МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО» (Университет ИТМО)

Факультет систем управления и робототехники

Лабораторная работа №4
по дисциплине
«Имитационное моделирование робототехнических систем»

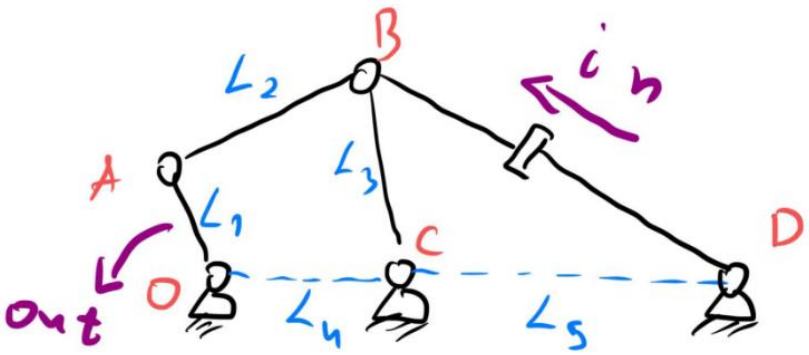
Студент:
Группа № R4133c *М. В. Рогальский*

Преподаватель:
Ассистент ФСУиР *Е. А. Ракшин*

Санкт-Петербург 2025

Задание:

1. To the model you have created in the previous task, you need to add actuators. For Optimus mechanism there is one actuator (q1), for tendon mechanism there are two actuators (q1 and q2).
2. Modify the .xml file by adding <actuator> and <sensor> containers (look at the examples in the previous task).
3. Define control effort via PD regulator. The $q^{des} = AMP \cdot sin(FREQ \cdot t) + BIAS$. Look in the table for sine wave parameters. If the control sequence goes beyond workspace of the mechanism, decrease the amplitude and tune bias only if needed.

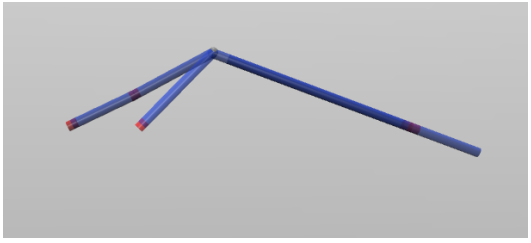


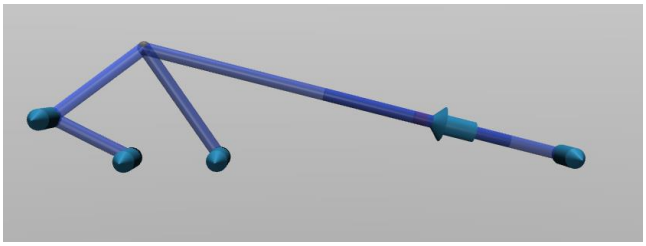
Данные по варианту (506153):

AMP, deg	FREQ, Hz	BIAS, deg
19.83	2.11	23.6

Решение:

Реализация механизма была в прошлой лабораторной работе. Вот, как выглядела модель:





Добавим контейнеры <actuator> для входного воздействия к joint “slider” и <sensor> для отслеживания выходных значений к site “sA” после <worldbody>

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

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

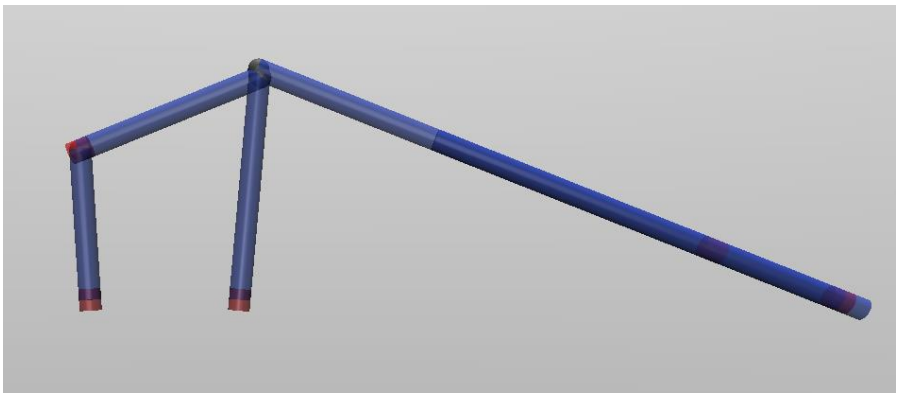
В программную часть добавим следующее (полный код см. в конце):

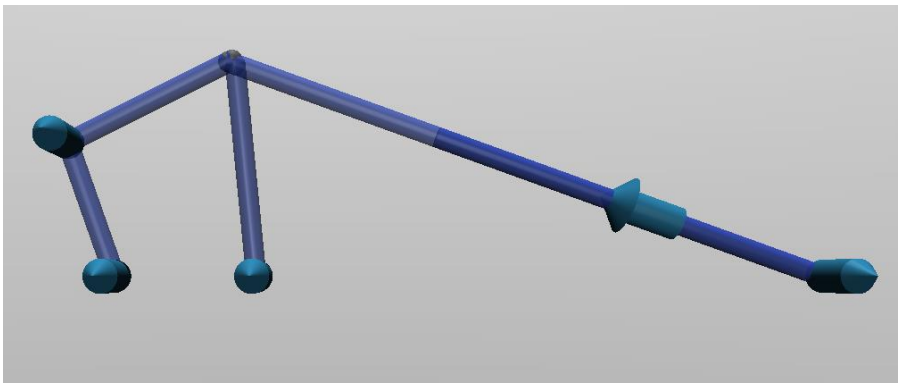
```
def set_torque(mj_data, KP, KV, theta):
    d.ctrl[0] = KP * (-mj_data.qpos[0] + theta) + KV * (0 - mj_data.qvel[0])

SIMEND = 50
TIMESTEP = 0.01
STEP_NUM = int(SIMEND / TIMESTEP)
timeseries = np.linspace(0, SIMEND, STEP_NUM)

#T = 2 # [s]
FREQ = 2.11 # [Hz]
AMP = np.deg2rad(19.83) # [rad]
BIAS = np.deg2rad(23.6) # [rad]

theta_des = AMP * np.sin(FREQ * timeseries) + BIAS


qpos_log = []
qvel_log = []
ctrl_log = []
theta_des_log = []

EE_position_x = []
EE_position_z = []

…

В цикле:

set_torque(d, 25, 10, theta_des[i])

mujoco.mj_step(m, d)
viewer.render()

qpos_log.append(d.qpos[0])
qvel_log.append(d.qvel[0])
ctrl_log.append(d.ctrl[0])
position_EE = d.site_xpos[1]
EE_position_x.append(position_EE[0])
```

```
EE_position_z.append(position_EE[2])
theta_des_log.append(theta_des[i])
```

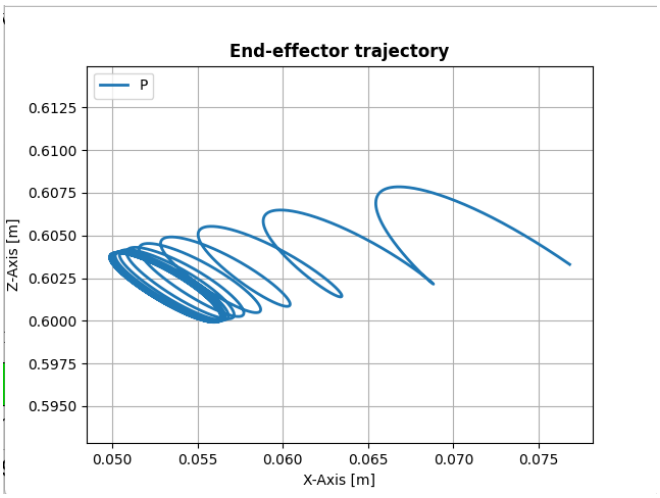
Результаты работы программы

Было задано входное воздействие и сняты положения в списки для последующего отображения в графики, так же установлен PD регулятор с коэффициентами KP=25, KV=10, что позволило как более возможно стабилизировать модель. Изображения модели и графиков:

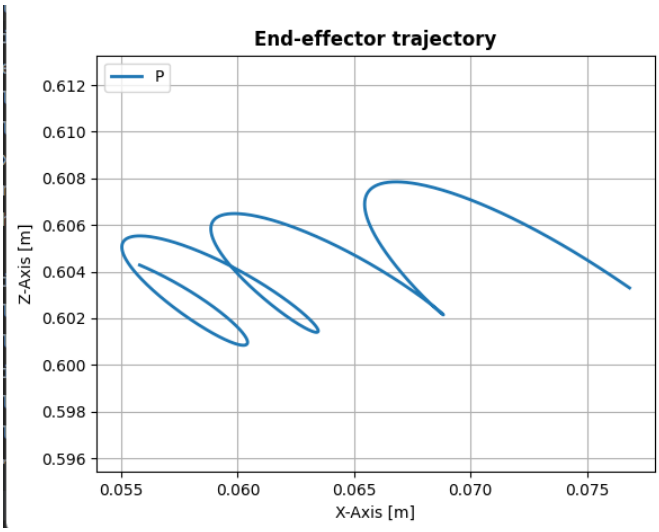




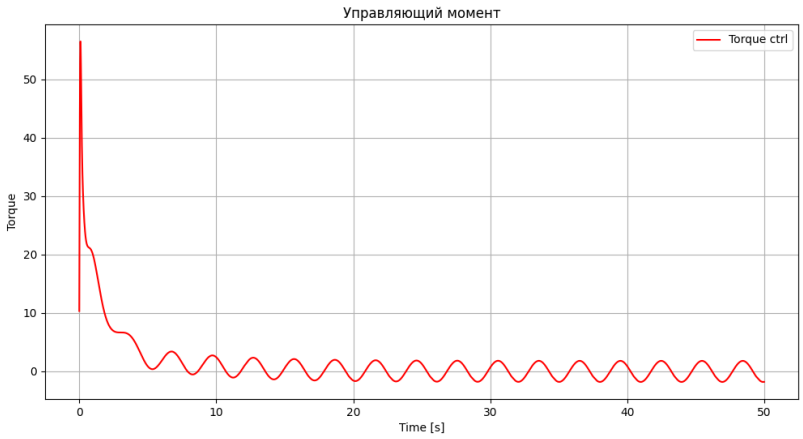
По положению видим, как ОА стабилизируется в левой части, «уходя» с правой.

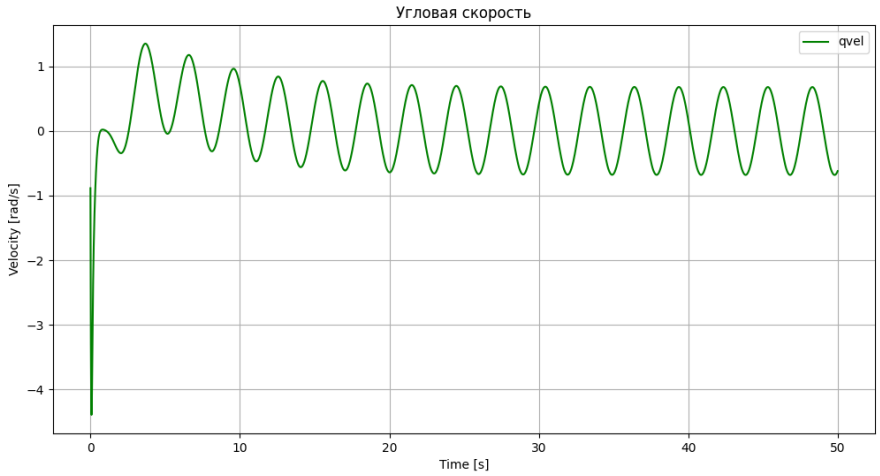


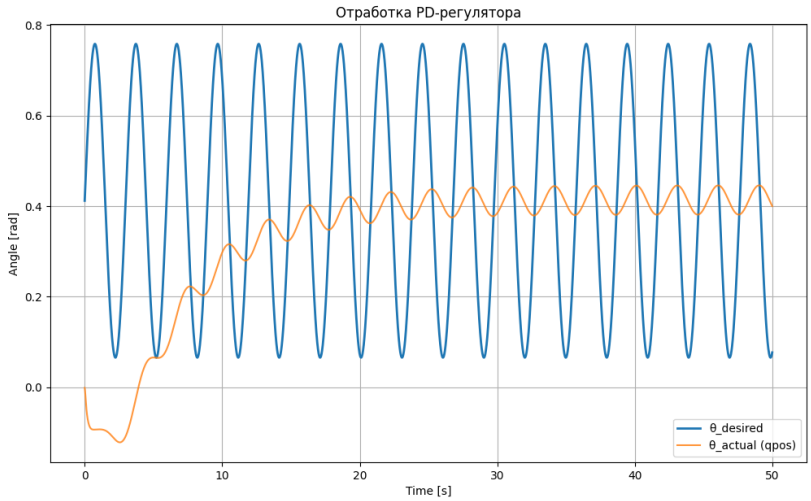
Взяв меньшее число шагов моделирования, видим, как положение идет справа налево:



Графики управляющего момента, угловой скорости и положения при PD регуляторе (видим стабилизирование):







Вывод:

Таким образом, было задано желаемое воздействие с PD регулированием к модели замкнутой кинематики при помощи <actuator>, <sensor> и программной части. Продемонстрированы полученные результаты.

Листинг optimus.xml:

```
    <?xml version='1.0' encoding='UTF-8'?>
<mujoco>

    <option timestep="1e-3"/>
    <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>
    <camera name="side view" pos="0.1 -0.4 0.5" euler="90 0 0" fovy="60"/>
    <light pos="0 0 10"/>
    <geom type="plane" size="0.5 0.5 0.1" material="grid"/>

    <body name="OAB1" pos="0 0 0.5" euler="0 0 0">

            <joint name="O" type="hinge" axis="0 -1 0" stiffness="0" springref="0"
damping="0.1"/>
            <geom name="point O" type="cylinder" pos="0 0 0" size="0.005 0.005"
rgba="0.89 0.14 0.16 0.5" euler="0 0 0" contype="0"/>
            <geom  name="link  OA"  type="cylinder"  pos="0  0  0.0345"  size="0.005
0.0345" rgba="0.21 0.32 0.82 0.5" euler="0 0 0" contype="0"/>
            <site name="sA" size="0.005" pos="0 0 0.069"/>

            <body name="AB1" pos="0 0 0.069" euler="0 0 0">

                <joint   name="A"   type="hinge"   axis="0   -1   0"   stiffness="0"
springref="0" damping="0.1"/>
                <geom  name="point  B"  type="cylinder"  pos="0  0  0"  size="0.005
0.005" rgba="0.89 0.14 0.16 0.5" euler="0 0 0" contype="0"/>
                <geom   name="link   AB1"   type="cylinder"   pos="0   0   0.04485"
size="0.005 0.04485" rgba="0.21 0.32 0.82 0.5" euler="0 0 0" contype="0"/>
                <site name="sC1" size="0.005" pos="0 0 0.0897"/>
```

```
            </body>
    </body>

    <body name="CB2" pos="0.069 0 0.5" euler="0 0 0">
        <joint name="C" type="hinge" axis="0 -1 0" stiffness="0" springref="0"
damping="0.1"/>
        <geom name="point C" type="cylinder" pos="0 0 0" size="0.005 0.005"
rgba="0.89 0.14 0.16 0.5" euler="0 0 0" contype="0"/>
        <geom name="link CB2" type="cylinder" pos="0 0 0.05175" size="0.005
0.05175" rgba="0.21 0.32 0.82 0.5" euler="0 0 0" contype="0"/>
        <site name="sC2" size="0.005" pos="0 0 0.1035"/>
    </body>

    <body name="DFB3" pos="0.345 0 0.5" euler="0 0 0">
        <joint name="D" type="hinge" axis="0 -1 0" stiffness="0" springref="0"
damping="0.1"/>
        <geom name="point D" type="cylinder" pos="0 0 0" size="0.005 0.005"
rgba="0.89 0.14 0.16 0.5" euler="0 0 0" contype="0"/>
        <geom name="link DB3" type="cylinder" pos="0 0 0.1" size="0.005 0.1"
rgba="0.21 0.32 0.82 0.5" euler="0 0 0" contype="0"/>

        <body name="FB3" pos="0 0 0.075" euler="0 0 0">
            <joint name="slider" type="slide" axis="0 0 1" limited="true" range="-
0.2 0.2" stiffness="0" springref="0" damping="0.1"/>
            <geom name="point B3" type="cylinder" pos="0 0 0" size="0.005 0.005"
rgba="0.89 0.14 0.16 0.5" euler="0 0 0" contype="0"/>
            <geom name="link FB3" type="cylinder" pos="0 0 0.075" size="0.005
0.15" rgba="0.21 0.32 0.82 0.5" euler="0 0 0" contype="0"/>
            <site name="sC3" size="0.005" pos="0 0 0.225"/>
        </body>
    </body>

    </worldbody>

    <equality>
        <connect site1="sC1" site2="sC2"/>
        <connect site1="sC1" site2="sC3"/>
    </equality>
```

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

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

</mujoco>
```

Листинг main.py:

```
import time

import matplotlib.pyplot as plt

import mujoco

import mujoco.viewer

import mujoco_viewer

import numpy as np


paused = False



# def key_callback(keycode):

#     if chr(keycode) == ' ':

#         global paused

#         paused = not paused



m = mujoco.MjModel.from_xml_path('optimus.xml')

d = mujoco.MjData(m)
```

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


SIMEND = 50
TIMESTEP = 0.01
STEP_NUM = int(SIMEND / TIMESTEP)
timeseries = np.linspace(0, SIMEND, STEP_NUM)


#T = 2 # [s]
FREQ = 2.11 # [Hz]
AMP = np.deg2rad(19.83) # [rad]
BIAS = np.deg2rad(23.6) # [rad]


theta_des = AMP * np.sin(FREQ * timeseries) + BIAS


qpos_log = []
qvel_log = []
ctrl_log = []
theta_des_log = []


EE_position_x = []
EE_position_z = []


viewer = mujoco_viewer.MujocoViewer(m,
                                    d,
                                    title="OPTIMUS",
                                    width=1920,
                                    height=1080)
```

```
start_time = time.time()


for i in range(STEP_NUM):
    if not viewer.is_alive:
        break


    step_start = time.time()  # время начала шага


    if not paused:
        set_torque(d, 25, 10, theta_des[i])


        mujoco.mj_step(m, d)
        viewer.render()


        qpos_log.append(d.qpos[0])
        qvel_log.append(d.qvel[0])
        ctrl_log.append(d.ctrl[0])
        position_EE = d.site_xpos[1]
        EE_position_x.append(position_EE[0])
        EE_position_z.append(position_EE[2])
        theta_des_log.append(theta_des[i])


        time_until_next_step = m.opt.timestep - (time.time() - step_start)
        if time_until_next_step > 0:
            time.sleep(time_until_next_step)


viewer.close()
```

```
t = np.arange(len(theta_des_log)) * TIMESTEP  # правильная ось времени


midlength = int(STEP_NUM/2)



plt.plot(EE_position_x[600:], EE_position_z[600:], '-', linewidth=2, label='P')
plt.title('End-effector trajectory', fontsize=12, fontweight='bold')
plt.legend(loc='upper left')
plt.xlabel('X-Axis [m]')
plt.ylabel('Z-Axis [m]')
plt.axis('equal')
plt.grid()
plt.show()


plt.figure(figsize=(12, 7))
plt.plot(t, theta_des_log, label="θ_desired", linewidth=2)
plt.plot(t, qpos_log, label="θ_actual (qpos)", alpha=0.8)
plt.title("Отработка PD-регулятора")
plt.xlabel("Time [s]")
plt.ylabel("Angle [rad]")
plt.grid(True)
plt.legend()


plt.figure(figsize=(12, 6))
plt.plot(t, qvel_log, label="qvel", color="green")
plt.title("Угловая скорость")
plt.xlabel("Time [s]")
plt.ylabel("Velocity [rad/s]")
plt.grid(True)
```

```
plt.legend()


plt.figure(figsize=(12, 6))
plt.plot(t, ctrl_log, label="Torque ctrl", color="red")
plt.title("Управляющий момент")
plt.xlabel("Time [s]")
plt.ylabel("Torque")
plt.grid(True)
plt.legend()


plt.show()
```