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

Отчет
по практической работе № 4
по дисциплине «Имитационное моделирование робототехнических систем»

Автор: Михин Н. С.
Факультет: СУиР
Группа: R4150
Вариант (ИСУ) № 335259
Преподаватель: Ракшин Е. А.

Санкт-Петербург, 2025

Условия варианта:

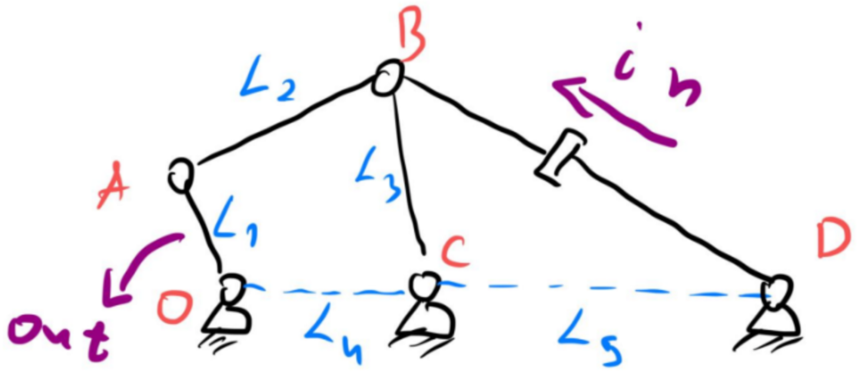


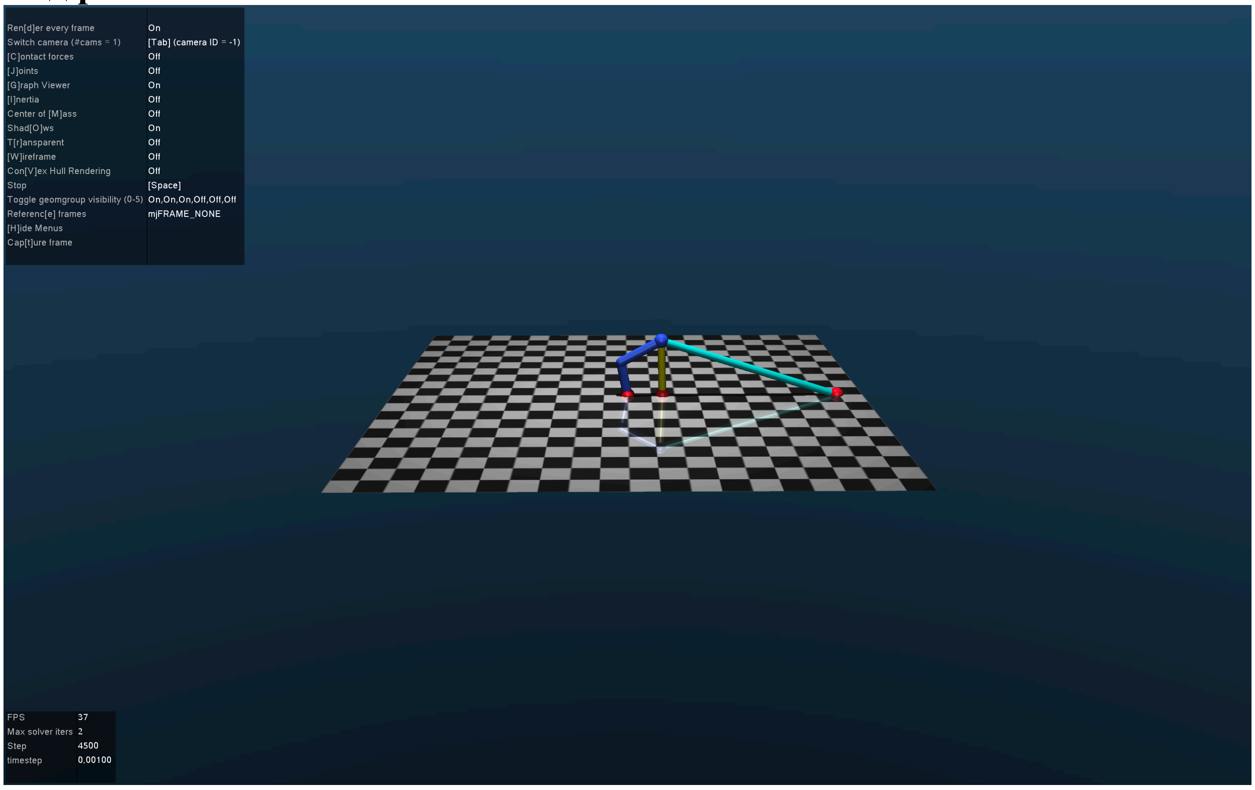
Рисунок 1 - OPTIMUS

$$AMP = 39.6$$

$$FREQ = 1.29$$

$$BIAS = 30.4$$

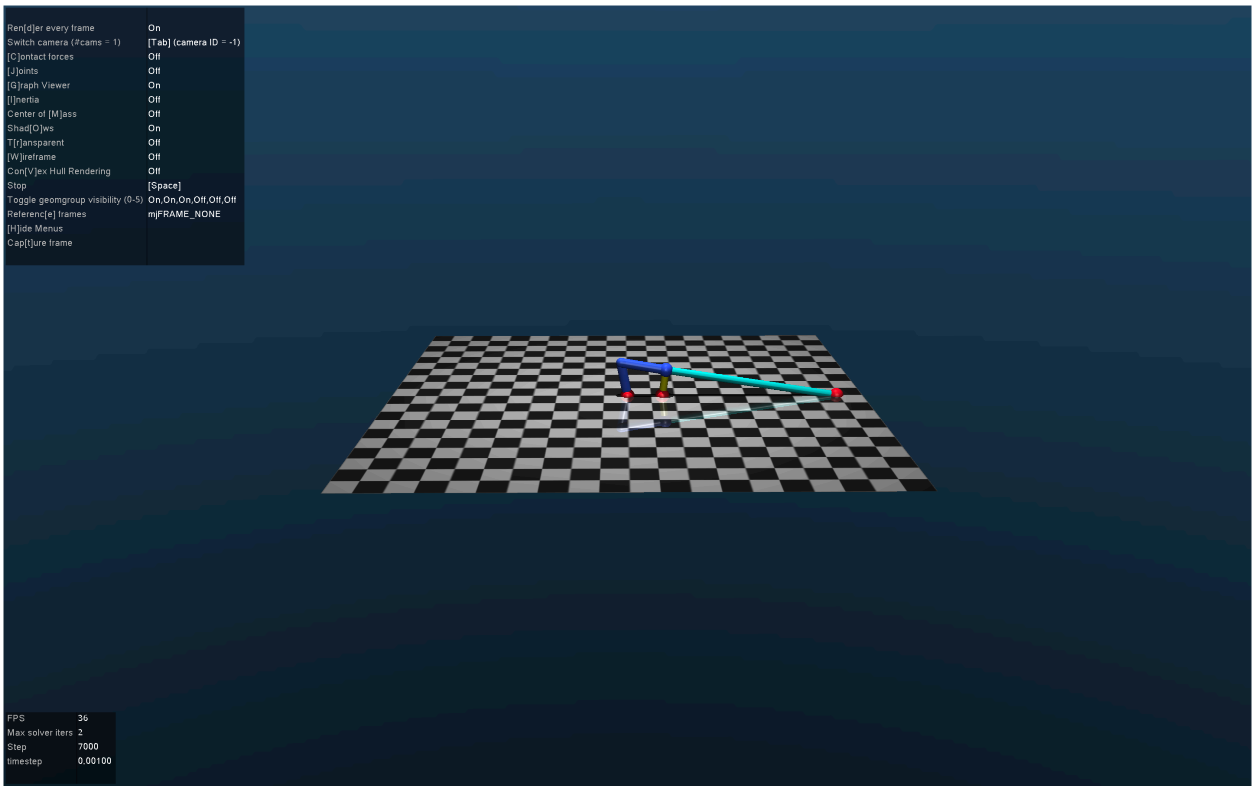
Ход работы:



```
Ren[d]er every frame                 On
Switch camera (#cams = 1)            [Tab] (camera ID = -1)
[C]ontact forces                     Off
[J]oints                             Off
[G]raph Viewer                       On
[I]nertia                            Off
Center of [M]ass                     Off
Shad[O]ws                            On
T[r]ansparent                        Off
[W]ireframe                          Off
Con[V]ex Hull Rendering              Off
Stop                                 [Space]
Toggle geomgroup visibility (0-5)    On,On,On,Off,Off,Off
Referenc[e] frames                   mjFRAME_NONE
[H]ide Menus
Cap[t]ure frame
```

```
FPS               37
Max solver iters  2
Step              4500
timestep          0.00100
```

Рисунок 2 – Колебательный процесс, первое положение



```
Ren[d]er every frame                 On
Switch camera (#cams = 1)            [Tab] (camera ID = -1)
[C]ontact forces                     Off
[J]oints                             Off
[G]raph Viewer                       On
[I]nertia                            Off
Center of [M]ass                     Off
Shad[O]ws                            On
T[r]ansparent                        Off
[W]ireframe                          Off
Con[V]ex Hull Rendering              Off
Stop                                 [Space]
Toggle geomgroup visibility (0-5)    On,On,On,Off,Off,Off
Referenc[e] frames                   mjFRAME_NONE
[H]ide Menus
Cap[t]ure frame
```

```
FPS               36
Max solver iters  2
Step              7000
timestep          0.00100
```

Рисунок 3 – Колебательный процесс - второе положение

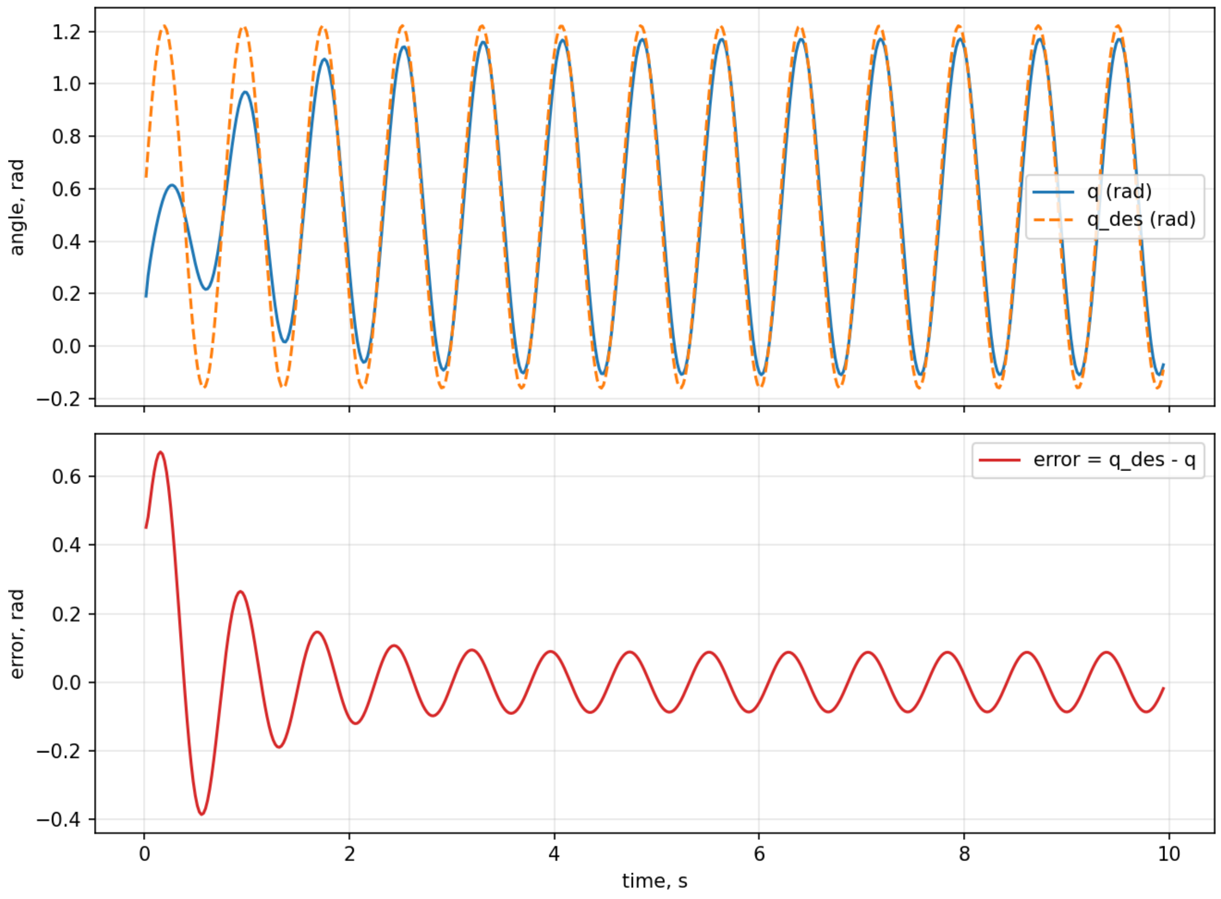


График 1 - Переходный процесс и ошибка

Листинг 1 - XML модели

```
<?xml version="1.0" encoding="UTF-8"?>
<mujoco model="optimus_knee_task4">
    <compiler angle="radian" inertiafromgeom="true"/>

    <option integrator="Euler" timestep="0.001" gravity="0 0 0"/>

    <visual>
        <map znear="0.01" zfar="50"/>
        <rgba haze="0.15 0.25 0.35 1"/>
    </visual>

    <asset>
        <texture type="skybox" builtin="gradient"
                 rgb1="0.3 0.5 0.7" rgb2="0 0 0"
                 width="320" height="320"/>
        <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 2" dir="0 0 -1" diffuse="1 1 1"/>
        <geom name="ground" type="plane" size="0.5 0.5 0.1"
material="grid"/>

        <site name="site_O" pos="0 0 0"       size="0.015" rgba="0.9
0.2 0.2 1"/>
        <site name="site_C" pos="0.074 0 0"   size="0.015" rgba="0.9
0.2 0.2 1"/>
        <site name="site_D" pos="0.444 0 0"   size="0.015" rgba="0.9
0.2 0.2 1"/>

        <body name="L1_body" pos="0 0 0">
            <inertial pos="0 0 0.037" mass="0.2"
                      diaginertia="0.01 0.01 0.001"/>
            <joint name="J_O" type="hinge" axis="0 1 0"
                   limited="true" range="-1e-6 1e-6" damping="0.5"/>
            <geom name="G_L1" type="capsule"
                  fromto="0 0 0   -0.0142133 0 0.0726222"
                  size="0.01" rgba="0.21 0.32 0.82 0.9"/>

            <body name="L2_body" pos="-0.0142133 0 0.0726222">
                <inertial pos="0 0 0.0481" mass="0.2"
                          diaginertia="0.01 0.01 0.001"/>
                <joint name="J_A" type="hinge" axis="0 1 0"
                       limited="true" range="-2.5 2.5"
damping="0.5"/>
                <geom name="G_L2" type="capsule"
                      fromto="0 0 0   0.0882133 0 0.0383778"
                      size="0.01" rgba="0.21 0.32 0.82 0.9"/>

                <body name="B_body" pos="0.0882133 0 0.0383778">
                    <site name="site_B" pos="0 0 0"
                          size="0.015" rgba="0.2 0.3 0.9 1"/>
                </body>
            </body>
        </body>
    </worldbody>

    <tendon>
        <spatial name="tendon_BC" width="0.008"
```

```xml
                  rgba="0.8 0.8 0.2 1"
                  stiffness="0.0"
                  damping="0.0"
                  springlength="0.111">
             <site site="site_B"/>
             <site site="site_C"/>
         </spatial>

         <spatial name="tendon_BD" width="0.008"
                  rgba="0.2 0.8 0.8 1"
                  limited="false">
             <site site="site_B"/>
             <site site="site_D"/>
         </spatial>
     </tendon>

     <actuator>
         <motor name="q1_motor" joint="J_A" gear="1"
                ctrllimited="true" ctrlrange="-20 20"/>
     </actuator>

     <sensor>
         <jointpos name="q1_pos" joint="J_A"/>
         <jointvel name="q1_vel" joint="J_A"/>
     </sensor>

</mujoco>
```

Листинг 2 - Код для моделирования

```
import os
import math
import numpy as np

import mujoco
import mujoco_viewer
import matplotlib.pyplot as plt

AMP_DEG = 39.6
FREQ_HZ = 1.29
BIAS_DEG = 30.4

AMP = math.radians(AMP_DEG)
BIAS = math.radians(BIAS_DEG)
OMEGA = 2.0 * math.pi * FREQ_HZ

KP = 25.0
KD = 2.0

REF_RAMP_TAU = 0.8

VEL_ERR_MAX = 3.0


def make_controller(model: mujoco.MjModel, data: mujoco.MjData):
    jnt_id = mujoco.mj_name2id(model, mujoco.mjtObj.mjOBJ_JOINT,
"J_A")
    act_id = mujoco.mj_name2id(model, mujoco.mjtObj.mjOBJ_ACTUATOR,
"q1_motor")
    if jnt_id < 0 or act_id < 0:
        raise RuntimeError("Failed to resolve joint/actuator IDs
(J_A / q1_motor)")

    qpos_addr = model.jnt_qposadr[jnt_id]
    qvel_addr = model.jnt_dofadr[jnt_id]

    ctrl_min = float(model.actuator_ctrlrange[act_id, 0]) if
model.actuator_ctrllimited[act_id] else -np.inf
    ctrl_max = float(model.actuator_ctrlrange[act_id, 1]) if
model.actuator_ctrllimited[act_id] else np.inf

    u_prev = 0.0
    alpha_u = 0.3

    def controller(_model, _data):
        nonlocal u_prev
        t = float(_data.time)

        amp_scale = 1.0 - math.exp(-t / REF_RAMP_TAU)
        q_des = (AMP * amp_scale) * math.sin(OMEGA * t) + BIAS
        dq_des = (AMP * amp_scale) * OMEGA * math.cos(OMEGA * t)

        q = float(_data.qpos[qpos_addr])
        dq = float(_data.qvel[qvel_addr])

        v_err = max(min(dq_des - dq, VEL_ERR_MAX), -VEL_ERR_MAX)
        u_raw = KP * (q_des - q) + KD * v_err

        u = alpha_u * u_raw + (1.0 - alpha_u) * u_prev
        u = max(min(u, ctrl_max), ctrl_min)
        u_prev = u

        u = max(min(u, ctrl_max), ctrl_min)
        _data.ctrl[act_id] = u

    return controller


def main():
    here = os.path.dirname(os.path.abspath(__file__))
    xml_path = os.path.join(here, "optimus.xml")

    print(f"Loading model: {xml_path}")
    model = mujoco.MjModel.from_xml_path(xml_path)
    data = mujoco.MjData(model)

    mujoco.mj_forward(model, data)

    mujoco.set_mjcb_control(make_controller(model, data))

    jnt_id = mujoco.mj_name2id(model, mujoco.mjtObj.mjOBJ_JOINT,
"J_A")
    qpos_addr = model.jnt_qposadr[jnt_id]
    qvel_addr = model.jnt_dofadr[jnt_id]

    act_id = mujoco.mj_name2id(model, mujoco.mjtObj.mjOBJ_ACTUATOR,
"q1_motor")
    if act_id < 0:
        raise RuntimeError("Actuator q1_motor not found")
    ctrl_min = float(model.actuator_ctrlrange[act_id, 0]) if
model.actuator_ctrllimited[act_id] else -np.inf
    ctrl_max = float(model.actuator_ctrlrange[act_id, 1]) if
model.actuator_ctrllimited[act_id] else np.inf

    times = []
    q_traj = []
    dq_traj = []
    qdes_traj = []
    dqdes_traj = []
    err_traj = []
    u_traj = []
    sat_traj = []
    LOG_DECIMATE = 20
    step_count = 0

    title = "Practice 4 — Optimus knee q1 PD"
    viewer = mujoco_viewer.MujocoViewer(model, data, title=title)
    try:
        print("Running simulation. Press ESC in the viewer to
quit.")
        STEPS_PER_RENDER = 10
        while True:
            alive_attr = getattr(viewer, "is_alive")
            alive = alive_attr if isinstance(alive_attr, bool) else
alive_attr()
            if not alive:
                break
            for _ in range(STEPS_PER_RENDER):
                mujoco.mj_step(model, data)
                step_count += 1
                if step_count % LOG_DECIMATE == 0:
                    t = float(data.time)
                    q = float(data.qpos[qpos_addr])
                    dq = float(data.qvel[qvel_addr])
                    q_des = AMP * math.sin(OMEGA * t) + BIAS
                    dq_des = AMP * OMEGA * math.cos(OMEGA * t)
                    e = q_des - q
                    v_err = max(min(dq_des - dq, VEL_ERR_MAX), -
VEL_ERR_MAX)
                    u = KP * e + KD * v_err
                    saturated = False
                    if u > ctrl_max:
                        u = ctrl_max
                        saturated = True
                    elif u < ctrl_min:
                        u = ctrl_min
                        saturated = True

                    times.append(t)
                    q_traj.append(q)
                    dq_traj.append(dq)
                    qdes_traj.append(q_des)
                    dqdes_traj.append(dq_des)
                    err_traj.append(e)
                    u_traj.append(u)
                    sat_traj.append(1 if saturated else 0)
            viewer.render()
    finally:
        viewer.close()

    if len(times) > 1:
        here = os.path.dirname(os.path.abspath(__file__))
        png_path = os.path.join(here, "timeseries.png")
        csv_path = os.path.join(here, "timeseries.csv")

        fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(9, 7),
sharex=True)
        ax1.plot(times, q_traj, label="q (rad)")
        ax1.plot(times, qdes_traj, '--', label="q_des (rad)")
        ax1.set_ylabel("angle, rad")
        ax1.grid(True, alpha=0.3)
        ax1.legend()

        ax2.plot(times, err_traj, color='C3', label="error = q_des -
q")
        ax2.set_xlabel("time, s")
        ax2.set_ylabel("error, rad")
        ax2.grid(True, alpha=0.3)
        ax2.legend()

        fig.suptitle("Optimus knee q1: angle and error")
        fig.tight_layout()
        fig.savefig(png_path, dpi=150)
        plt.close(fig)

        import numpy as np
        arr = np.column_stack([times, q_traj, dq_traj, qdes_traj,
dqdes_traj, err_traj, u_traj, sat_traj])
        np.savetxt(
            csv_path,
            arr,
            delimiter=",",
            header="time,q,dq,q_des,dq_des,error,u,sat",
            comments="",
        )
        print(f"Saved plot: {png_path}")
        print(f"Saved data: {csv_path}")
    else:
        print("No samples collected for plotting (simulation too
short).")


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