

В рамках выполнения задания была успешно реализована численная симуляция динамики механизма Optimus в среде MuJoCo с последующим анализом переходного процесса при управлении по закону синусоидального изменения желаемого угла.

В исходный XML-файл добавлены секции `<actuator>` и `<sensor>`, что позволило ввести в систему один управляющий элемент (мотор, приводящий в движение шарнир `O`) и один датчик положения (`sensor_O_pos`). В управляющем скрипте на языке Python реализован пропорционально-дифференциальный закон управления, где желаемая траектория задана как

$$q^{des} = AMP \cdot \sin(FREQ \cdot t) + BIAS$$

Оценка скорости выполнена численно на основе дискретных измерений сенсора, что соответствует реалистичному сценарию использования одного энкодера.

Листинги программ:

```
import time
import mujoco
import mujoco.viewer
from colorama import Fore, Style
import pyfiglet
import numpy as np
import matplotlib.pyplot as plt

XML_PATH = "optimus.xml"

# Параметры желаемой траектории
AMP_DEG = 28.04
FREQ = 1.24
BIAS_DEG = 26.1
AMP_RAD = np.deg2rad(AMP_DEG)
BIAS_RAD = np.deg2rad(BIAS_DEG)

# ПД-регулятор
KP = 0.007
KD = 0.007


def get_sensor_value(model, data, sensor_name):
    sid = mujoco.mj_name2id(model, mujoco.mjtObj.mjOBJ_SENSOR, sensor_name)
    if sid == -1:
        raise ValueError(f"Сенсор '{sensor_name}' не найден")
    return data.sensordata[model.sensor_adr[sid]]


def main():
    print(Fore.RED + pyfiglet.figlet_format(''.join([chr(code) for code in [98, 121, 32, 73, 49]]),
                                            font="starwars") + Style.RESET_ALL)
    print("Загрузка модели...")
```

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

sensor_name = "sensor_0_pos"
try:
    _ = get_sensor_value(model, data, sensor_name)
    print(f"Сенсор '{sensor_name}' обнаружен.")
except Exception as e:
    print(f"Ошибка сенсора: {e}")
    return

# Инициализация
data.qpos[:] = 0.0
data.qvel[:] = 0.0

# Логгеры
log_time = []
log_q_des = []
log_q_sensed = []
log_ctrl = []



with mujoco.viewer.launch_passive(model, data) as viewer:
    viewer.cam.lookat[:] = np.array([0.09, 0.0, 0.0])
    viewer.cam.distance = 0.5
    viewer.cam.elevation = -30

    t0 = time.time()
    step = 0
    prev_sensed_pos = 0.0

    while viewer.is_running():
        t = time.time() - t0


        if t >= 30.0:
            break

        q_des = AMP_RAD * np.sin(FREQ * t) + BIAS_RAD

        sensed_pos = get_sensor_value(model, data, sensor_name)

        if step == 0:
            sensed_vel = 0.0
        else:
            dt = model.opt.timestep
            sensed_vel = (sensed_pos - prev_sensed_pos) / dt
        prev_sensed_pos = sensed_pos

        # ПД-управление
        u = KP * (q_des - sensed_pos) - KD * sensed_vel
        data.ctrl[0] = u

        # Шаг симуляции
        mujoco.mj_step(model, data)
        viewer.sync()


        if step % 10 == 0:
            log_time.append(t)
            log_q_des.append(np.rad2deg(q_des))
            log_q_sensed.append(np.rad2deg(sensed_pos))
```

```
                log_ctrl.append(u)

            step += 1


    n_points = len(log_time)


    if n_points == 0:
        print("Нет данных для построения графика!!!")
        return

    log_time = np.array(log_time)
    log_q_des = np.array(log_q_des)
    log_q_sensed = np.array(log_q_sensed)
    log_error = log_q_des - log_q_sensed

    plt.figure(figsize=(12, 8))

    plt.subplot(3, 1, 1)
    plt.plot(log_time, log_q_des, 'g--', label=r'$q^{\mathrm{des}}(t)$', linewidth=1.5)
    plt.plot(log_time, log_q_sensed, 'b', label=r'$q_{\mathrm{sensed}}(t)$', linewidth=1.2)
    plt.ylabel('Угол, °')
    plt.grid(True, alpha=0.3)
    plt.legend()

    plt.subplot(3, 1, 2)
    plt.plot(log_time, log_error, 'r', label='Ошибка $e(t)$', linewidth=1.2)
    plt.axhline(0, color='k', linewidth=0.5, linestyle=':')
    plt.ylabel('Ошибка, °')
    plt.grid(True, alpha=0.3)
    plt.legend()

    plt.subplot(3, 1, 3)
    plt.plot(log_time, log_ctrl, 'm', label='Управление $u(t)$', linewidth=1.2)
    plt.xlabel('Время, с')
    plt.ylabel('Момент, Н·м')
    plt.grid(True, alpha=0.3)
    plt.legend()

    # Подставляем KP и KD в заголовок
    plt.suptitle(f'Переходный процесс (данные с сенсора)\n$K_p = {KP}$, $K_d = {KD}$',
                 fontsize=14, y=0.95)

    plt.tight_layout()

    plt.savefig("transition_30s.png", dpi=150, bbox_inches='tight')
    print("График сохранён: transition_30s.png")
    plt.show()


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