Report
Team 9

Tirth Kamdar, Vamshikrishna Gadde

Project: Monopaw: Single leg canine gait robot

Walk-through of the notebook

The Jupyter notebook builds and analyzes a MuJoCo simulation of a single 5-bar leg attached to a cart-like rig. First, it installs and imports the required packages and sets up the MuJoCo rendering backend. Then it defines the main gait parameters (joint angle ranges, oscillation frequency, phase offset) and a simplified DC motor model that converts desired joint angles into torques. The robot itself is described in an MJCF XML: a flat ground plane, a sliding rig with a 3 kg box mass, two low-friction support stands, a 5-bar leg with a high-friction spherical foot, a loop-closure constraint, and two actuated joints. The notebook then runs a warm-up to settle contacts, simulates the gait over time, and logs foot position, estimated forward velocity, and mechanical energy while rendering frames for a video. Finally, it computes simple gait metrics (stride length) and creates plots of foot height, forward velocity, and kinetic/potential energy versus time.

Assumptions and simplifications

This model makes several simplifying assumptions compared to a real robot. All motion is planar and bodies are perfectly rigid, with exactly known masses, inertias, and link lengths; there are no manufacturing tolerances, backlash, or structural flex. Actuation is modeled as an ideal DC motor with constant parameters and a simple proportional controller, ignoring motor saturation details, efficiency losses, and thermal effects. Contacts use a flat, perfectly known ground with idealized Coulomb friction, and the foot and stands are given extreme friction values (very high at the foot, zero at the stands). Sensing is assumed perfect, with no noise or delays, and only a single leg plus rig is simulated—there is no multi-leg coordination, body pitching, or 3D effects as would be present in a full quadruped.

Differences from real-world conditions

Because of these assumptions, several aspects of the simulation differ from realistic hardware. Most notably, the rig is modeled as a relatively heavy 3 kg box, which is not meant to match a specific prototype but is chosen to make contact behavior usable: with our very high foot friction and zero-friction stands, a light rig does not push the system into the ground strongly enough, so the contact solver tends to pin the foot to the floor and the stands may never properly share the load, effectively "locking" the leg. Increasing the rig mass forces better ground contact at the stands and prevents the foot from acting like it is glued in place, allowing realistic stepping and making it possible to study how phase offset and link length affect forward velocity and mechanical energy. In practice, a physical robot would have different mass distribution, more moderate friction, structural compliance, and additional losses, so the exact numerical values will differ, but the trends in velocity and energy across different parameter choices are still informative for gait and design optimization.

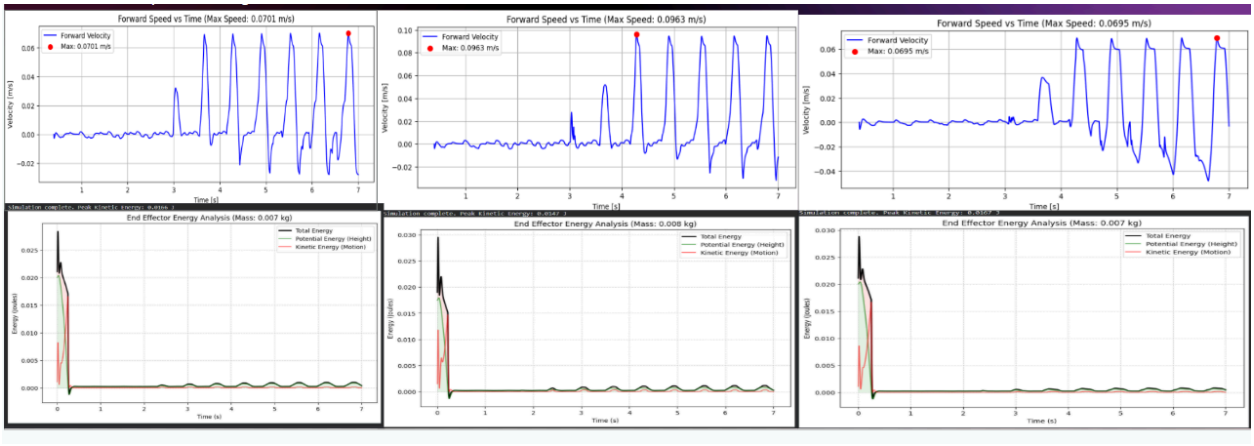
Design variables

In this project I chose two main design variables: the phase offset between the two actuated joints and the link lengths of the 5-bar leg (including a uniformly scaled-up version at +25%). The phase offset controls the *timing* between joint motions, which directly shapes the foot trajectory and how joint motion is split into "lift" (vertical) versus "push" (horizontal) during stance. By sweeping this parameter ($\pi/4$, $\pi/2$, $3\pi/4$, π) while keeping everything else fixed, I can see how different coordination patterns affect both the maximum forward velocity of the end-effector and the total mechanical energy in the system. Link length, on the other hand, sets the mechanical leverage and workspace of the leg: longer links change stride length and how effectively motor torques are converted into foot motion. Comparing the baseline geometry to a 25% scaled-up version at the same phase offset ($\pi/2$) shows how purely geometric changes influence speed and energy usage without changing the control law. Together, these two variables are physically meaningful (you really would choose leg lengths and gait phasing when designing a robot) and they have a clear, interpretable impact on the two outputs I care about: how fast the leg can move the foot forward, and how much energy it spends or wastes to do so.

Global parameter sweep and optimization

For this project I used a global parameter sweep rather than a fancy optimizer. I treated my design variables as: (1) phase offset between the two actuated joints, and (2) link length scale of the 5-bar leg. For the phase offset, I did a 1-D global sweep over a wide range of physically meaningful values: $\varphi \in \{\pi/4, \pi/2, 3\pi/4, \pi\}$. For each φ I ran a full MuJoCo simulation with the same initial conditions and motor parameters, then extracted two performance metrics: the maximum forward velocity of the end effector (from the foot's x-velocity trace) and the peak total kinetic energy (from MuJoCo's energy calculation). This gave me a direct mapping "phase offset → (max velocity, max KE)", and I could see that $\pi/2$ gave smoother, faster gaits while $\pi/4$ produced large energy spikes with poor forward motion. In a second sweep I compared geometry by running the same $\pi/2$ gait on the baseline leg and on a version with all link lengths scaled up by 25%, again comparing max foot velocity and peak KE; this showed that the scaled leg achieved the best velocity with the lowest energy. I chose this global sweep / brute-force approach because the system is highly nonlinear and contact-rich, so gradients are not obvious or reliable, and the design space (a few candidate phase offsets and one scale factor) is small enough that evaluating a discrete grid of cases is simple, robust, and easy to interpret physically.

Result aligning with what might be expected on the physical device or not.



In simulation I compared three main configurations: a baseline leg geometry with phase offset π/2, the same geometry with phase offset π/4, and a 25% link-length scaled geometry with phase offset π/2. Using the end-effector's forward velocity and MuJoCo's total kinetic energy as performance metrics, I found that the scaled π/2 case produced the highest forward speed and the lowest peak kinetic energy; the baseline π/2 case was in the middle for both; and the π/4 case had the lowest forward speed but the largest kinetic-energy spikes. This trend is physically sensible: a π/2 phase offset gives a "clean" separation between lifting and pushing, so more of the motion is converted into useful forward propulsion, whereas π/4 timing makes the leg thrash and bounce, wasting energy in vertical and internal motion without much net progress. Scaling the links up by 25% at π/2 increases stride length and the foot's horizontal travel for similar joint motions, so the leg moves faster without needing proportionally more internal motion, which shows up as higher velocity with lower total kinetic energy.

I expect these relative trends to largely carry over to a physical robot: better timing between joints (around π/2) and a sensible choice of link length should produce faster, smoother and more energy-efficient gait than poorly phased motion like π/4. However, the absolute values of speed and energy, and the exact ranking, will differ on hardware because the model makes several simplifications. The simulation assumes perfectly rigid links, ideal DC motors with a simple torque–speed model, exact parameters, and a very stylized contact model with a 3 kg rig mass, extremely high foot friction, and zero-friction stands. In reality, the rig

would likely be lighter and more compliant, friction would be moderate and variable, motor torque and efficiency would limit how much benefit you get from longer links, and impacts would be softened by flexibility and damping.

Leading sources of error between your model and prototype and their rectification

Planar model vs 3D, bent leg in hardware

The MuJoCo model is strictly planar and perfectly symmetric, so the leg can only move in the x–z plane. In the physical prototype, the leg is slightly bent sideways, which introduces a lateral (y-direction) component to the reaction forces. That causes the robot to drift to the right in the real world, while the simulation predicts purely straight-line motion. This 3D misalignment is probably the single biggest qualitative difference.

Geometry and mass distribution (including the 3 kg rig)

In sim, link lengths and masses are exact and the rig is modeled as a **3 kg** block chosen for numerical/ground-contact reasons, not because it matches your actual prototype. The real structure is likely lighter, with different mass distribution and center of mass. That changes impact behavior, how easily the rig accelerates, and the absolute values of velocity and kinetic energy.

Contact and friction modeling

The floor is a perfectly flat plane with idealized friction; the foot has very high friction and the stands have zero friction. In reality, the ground is not perfectly flat, friction depends on load and slip speed, and the foot/stands both have some finite, anisotropic friction. These differences strongly affect how the leg grips, slips, and how much it bounces.

Actuator and control simplifications

The DC motor model in MuJoCo uses a simple torque–speed relationship with a proportional angle controller and no sensor noise, no backlash, no voltage drop, and no current limits. The real motors and drivers will saturate, heat up, have dead

zones and backlash, and the real controller has sampling, delays, and noise. That changes timing and the true torque you get at the joints.

No structural compliance or play
The simulated leg is perfectly rigid with ideal joints; the real leg has flex, loose screws, and small joint clearances. This compliance can absorb impacts, reduce energy spikes, and slightly change the effective phase relationships and foot trajectory.

If we wanted to make the simulation track the prototype more closely, we could:

Model the 3D geometry including the sideways bend
Build a 3D MuJoCo model (no longer strictly planar) and tilt the leg or offset the joints to match the measured sideways bend. That would let the sim reproduce the rightward drift you see on hardware.

Calibrate geometry and masses from the real robot
Measure actual link lengths, thicknesses, and approximate masses, then update the MJCF to match. Replace the 3 kg "hack" mass with a more realistic rig mass and inertia that approximates your prototype.

Tune contact and friction from experiments
Do simple tests (e.g., drag the foot across the surface, drop the rig from a small height) to estimate normal restitution and friction coefficients, then tune the ground/foot/stand friction and contact parameters to reproduce those behaviors in sim.

Use a more realistic motor model and real control code
Identify your motor torque–speed curve and driver limits, and adjust the DC motor parameters accordingly. If possible, run a controller in sim that mirrors what runs

on your microcontroller (same gains, sampling time, and saturations).

Add compliance where it matters most
Introduce small joint springs/dampers or flexible elements in the MJCF (e.g., at the foot or between segments) to mimic the observed flex in the physical leg. This can help match impact behavior and energy dissipation.

V2 version

In the next version I would go from one leg to two legs on the same cart, using a mirrored copy of the existing 5-bar leg. The main goals would be: (1) fix the current sideways bend by redesigning or re-manufacturing the leg so the motion stays in a single plane, (2) mount a second leg on the opposite side at a known spacing, and (3) test simple coordination patterns between the two legs (in-phase vs out-of-phase) using the same DC-motor control structure I already have. In parallel, I'd update the MuJoCo model to match the new two-leg geometry and better-calibrated masses/friction so I can compare single-leg vs two-leg performance in terms of forward velocity and energy. This is a small but concrete step toward more realistic locomotion