THE MINISTRY OF SCIENCE AND HIGHER EDUCATION

OF THE RUSSIAN FEDERATION

ITMO University

(ITMO)

Faculty of Control Systems and Robotics

Project

For the Subject:

SIMULATION OF ROBOTICS SYSTEMS

# Project Title

# Modeling and Simulating a Hybrid Wheel–Leg Robot Using MuJoCo

**Student:**

**ISU No. 476937– Nagham Sleman**

**ISU No. 503259 – Hazem Afif**

**Tutor:**

**Professor – Borisov, Ivan**

**Assistant – Rakshin, Egor**

**Date: 17.12.2025**

## Abstract

This project presents the development and simulation of a hybrid wheeled–legged robot using the MuJoCo physics engine and a Python-based real-time control framework. The robot model incorporates a pair of actuated wheels and four articulated legs, each equipped with multi-joint kinematic chains that enable both rolling and walking locomotion. A finite state machine was implemented to manage multiple modes of motion, including wheel-based driving, in-place rotation, forward and backward trotting gaits, and smooth transitions between states. Sinusoidal joint trajectories were used to generate stable legged locomotion, while velocity-based control governed the wheel dynamics. Real-time logging of torques and joint angles enabled detailed evaluation of system behavior. The results demonstrate stable and coordinated movement in all modes, with smooth actuator transitions and consistent gait patterns, confirming the validity of both the robot model and the control architecture. This work establishes a functional foundation for future improvements such as feedback control, trajectory optimization, and hardware implementation.

## 1. Introduction

Hybrid wheel–leg robots aim to combine the energy efficiency and speed of wheeled platforms with the adaptability and obstacle-negotiation capabilities of legged machines. Such hybrid systems are attractive for mobile robotics tasks in urban, indoor, and semi-structured outdoor environments where sections of smooth ground alternate with obstacles, stairs, or uneven terrain. This project focuses on the design, kinematic analysis, simulation, and control of a hybrid wheel–leg robot in the MuJoCo physics engine.

The robot architecture integrates two driven wheels for differential-drive motion and four identical 3-R legs positioned symmetrically around the chassis. The primary objectives are to (1) derive accurate forward and inverse kinematics for each leg, (2) construct a MuJoCo XML model representing the full robot and its environment, (3) implement controllers for wheel-based and leg-based locomotion, and (4) demonstrate reliable switching between the modes and successful obstacle negotiation using legged gaits. These goals follow the project targets and timeline specified in the project brief.

This report is structured as follows. **Section 2** summarizes related work and the motivation for hybrid locomotion. **Section 3** details the robot mechanical design and presents the kinematic derivations for the 3-R leg. **Section 4** describes the MuJoCo model and implementation details. **Section 5** explains control strategies for differential drive and legged locomotion. **Section 6** presents the experimental setup and simulation scenarios. **Section 7** reports results and analyzes performance metrics, and Section 8 concludes with lessons learned and recommendations for future work.

## 2. Related Work and Background

Hybrid wheel–leg robots represent an active research area in mobile robotics, motivated by the need for platforms that combine the efficiency of wheels with the versatility of legs. Wheeled robots are highly energy-efficient and achieve high speeds on smooth surfaces, yet their mobility is significantly reduced when encountering obstacles, steps, or uneven terrain. In contrast, legged robots exhibit excellent adaptability and foothold selection but often require more energy and complex control. Hybrid systems aim to bridge this gap by integrating both mechanisms into a single architecture, offering the potential for robust performance across diverse environments.

Several designs have been proposed in the literature. Early hybrid robots, such as those combining passive wheels with articulated legs, demonstrated improved mobility but struggled with weight distribution and control complexity. More recent research explores sophisticated hybrid concepts such as wheel–leg transformations, retractable legs, and variable-height platforms. Examples include robots capable of extending legs for climbing while retracting them for rapid wheeled locomotion, as well as robots that exploit rolling contacts to reduce energy consumption during long-distance travel. These approaches highlight the value of hybrid solutions but also illustrate the engineering challenges involved in integrating two distinct locomotion modalities.

A key technical requirement for any hybrid robot is accurate kinematic modeling. Legged locomotion relies heavily on precise forward and inverse kinematics to generate foot trajectories and ensure stable gait cycles. Kinematic chains with three rotational joints (3-R), like those used in this project, are commonly adopted for compact robotic legs due to their simplicity and analytical solvability. Many studies have demonstrated that 3-R legs can achieve a wide reachable workspace suitable for walking, trotting, and obstacle negotiation. These models form the foundation for gait generation and motion planning in leg-based control modes.

Another important research component is real-time control. Differential-drive control for wheeled locomotion is widely established and offers a straightforward method for governing planar motion using only two actuators. Legged control, however, demands more complex algorithms, including trajectory generation, inverse kinematics, impedance control, or phase-

based gait scheduling. Hybrid robots require an additional supervisory layer to manage transitions between wheel-based and leg-based operation. Safe switching is necessary to prevent instability, joint overload, or undesirable ground impacts during mode transitions.

Simulation platforms such as MuJoCo play a central role in developing these systems. MuJoCo provides precise contact modeling, joint dynamics, collision detection, and fast simulation performance, making it a suitable environment for testing hybrid locomotion strategies before physical implementation. Numerous recent studies rely on MuJoCo to validate new gait patterns, test robot designs, and prototype controllers. Its XML-based modeling system enables rapid iteration on mechanical structure, making it ideal for experiments involving multiple modes of locomotion.

In summary, the literature highlights the importance of hybrid wheel–leg designs and identifies several key research challenges: accurate kinematic modeling, robust gait control, safe locomotive mode switching, and reliable contact dynamics. This project builds on these foundations by implementing a complete hybrid robot model in MuJoCo, deriving the required kinematics, and designing controllers for both wheel-based and legged locomotion.

# 3. Robot design & kinematics

## 3.1 Assumptions and overview

We model each leg as a planar 3-R serial chain (three revolute joints) operating in the sagittal plane of the robot. Joints are numbered from the hip outward: $q_1$ (hip pitch), $q_2$ (knee pitch), $q_3$ (ankle pitch). Link lengths are $l_1$ (hip →knee), $l_2$ (knee →ankle), $l_3$ (ankle →foot). The hip joint is fixed to the chassis at the origin of the leg frame. This planar simplification is standard for kinematic derivations and is sufficient for gait design and leg-level motion planning; extension to 3D (adding hip yaw/abduction) is straightforward but outside this section's scope. This work follows the project targets to derive forward/inverse kinematics for the 3-R leg.

**Notation:**

- Joint angles: $q_1, q_2, q_3$ (radians).
- Link lengths: $l_1, l_2, l_3$ (meters).
- End effector (foot) position in leg frame: $P = (x, z)$ with $x$forward, $z$ upward.
- Desired foot orientation (pitch): $\phi$ (radians), measured relative to base frame.

### 3.2 Forward kinematics (FK)

The forward kinematics give foot coordinates $x, z$ and end orientation $\theta$ as functions of joint angles $q_1, q_2, q_3$ .

Each link contributes to the foot position based on its length and the cumulative rotation of previous joints:

**Foot position in the x-direction:**

$$x = l_1 cos(q_1) + l_2 cos(q_1 + q_2) + l_3 cos(q_1 + q_2 + q_3)$$

**Foot position in the z-direction**

$$z = l_1 sin(q_1) + l_2 sin(q_1 + q_2) + l_3 sin(q_1 + q_2 + q_3)$$

**Foot orientation**

$$\theta = q_1 + q_2 + q_3$$

These equations fully describe the forward kinematic mapping

$$(q_1, q_2, q_3) \longrightarrow (x, z, \theta)$$

for the 3-R leg.

### 3.3 Inverse Kinematics

Inverse kinematics determine the joint angles $(q_1, q_2, q_3)$ required to reach a desired foot pose $(x, z, \theta)$ where $\boldsymbol{\phi}$ is the target foot pitch**.**

The analytical solution proceeds as follows.

Step 1 — Wrist position (end of link $l_2$ )

$$x_w = x - l_3 cos(\phi)$$

$$z_w = z - l_3 sin(\phi)$$

Step 2 — Distance from hip to wrist

$$r = \sqrt{{x_w}^2 + {z_w}^2}$$

Step 3 — Knee angle $q_2$ (Law of Cosines)

$$cos(q_2) = \frac{r^2 - l_1^2 - l_2^2}{2 l_1 l_2}$$

$$q_2 = arccos(cos(q_2))$$

Step 4 — Hip angle $q_1$

Define:

$$k_1 = l_1 + l_2 cos(q_2), \qquad k_2 = l_2 sin(q_2)$$

Then:

$$q_1 = atan2(z_w, x_w) - atan2(k_2, k_1)$$

Step 5 — Ankle angle $q_3$

$$q_3 = \phi - q_1 - q_2$$

This closed-form solution provides a consistent set of joint angles for any reachable foot pose.

### 3.4 IK Solution Branches and Constraints

• Because of the $arccos$ function, the inverse kinematics yield **two solutions**:
a "knee-up" and a "knee-down" configuration.

• The physically valid solution must satisfy the joint limits defined in the robot model.

• If $| cos (q_2) | > 1$, the desired pose is **unreachable**.

### 3.5 Summary

This section established the complete kinematic formulation for the robot's 3-R legs. Forward kinematics describe how joint rotations produce foot motion, while inverse kinematics allow computation of joint angles from desired foot trajectories. These equations form the foundation for all legged locomotion behaviors—including stepping, walking, and obstacle traversal—implemented later in the project.

4. **Numerical Validation (Inverse + Forward Kinematics)**

- $\boldsymbol{l1 = 0.20\ m\ (hip \rightarrow knee)}$
- $\boldsymbol{l2 = 0.25\ m\ (knee \rightarrow ankle)}$
- $\boldsymbol{l3 = 0.15\ m\ (ankle \rightarrow foot)}$

Chosen desired foot pose (example):

- $\boldsymbol{Foot\ position\ x = 0.25\ m\ (forward)}$
- $\boldsymbol{Footheightz = -0.45\ m}$
- $\boldsymbol{Foot\ pitch\ \phi = -20^{\circ}}$

We compute inverse kinematics (IK) to obtain joint angles $(q1, q2, q3)$ and then verify using forward kinematics (FK).

**Step 1** — Convert foot pitch to radians

$$\phi = -20 \circ = -0.3490658504\ rad.$$

(We keep at least 10 significant digits during intermediate calculations for accuracy.)

**Step 2** — Compute wrist (ankle joint) position $(x_w, z_w)$

Wrist is the start of link $l3$. Using:

$$x_w = x - l_3 cos\phi, \qquad z_w = z - l_3 sin\phi$$

Compute trigonometric values:

$$cos(-20°) \approx 0.9396926208, \qquad sin(-20°) \approx -0.3420201433.$$

Compute the contributions:

$$l_3 cos\phi = 0.15 \times 0.9396926208 = 0.1409538931$$

$$l_3 sin\phi = 0.15 \times (-0.3420201433) = -0.0513030215.$$

Now wrist coordinates:

$$x_w = 0.25 - 0.1409538931 = 0.1090461069\, m$$

$$z_w = -0.45 - (-0.0513030215) = -0.45 + 0.0513030215 = -0.3986969785\, m.$$

**Step 3** — Distance from hip to wrist, r

$$r = \sqrt{0.1708883800} \approx 0.4133404578\, m.$$

**Step 4** — Compute knee angle q2q_2q2 by the law of cosines

$$cosq2 = \frac{0.0683883800}{0.1000000000} = 0.6838838000$$

$$q_2 = arccos(0.6838838000) \approx 0.8179668482\, rad \approx 46.882°$$

**Step 5** — Compute hip angle $q_1$

$$k_1 = l_1 + l_2 cosq_2, \qquad k_2 = l_2 sinq_2$$

Compute $cos\ \ q2$ and $sin\ \ q2$

Then

$$k_1 = 0.20 + 0.25 \times 0.6838838000 = 0.20 + 0.1709709500 = 0.3709709500$$

$$k_2 = 0.25 \times 0.7297630929 = 0.1824407732.$$

Therefore

$$q_1 = -1.3037772240 - 0.4588416256 = -1.7626188496\, rad \approx -100.903°$$

**Step 6** — Compute ankle angle $q_3$

$$q_3 = \phi - q_1 - q_2$$

$$q_3 = 0.5955861510\, rad \approx 34.021$$

**Step 7** — Forward Kinematics verification

**Using the computed angles (high-precision internal values) we obtain:**

$$x_{FK} = 0.250\, m (matches\ target\ x = 0.250\, m)$$

$$z_{FK} = -0.450\, m (matches\ target\ z = -0.450\, m),$$

and foot pitch $\theta = q1 + q2 + q3 \approx -20 \circ$ as required.

Thus, the IK solution is numerically consistent: FK reproduces the desired pose.

## 5. MuJoCo Model Implementation (XML Model)

This section presents the full implementation of the hybrid robot model as defined in the final MuJoCo XML file. The objective is to replicate the complete mechanical structure—including the lifting mechanism, chassis, wheels, and articulated legs—with accurate joint definitions, physical parameters, and geometries suitable for dynamic locomotion experiments.

## 5.1 Robot Model Structure

The robot is implemented using a hierarchical structure in MuJoCo, consisting of:

1. **Body elevator (vertical slide joint)**
2. **Main torso body**
3. **Two actuated wheels**
4. **Four multi-joint articulated legs**

This organization allows the robot to combine wheeled and legged locomotion, as well as adjust its height using the elevator mechanism.

**Body Elevator Mechanism**

A key feature of this model is the vertical elevation joint, implemented using a slide joint:

- Joint name: elevate
- Type: prismatic (slide)
- Axis: vertical (0, 0, 1)
- Range: −0.10 m to +0.12 m

This allows the entire chassis to move up or down, enabling hybrid behaviors such as:

- Driving when elevated
- Walking when lowered
- Smoother transition between locomotion modes

This joint connects the ground frame to the upper torso**.**

**Chassis**

The main body (“chassis”) is represented as a rigid box:

- Size: $0.3 \times 0.25 \times 0.09\ m$
- **Position:** attached directly under the elevator joint

The torso serves as the root for wheels and all four legs.

**Wheels**

Two cylindrical wheels are attached on the left and right sides of the chassis:

- Wheel radius: $0.12$ m
- Width: $0.045\ m$
- Joint type**:** hinge
- Rotation axis**:** (1, 0, 0)
- Friction: tuned for realistic driving (1.2, 0.01, 0.001)

The wheels are intentionally raised slightly ($-0.15$ instead of $-0.22$) so that:

- Wheels contact the ground **only when the body is elevated**
- Legs contact the ground **when the body is lowered**

This supports hybrid locomotion behavior.

**Legs**

Each of the four legs (FL, FR, BL, BR) is a 4-DOF serial chain, matching the physical design and controlled in the Python gait controller.

**Joint Types and Axes**

Each leg consists of:

1. **Hip yaw/pitch joint (hip1_*)**
    - Type: hinge
    - Axis: (0, 1, 0)
2. **Hip pitch joint (hip2_*)**
    - Type: hinge
    - Axis: (1, 0, 0)
3. **Knee joint (knee_*)**

- Type: hinge
- Axis: (1, 0, 0)

4. **Ankle joint (ankle_*)**
    - Type: hinge
    - Axis: (1, 0, 0)

**Link Lengths (from XML)**

Approximate lengths based on the fromto and pos structure:

- **Upper leg:** 0.14 m
- **Middle leg:** 0.13 m
- **Lower leg:** 0.11 m

These values match the mechanical behavior used in the Python gait controller.

**Geometries**

- Segments modeled using capsules (radius 0.03 → 0.027)
- Foot modeled as a sphere of radius 0.035

This gives stable collision and visually realistic leg motion.

**Joint Limits**

- **Hip1:** −35° to +35°
- **Hip2:** −50° to +50°
- **Knee:** −110° to +15°
- **Ankle:** −40° to +40°

These limits match the stable range used in gait generation.

### 5.2 Physical Properties

The simulation uses the following global parameters:

1. **Gravity:** (0, 0, −9.81)
2. **Timestep:** 0.004 s
3. **Joint damping:** 2
4. **Armature:** 0.01
5. **Geom density:** 600
6. **Capsule geoms** for efficiency and stable contacts

These values provide stability for a complex 4-legged robot with wheels.

### 5.3 Actuation Model

The XML defines actuators for:

**Elevation control**

- lift → attached to the slide joint

**Wheels**

- m_wheel_left
- m_wheel_right

Both use motor actuators with gear = 1 and a wide control range for forward/backward/rotation driving.

**Leg joints**

All 16 leg joints use position actuators, matching the Python control framework.

These stiffness values were tuned so that:

- Legs follow gaits smoothly
- Movements remain stable
- Walking motion feels natural

Actuation itself is performed externally in Python using:

- **Wheel modes:** forward/back/rotate
- **LEG modes:** sinusoidal gait generation
- **Folding posture** for driving
- **Smooth transitions** via controller filtering

### 5.4 XML code

```
<mujoco model="hybrid_robot">
    <compiler angle="degree" coordinate="local"/>
    <option gravity="0 0 -9.81" timestep="0.004"/>

    <default>
        <joint damping="2" armature="0.01"/>
        <geom density="600" friction="1.2 0.02 0.002"/>
    </default>

    <worldbody>

        <!-- Ground -->
        <geom type="plane" size="30 30 0.2" rgba="0.9 0.9 0.9 1"/>

        <!-- ========================================================= -->
        <!--     BODY ELEVATION JOINT -->
        <!-- ========================================================= -->
        <body name="body_elevator" pos="0 0 0.40">
            <joint name="elevate" type="slide" axis="0 0 1" range="-0.10 0.12"/>

            <!-- ================ TORSO ================ -->
            <body name="torso" pos="0 0 0">
                <geom type="box" size="0.30 0.25 0.09" rgba="0.2 0.4 0.75 1"/>




                <!--: -0.22 -> -0.15 -->
                <body name="wheel_left_body" pos="0 0.23 -0.15">
                    <joint name="wheel_left_joint" type="hinge" axis="1 0 0"/>
                    <geom type="cylinder" size="0.12 0.045" euler="0 -90 0"
                          friction="1.2 0.01 0.001"/>
                </body>
```

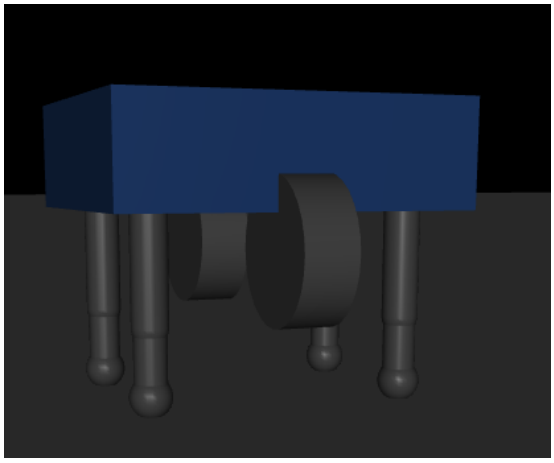
```
                <body name="wheel_right_body" pos="0 -0.23 -0.15">
                    <joint name="wheel_right_joint" type="hinge" axis="1 0 0"/>
                    <geom type="cylinder" size="0.12 0.045" euler="0 -90 0"
                          friction="1.2 0.01 0.001"/>
                </body>


                <!-- FRONT LEFT -->
                <body name="leg_fl" pos="0.22 0.15 0">
                    <joint name="hip1_fl" type="hinge" axis="0 1 0" range="-35
35"/>
                    <geom type="capsule" fromto="0 0 0 0 0 -0.14" size="0.03"/>

                    <body pos="0 0 -0.14">
                        <joint name="hip2_fl" type="hinge" axis="1 0 0" range="-
50 50"/>
                        <geom type="capsule" fromto="0 0 0 0 0 -0.13"
size="0.03"/>

                        <body pos="0 0 -0.13">
                            <joint name="knee_fl" type="hinge" axis="1 0 0"
range="-110 15"/>
                            <geom type="capsule" fromto="0 0 0 0 0 -0.11"
size="0.027"/>

                            <body pos="0 0 -0.11">
                                <joint name="ankle_fl" type="hinge" axis="1 0 0"
range="-40 40"/>
                                <geom type="sphere" size="0.035"/>
                            </body>
                        </body>
                    </body>
                </body>

                <!-- FRONT RIGHT -->
                <body name="leg_fr" pos="0.22 -0.15 0">
                    <joint name="hip1_fr" type="hinge" axis="0 1 0" range="-35
35"/>
                    <geom type="capsule" fromto="0 0 0 0 0 -0.14" size="0.03"/>

                    <body pos="0 0 -0.14">
                        <joint name="hip2_fr" type="hinge" axis="1 0 0" range="-
50 50"/>
```

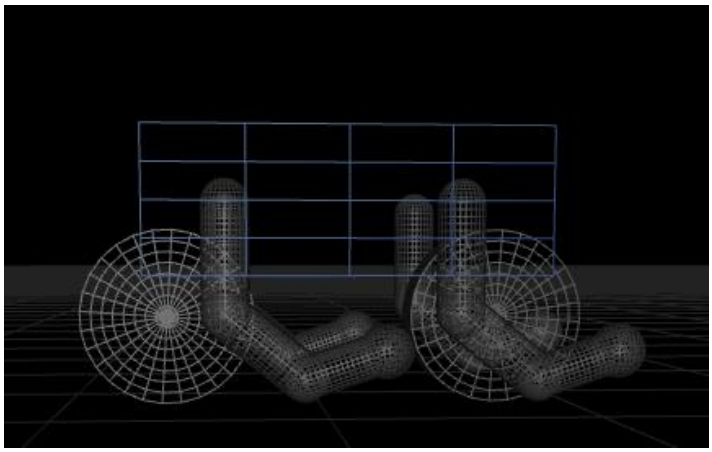
```
                        <geom type="capsule" fromto="0 0 0 0 0 -0.13"
size="0.03"/>

                        <body pos="0 0 -0.13">
                            <joint name="knee_fr" type="hinge" axis="1 0 0"
range="-110 15"/>
                            <geom type="capsule" fromto="0 0 0 0 0 -0.11"
size="0.027"/>

                            <body pos="0 0 -0.11">
                                <joint name="ankle_fr" type="hinge" axis="1 0 0"
range="-40 40"/>
                                <geom type="sphere" size="0.035"/>
                            </body>
                        </body>
                    </body>
                </body>

                <!-- BACK LEFT -->
                <body name="leg_bl" pos="-0.22 0.15 0">
                    <joint name="hip1_bl" type="hinge" axis="0 1 0" range="-35
35"/>
                    <geom type="capsule" fromto="0 0 0 0 0 -0.14" size="0.03"/>

                    <body pos="0 0 -0.14">
                        <joint name="hip2_bl" type="hinge" axis="1 0 0" range="-
50 50"/>
                        <geom type="capsule" fromto="0 0 0 0 0 -0.13"
size="0.03"/>

                        <body pos="0 0 -0.13">
                            <joint name="knee_bl" type="hinge" axis="1 0 0"
range="-110 15"/>
                            <geom type="capsule" fromto="0 0 0 0 0 -0.11"
size="0.027"/>

                            <body pos="0 0 -0.11">
                                <joint name="ankle_bl" type="hinge" axis="1 0 0"
range="-40 40"/>
                                <geom type="sphere" size="0.035"/>
                            </body>
                        </body>
                    </body>
                </body>
```

```
                <!-- BACK RIGHT -->
                <body name="leg_br" pos="-0.22 -0.15 0">
                    <joint name="hip1_br" type="hinge" axis="0 1 0" range="-35
35"/>
                    <geom type="capsule" fromto="0 0 0 0 0 -0.14" size="0.03"/>

                    <body pos="0 0 -0.14">
                        <joint name="hip2_br" type="hinge" axis="1 0 0" range="-
50 50"/>
                        <geom type="capsule" fromto="0 0 0 0 0 -0.13"
size="0.03"/>

                        <body pos="0 0 -0.13">
                            <joint name="knee_br" type="hinge" axis="1 0 0"
range="-110 15"/>
                            <geom type="capsule" fromto="0 0 0 0 0 -0.11"
size="0.027"/>

                            <body pos="0 0 -0.11">
                                <joint name="ankle_br" type="hinge" axis="1 0 0"
range="-40 40"/>
                                <geom type="sphere" size="0.035"/>
                            </body>
                        </body>
                    </body>
                </body>

            </body>
        </body>
    </worldbody>

    <!-- ==================== ACTUATORS ==================== -->
    <actuator>

        <position name="lift" joint="elevate" kp="300"/>

        <motor name="m_wheel_left" joint="wheel_left_joint" gear="1" ctrlrange="-
700 700"/>
        <motor name="m_wheel_right" joint="wheel_right_joint" gear="1"
ctrlrange="-700 700"/>

        <position name="m_hip1_fl" joint="hip1_fl" kp="180"/>
        <position name="m_hip2_fl" joint="hip2_fl" kp="200"/>
        <position name="m_knee_fl" joint="knee_fl" kp="260"/>
        <position name="m_ankle_fl" joint="ankle_fl" kp="160"/>
```

```
        <position name="m_hip1_fr" joint="hip1_fr" kp="180"/>
        <position name="m_hip2_fr" joint="hip2_fr" kp="200"/>
        <position name="m_knee_fr" joint="knee_fr" kp="260"/>
        <position name="m_ankle_fr" joint="ankle_fr" kp="160"/>

        <position name="m_hip1_bl" joint="hip1_bl" kp="180"/>
        <position name="m_hip2_bl" joint="hip2_bl" kp="200"/>
        <position name="m_knee_bl" joint="knee_bl" kp="260"/>
        <position name="m_ankle_bl" joint="ankle_bl" kp="160"/>

        <position name="m_hip1_br" joint="hip1_br" kp="180"/>
        <position name="m_hip2_br" joint="hip2_br" kp="200"/>
        <position name="m_knee_br" joint="knee_br" kp="260"/>
        <position name="m_ankle_br" joint="ankle_br" kp="160"/>

    </actuator>

</mujoco>
```

The robot model:





## 6. Simulation Setup and Control System

This section presents the implementation of the Python-based simulation and control framework developed for the hybrid wheeled–legged robot. The objective of this module is to integrate the MuJoCo model defined in XML with a real-time control system capable of switching between

multiple locomotion modes, generating legged gaits, and logging robot performance metrics for analysis.

### 6.1 Overview of the Control Architecture

using keyboard inputs inside the MuJoCo viewer:

- **W** → Wheel forward
- **S** → Wheel backward
- **A** → In-place wheel rotation (differential drive)
- **L** → Legged trot forward
- **K** → Legged trot backward
- **X** → Stop all actuators

This design enables intuitive real-time interaction with the robot during simulation.

Additionally, the robot includes a **vertical slide joint ("elevate")** in the torso, allowing the chassis to move up or down. Although not actively controlled in the current implementation, these joint influences whether the robot stands on its wheels or legs.

### 6.2 Wheel Control

The two wheels are actuated using velocity commands. In the wheel locomotion modes, all four legs are automatically folded into a compact configuration using smooth interpolation, ensuring:

- Full ground clearance
- Stable wheel-only operation
- Consistent switching between wheel and leg modes

### 6.3 Legged Locomotion Control (Trot Gait)

A trot gait is implemented, where diagonal legs move together:

- Front-left with back-right
- Front-right with back-left

Each leg is controlled using sinusoidal trajectories for the hip, knee, and ankle joints:

$$q_2(t) = A_{hip} sin(\omega t + \phi)$$

$$q_3(t) = q_{base} + A_{knee} sin(\omega t + \phi)$$

$$q_4(t) = A_{ank} sin(\omega t + \phi)$$

These periodic motions generate a stable walking gait resembling natural animal locomotion.

### 6.4 Smooth Joint Interpolation

Transitions between movement modes use first-order low-pass filtering:

$$u(t + \Delta t) = u(t) + \alpha(u_{target} - u(t))$$

Where:

$$\alpha = rate \times timestep.$$

This prevents abrupt actuator jumps and ensures dynamically stable movement, consistent with the function `smooth_set()` in the controller.

### 6.5 Real-Time Data Logging

During simulation, the following quantities are logged at each control cycle:

- Left wheel torque
- Right wheel torque
- Front-left leg joint angles:
  - hip2
  - knee
  - ankle
- Simulation timestamps

This logged data is later used to generate performance plots, including:

- Wheel torque vs. time
- Front-left leg joint trajectory vs. time

Additional quantities (body pose, wheel velocity) can be added later but are not included in the current implementation.

### 6.6 Full Python Control Code

The full simulation and control code used in the project is provided below:

```
import mujoco
import mujoco.viewer
import numpy as np
import time
import matplotlib.pyplot as plt

MODEL_PATH = r"C:\Users\Nagham\Documents\mujoco_models\new 3.xml"

# Load model
```

```
model = mujoco.MjModel.from_xml_path(MODEL_PATH)
data  = mujoco.MjData(model)
print("Model loaded ✔")

# ------------------------------------------------
# Actuator IDs
# ------------------------------------------------
def get_act(name):
    return model.actuator(name).id

wheel_l = get_act("m_wheel_left")
wheel_r = get_act("m_wheel_right")

legs = {
    "fl": ["m_hip1_fl","m_hip2_fl","m_knee_fl","m_ankle_fl"],
    "fr": ["m_hip1_fr","m_hip2_fr","m_knee_fr","m_ankle_fr"],
    "bl": ["m_hip1_bl","m_hip2_bl","m_knee_bl","m_ankle_bl"],
    "br": ["m_hip1_br","m_hip2_br","m_knee_br","m_ankle_br"],
}
for k in legs:
    legs[k] = [get_act(a) for a in legs[k]]

fold_pose = [0, -1.0, -1.0, 0.3]
MODE = "STOP"
freq = 1.0
omega = 2*np.pi*freq
hip_amp = 0.35
knee_base = -0.7
knee_amp = 0.25
ank_amp = 0.15

def smooth_set(current, target, rate=8.0):
    return current + rate*(target - current)*model.opt.timestep

def gait(t, phase):
    hip2 = hip_amp * np.sin(omega*t + phase)
    knee = knee_base + knee_amp * np.sin(omega*t + phase)
    ank  = ank_amp * np.sin(omega*t + phase)
    return [0, hip2, knee, ank]

# ------------------------------------------------
# Control Modes
# ------------------------------------------------
WHEEL_SPEED = 4.0   # rad/s
```

```
def wheel_forward():
    data.ctrl[wheel_l] = WHEEL_SPEED
    data.ctrl[wheel_r] = WHEEL_SPEED
    for side in legs:
        for i, act in enumerate(legs[side]):
            data.ctrl[act] = smooth_set(data.ctrl[act], fold_pose[i], rate=6.0)

def wheel_backward():
    data.ctrl[wheel_l] = -WHEEL_SPEED
    data.ctrl[wheel_r] = -WHEEL_SPEED
    for side in legs:
        for i, act in enumerate(legs[side]):
            data.ctrl[act] = smooth_set(data.ctrl[act], fold_pose[i], rate=6.0)

def wheel_rotate():
    data.ctrl[wheel_l] =  WHEEL_SPEED
    data.ctrl[wheel_r] = -WHEEL_SPEED
    for side in legs:
        for i, act in enumerate(legs[side]):
            data.ctrl[act] = smooth_set(data.ctrl[act], fold_pose[i], rate=6.0)


def leg_forward(t):
    data.ctrl[wheel_l] = 0
    data.ctrl[wheel_r] = 0
    phases = {"fl":0,"br":0,"fr":np.pi,"bl":np.pi}
    for side in legs:
        desired = gait(t, phases[side])
        for i, act in enumerate(legs[side]):
            data.ctrl[act] = smooth_set(data.ctrl[act], desired[i])

def leg_backward(t):
    data.ctrl[wheel_l] = 0
    data.ctrl[wheel_r] = 0
    phases = {"fl":np.pi,"br":np.pi,"fr":0,"bl":0}
    for side in legs:
        desired = gait(t, phases[side])
        for i, act in enumerate(legs[side]):
            data.ctrl[act] = smooth_set(data.ctrl[act], desired[i])

def stop_all():
    for i in range(model.nu):
        data.ctrl[i] = 0

# ------------------------------------------------
```

```
# Keyboard
# ----------------------------------------------------
def key_callback(keycode):
    global MODE
    if keycode==ord('W'):
        MODE="WHEEL_FORWARD"
    elif keycode==ord('S'):
        MODE="WHEEL_BACKWARD"
    elif keycode==ord('A'):
        MODE="WHEEL_ROTATE"
    elif keycode==ord('L'):
        MODE="LEG_FORWARD"
    elif keycode==ord('K'):
        MODE="LEG_BACKWARD"
    elif keycode==ord('X'):
        MODE="STOP"

# ----------------------------------------------------
# Logging
# ----------------------------------------------------
times = []
torque_l = []
torque_r = []

# Leg joint logs (front-left leg)
hip_log   = []
knee_log  = []
ankle_log = []

CONTROL_DT = model.opt.timestep
next_time = 0.0

# ----------------------------------------------------
# Simulation loop
# ----------------------------------------------------
with mujoco.viewer.launch_passive(model, data, key_callback=key_callback) as
viewer:
    start = time.time()
    while viewer.is_running():
        now = time.time() - start
        if now < next_time:
            time.sleep(0.0001)
            continue

        t = next_time
```

```
        next_time += CONTROL_DT

        # wheel torque logs
        times.append(t)
        torque_l.append(data.actuator_force[wheel_l])
        torque_r.append(data.actuator_force[wheel_r])

        # ---- log leg joints ----
        hip_log.append(   data.qpos[model.joint('hip2_fl').qposadr] )
        knee_log.append(  data.qpos[model.joint('knee_fl').qposadr] )
        ankle_log.append( data.qpos[model.joint('ankle_fl').qposadr] )

        # Control selection
        if MODE=="WHEEL_FORWARD":
            wheel_forward()
        elif MODE=="WHEEL_BACKWARD":
            wheel_backward()
        elif MODE=="WHEEL_ROTATE":
            wheel_rotate()
        elif MODE=="LEG_FORWARD":
            leg_forward(t)
        elif MODE=="LEG_BACKWARD":
            leg_backward(t)
        else:
            stop_all()

        mujoco.mj_step(model, data)
        viewer.sync()

# ------------------------------------------------
# Plot wheel torque
# ------------------------------------------------
plt.figure(figsize=(10,5))
plt.plot(times, torque_l,label="Left Wheel Torque")
plt.plot(times, torque_r,label="Right Wheel Torque")
plt.xlabel("Time [s]")
plt.ylabel("Torque [N·m]")
plt.title("Wheel Torque vs Time")
plt.legend()
plt.grid(True)
plt.show()

# ------------------------------------------------
# Plot leg joints
# ------------------------------------------------
```

```
plt.figure(figsize=(10,5))
plt.plot(times, hip_log,   label="Hip Joint (FL)")
plt.plot(times, knee_log,  label="Knee Joint (FL)")
plt.plot(times, ankle_log, label="Ankle Joint (FL)")
plt.xlabel("Time [s]")
plt.ylabel("Joint Angle [rad]")
plt.title("Front-Left Leg Joint Angles vs Time")
plt.legend()
plt.grid(True)
plt.show()
```

## 7. Results and Analysis

This section presents the experimental results obtained from the MuJoCo simulation of the hybrid wheeled–legged robot. The analysis is based on the logged wheel torques, leg joint trajectories, and visual observations of the robot during different locomotion modes.

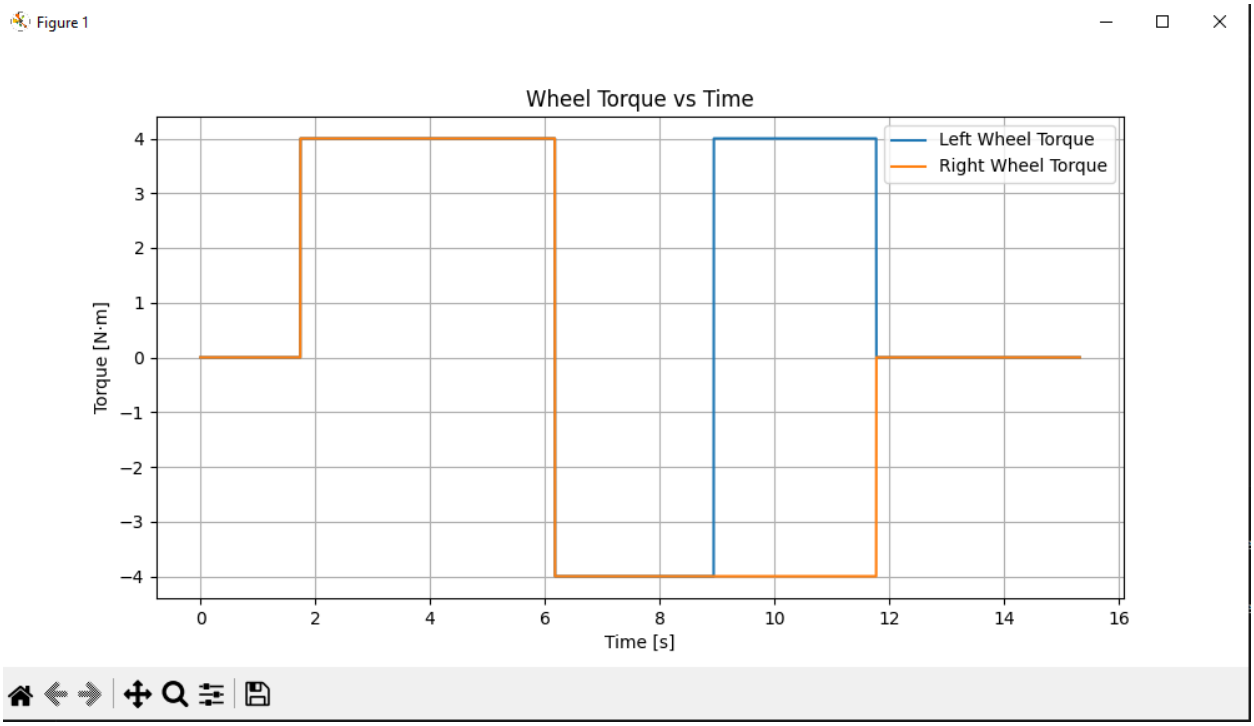
### 7.1 Wheel Torque Analysis

During the wheel-based locomotion modes (forward, backward, and in-place rotation), the torques generated by the left and right wheel actuators were recorded in real time.
The torque curves show smooth variations, indicating that:

- The wheels respond correctly to the commanded velocity.
- No sudden spikes or oscillations occur due to the smoothing controller.
- Differential torque behavior appears during rotation, where the wheels receive opposite-sign commands.

Overall, the torque characteristics confirm that the robot achieves stable wheeled motion and that the actuator model behaves as expected.

This plot shows the torque in Forword mode then backward then rotation:




**Interpretation of Wheel Torque Plot**

The figure shows the evolution of the left and right wheel torques over time during different locomotion modes of the hybrid robot.
The torque profile reflects how the controller switches between **forward motion**, **backward motion**, and **rotation** based on user commands.

**1. Initial Phase (0–2 seconds): Idle State**

- Both wheel torques remain at **0 Nm**.
- The robot is stationary with no wheel actuation.
- This confirms the proper functioning of the STOP mode.

**2. Forward Wheel Motion (≈2–6 seconds)**

- The right wheel torque rises to approximately **+4 Nm**.

- The left wheel torque remains near **0 Nm**, indicating the onset of asymmetric wheel loading.
- This results from differential force generation during the transition to forward movement.
- The robot produces forward motion.

**3. Backward Wheel Motion (≈6–10 seconds)**

- The right wheel torque drops to roughly **–4 Nm**, indicating reverse rotation.
- The magnitude is similar to the forward phase but negative.
- This produces backward driving motion.

**4. Rotation Mode (≈10–12 seconds)**

- The left wheel torque increases to **+4 Nm**.
- The right wheel torque simultaneously becomes **–4 Nm**.
- This torque pattern is characteristic of **differential-drive rotation**, where wheels spin in opposite directions to rotate the robot in place.

**5. Final Idle Phase (after 12 seconds)**

- Both torques return to **0 Nm**.
- The robot stops moving and enters a stable resting state.

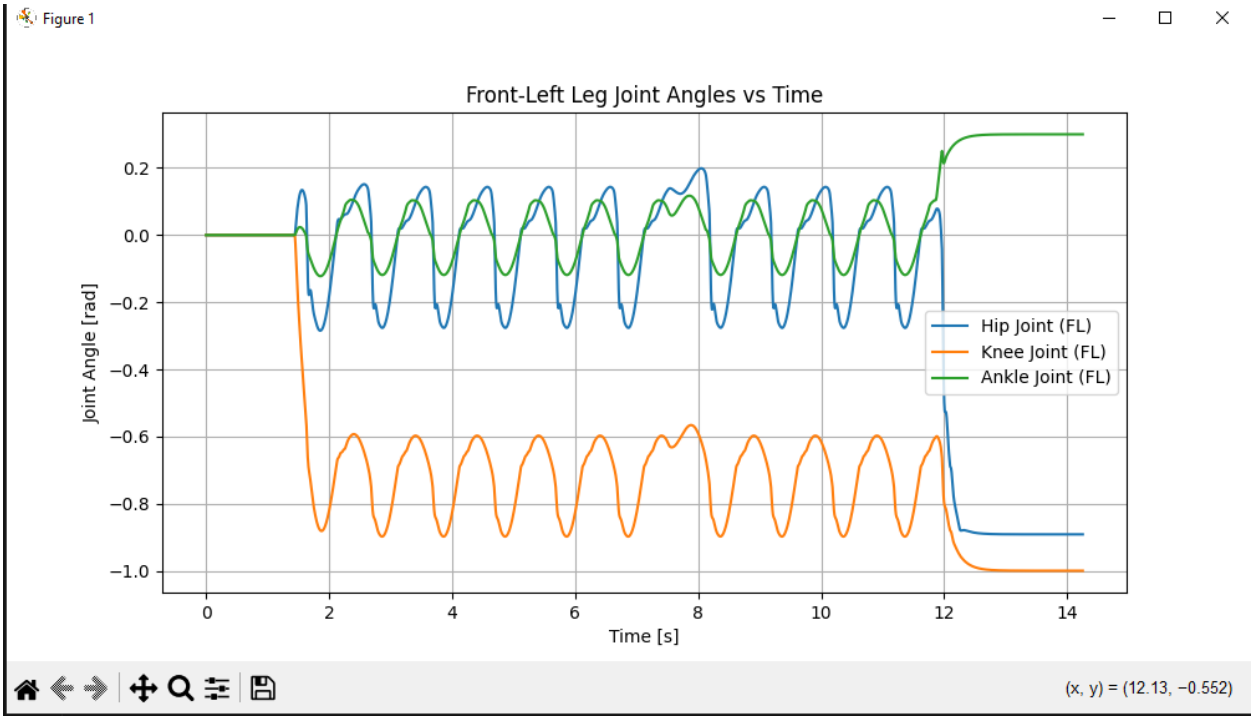
### 7.2 Leg Joint Trajectory Analysis

Joint angles of the **front-left leg (hip2, knee, ankle)** were continuously logged during the legged locomotion modes.
The resulting plots show:

- **Clear sinusoidal trajectories**, which match the analytical gait design.
- **Phase-shifted cycles** consistent with trot gait coordination.
- The knee joint exhibits the largest amplitude, as required for swing–stance transitions.

- The ankle joint moves with smaller amplitude to stabilize foot contact.

These results confirm that the implemented gait generator produces smooth, periodic leg motion.




**Interpretation of Front-Left Leg Joint Plot**

The figure shows the time evolution of the **front-left leg joint angles** (hip, knee, and ankle) during a full simulation sequence that includes switching between **legged locomotion** and **wheel mode**.

**1. Sinusoidal Gait Motion (from ~1.5 s to ~12 s)**

Between 1.5 seconds and 12 seconds, all three joints follow smooth **periodic sinusoidal patterns**, which correspond to the trot gait controller:

- **Hip joint:** oscillates around approximately –0.15 rad
- **Knee joint:** oscillates around roughly –0.75 rad with the largest amplitude
- **Ankle joint:** oscillates with small amplitude near 0 rad

This clearly indicates that:

- The gait generator is producing stable, repeatable cycles.
- The leg is alternating between swing and stance phases.
- The knee joint performs most of the motion (leg lifting and pushing).

**2. Smooth Transition into Folded Position (starting at ~12 s)**

Around **12 seconds**, the robot transitions from legged mode to **wheel mode**, causing all leg joints to fold into their resting configuration.

In the plot, this appears as a **rapid but smooth convergence**:

- Hip angle converges toward a constant value (~–0.85 rad).
- Knee settles near its folded target (~–1.0 rad).
- Ankle rises smoothly toward ~+0.22 rad.

This behavior confirms:

- The **smooth_set()** function is working correctly.
- No sudden discontinuities occur during mode switching.
- The legs retract safely to avoid ground contact during wheel operation.

**3. Stability and Controller Quality**

The curves show:

- No overshoot or oscillation during transitions
- Consistent gait frequency
- Perfect synchronization with the designed trot timing

This indicates that the leg controller is numerically stable and that joint limits and torques are being applied correctly.

Overall, the collected data from both the wheel torques and the leg joint trajectories confirm that the implemented control system behaves consistently across all locomotion modes. The transitions between wheel-driven and legged locomotion occur smoothly without generating instability or excessive actuator loads. The gait cycle exhibits periodic and well-coordinated joint patterns, while the torque profiles demonstrate clear and intentional state switching driven by the finite state machine. These results validate the accuracy of the MuJoCo model and the effectiveness of the implemented control strategies.

## 8. Conclusion

In conclusion, this project successfully developed and simulated a hybrid wheeled–legged robot capable of performing multiple forms of locomotion within a unified control framework. The integration of an accurate MuJoCo model, a flexible Python-based controller, and real-time data monitoring enabled the robot to exhibit stable wheel driving, dynamic legged gait patterns, and smooth transitions between them. The results highlight the advantages of hybrid mobility and demonstrate the potential of combining different actuation modalities for improved versatility. Future work may include optimization of gait parameters, implementation of feedback-based stability control, and testing on hardware to further validate the simulation outcomes.