

PPO-Based PID Gain Scheduling for Hexapod-Leg Control under Varying Gravity

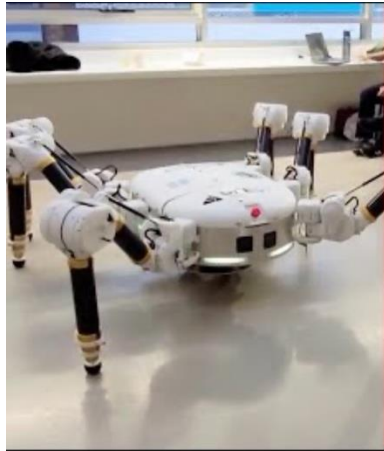
Challenges in AI-Based Embedded Systems Applications
CAIESA 2026 @ ESWEEK 2026

Barcelona, Spain
October 9, 2026

Ashton Mitchell, Aditya A. Krishnan, and Hokeun Kim
Arizona State University



Motivation



Forbes
**Meet the
spider-like
robot that
could build
structures on
the moon**



Motivating Question:
Can we send this robot
on an interstellar
mission to operate on
unexplored planets
with unknown gravity?

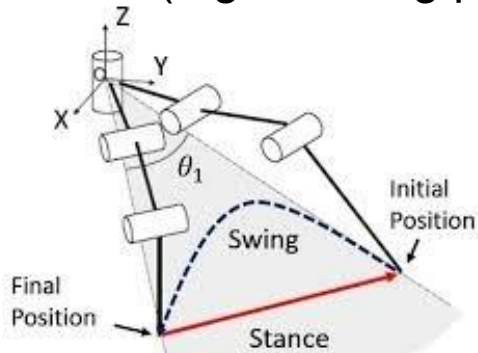
Charlotte, a spider-like autonomous
hexapod prototype robot built on
Earth to operate on the Moon

<https://www.designboom.com/technology/autonomous-robot-charlotte-3d-prints-homes-earth-space-without-cement-10-02-2025/>

Ashton Mitchell, Aditya A. Krishnan, and Hokeun Kim, Arizona State University

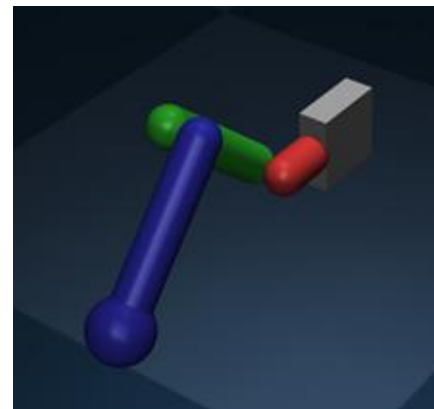
Challenges

- Hexapod locomotion assumes known magnitude and direction of gravity
 - Model dynamics
 - Control Parameters
 - Gait (e.g., walking pattern)
- However, **unknown gravity introduces many challenges:**
 - Incorrect torque scaling
 - Altered contact forces
 - Inconsistency in walking gait
 - Controller instability



Research Problem and Goals

- **Goal: To make control of a hexapod leg (with 3 DOF) adaptive to unknown gravity, and system model of such controller with determinism and repeatability:**
 - To maintain stable, repeatable tracking and locomotion of a hexapod leg without knowledge of the magnitude or direction of gravity
 - To enable adaptive, model-free control
 - To augment classic control with Reinforcement Learning (RL)
 - By using Proximal Policy Optimization (PPO) to learn an adaptive gain schedule for joint-specific PID controllers under unknown gravity



Contributions in the Context of CAIESA

- PPO-trained adaptive gain scheduler that retains the joint-level PID structure
- Evaluation against a fixed default-gain reference across various gravity environments and tracking tasks
- An open-source deterministic Lingua Franca (LF) executable model + MuJoCo physics simulation environment

Background – AI and Control

- PID control & gain scheduling for robotic legs
 - PID gains determine how strongly the controller reacts to tracking error and motion
 - Gain scheduling updates these PID gains
- Proximal Policy Optimization (PPO)
 - RL algorithm for an AI agent to learn decision-making strategies safely and stably without prior training

Thus, we use PPO-based gain scheduling to adapt to unknown gravity for motion control of robotic legs

Background – Modeling

- **Reactor model of computation (MoC)**

- Deterministic variant of the Hewitt/Agha Actor MoC
- With well-defined communication ports, hierarchical structure of reactors, and deterministic scheduling algorithms



REACTOR MODEL

<https://reactor-model.org/>

- **Lingua Franca (LF)**

- An open-source coordination language and runtime that supports specification, execution, and simulation of reactor-based programs



<https://www.lf-lang.org/>

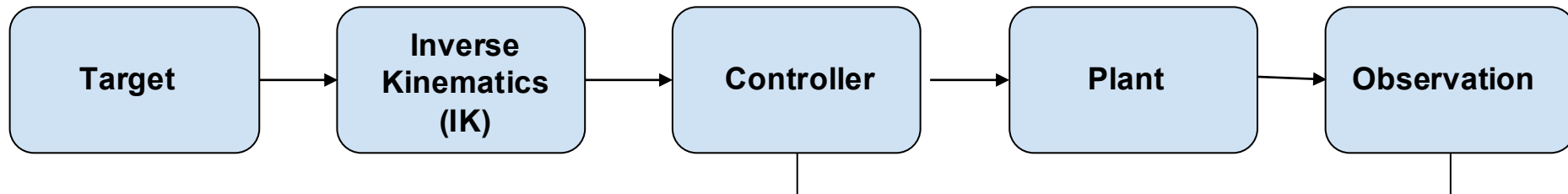
Background – Simulation of Physical Plant

- **MuJoCo (Multi-Joint dynamics with Contact)**
 - An open-source advanced physics engine and simulator specialized in multi-joint dynamics in robots



Our work leverages reactor MoC as the system model (not the control model) with business logic using PPO-based gain scheduling, together with advanced physics simulation of gravity-adaptive robot leg control

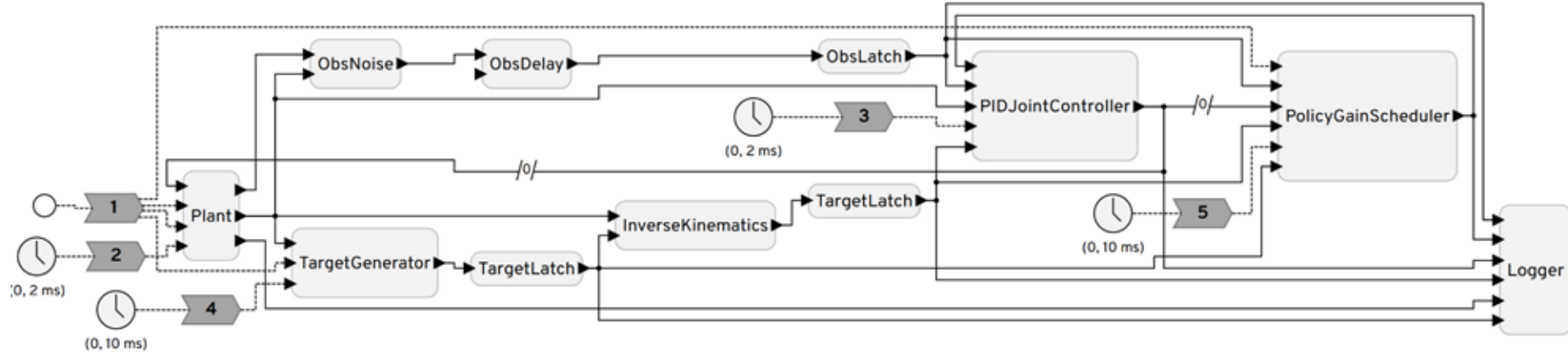
Overall Control Flow



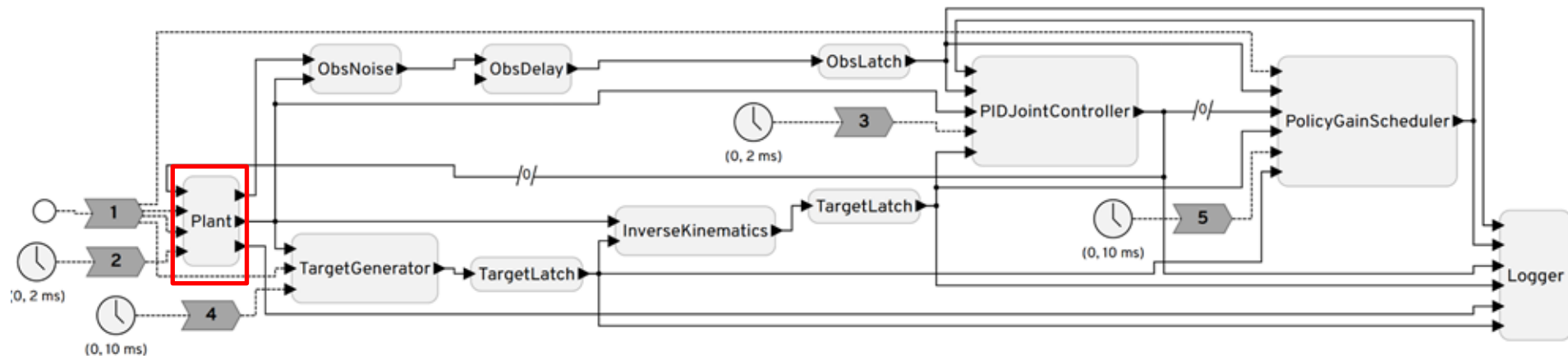
Target tasks

- Holding pose
- Following sinusoidal trajectory
- Point-to-point trajectory

Reactor Model Architecture Overview

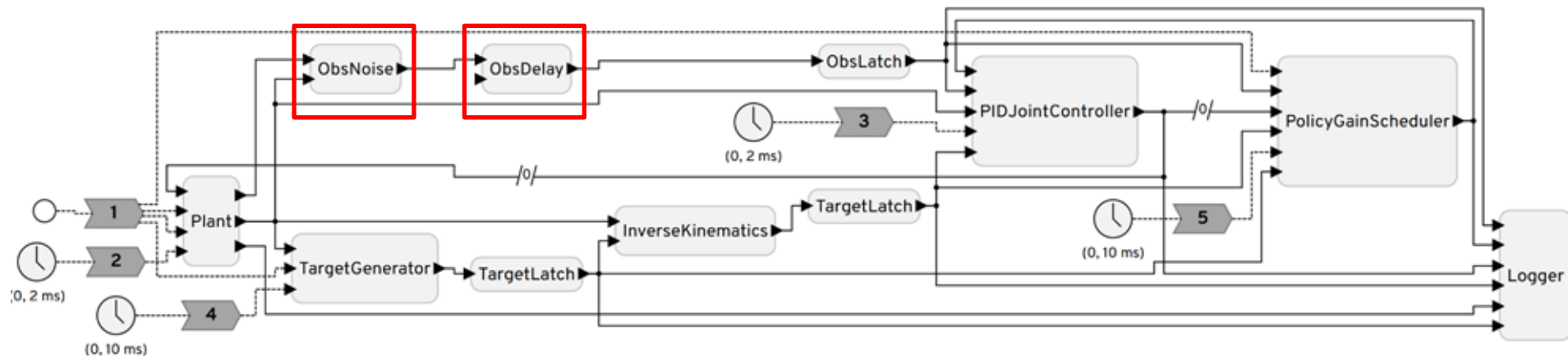


Reactor Model Architecture Overview



Plant: Simulates the 3-DoF MuJoCo leg model by applying the PID controller's joint torques and advancing the dynamics under a specified gravity vector, producing an observation vector at each step

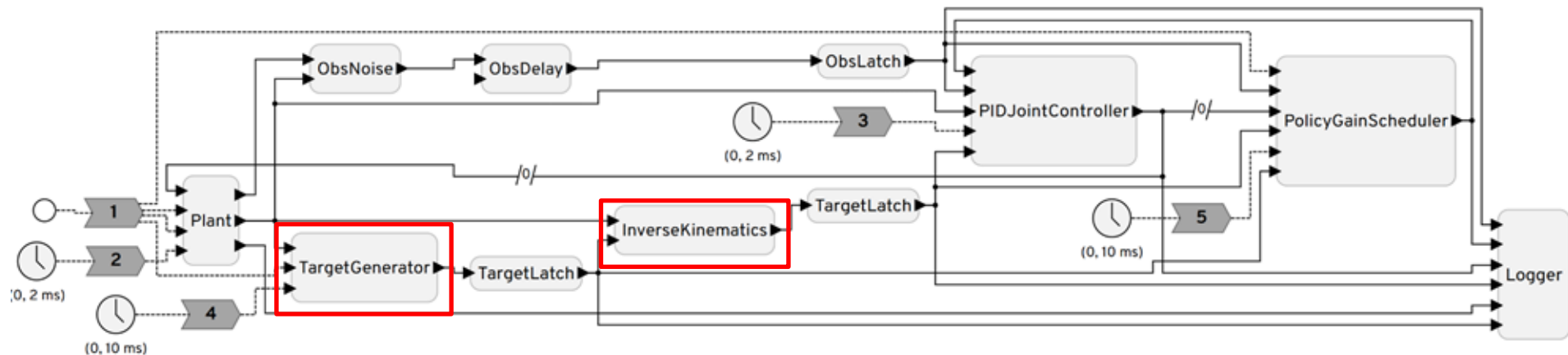
Reactor Model Architecture Overview



ObsNoise: Applies additive zero-mean Gaussian noise with σ to observations to simulate imperfect state estimation

ObsDelay: Delays observations by a configurable number of steps (shown as empty input) to emulate sensing/ communication latency of hardware deployment. Delayed observation feeds both PID feedback path and policy input

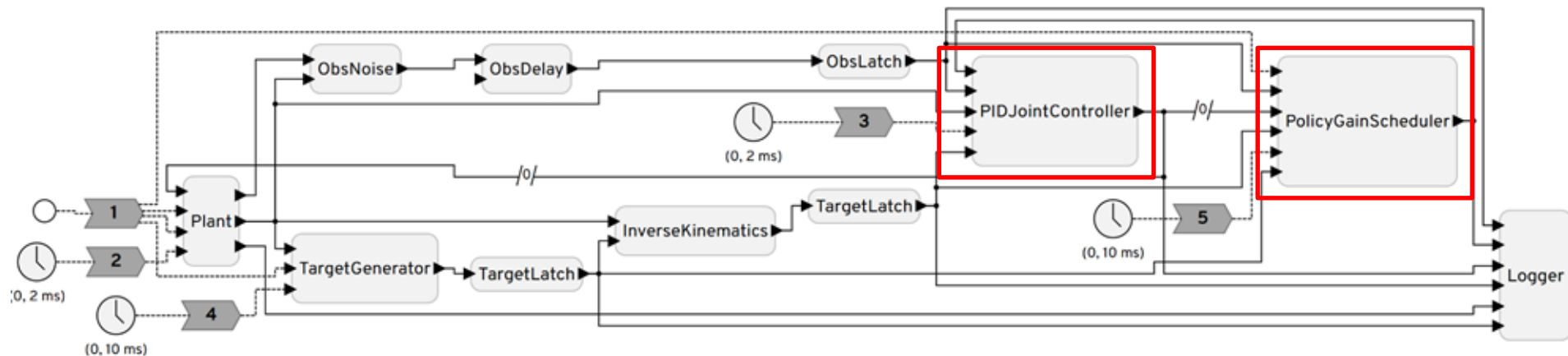
Reactor Model Architecture Overview



TargetGenerator: Generates a synthetic pose-holding, sinusoidal, or point-to-point tracking task

InverseKinematics: Converts generated task-space target into joint-space position & velocity targets

Reactor Model Architecture Overview



PIDJointController: Computes the control torque, parameterized by the proportional, integral, derivative, and anti-windup gains k_P, k_I, k_D and k_{AW}

PolicyGainScheduler: Sets the PID parameters from the observation vector, $\pi_\theta(O_t) \rightarrow \{\mathbf{k}_{P,t}, \mathbf{k}_{I,t}, \mathbf{k}_{D,t}, \mathbf{k}_{AW,t}\}$, with joint-wise gain vectors in $\mathbb{R}_{\geq 0}^3$

Reinforcement Learning (RL) Design

- Control problem as a partially observable Markov decision process (POMDP)
 - Direction and magnitude of gravity unknown
 - Sensor noise
- A policy is learned to approximate
 - $\pi_{\theta}(O_t) \rightarrow \{\mathbf{k}_{P,t}, \mathbf{k}_{I,t}, \mathbf{k}_{D,t}, \mathbf{k}_{AW,t}\}$
 - In PolicyGainScheduler
- Using PID controller structure to improve learning

Reward Function for Training of RL

$$R_t = -(r_{\text{track}} + r_{\text{ctrl}} + r_{\text{sat}} + r_{\text{gain}})$$

Each for tracking error, control effort, actuator saturation, and gain churn

- $r_{\text{track}} = w_x \|\mathbf{e}_t^{\text{task}}\|^2 + w_q \|\mathbf{e}_t^{\text{joint}}\|^2$ **[for track-space error and joint-space error]**
- $r_{\text{ctrl}} = w_v \|\dot{\mathbf{q}}_t\|^2 + w_\tau \|\mathbf{u}_t\|^2 + w_{\dot{\tau}} \|\Delta \mathbf{u}_t\|^2 + w_{\ddot{q}} \|\ddot{\mathbf{q}}_t\|^2$ **[for joint velocity, commanded effort, effort change, joint acceleration]**
- $r_{\text{sat}} = w_s \|\max(|\mathbf{u}_t| - 0.98, 0)\|^2$ **[for near-limit effort]**
- $r_{\text{gain}} = w_k \|\mathbf{k}_t \oslash \mathbf{k}_{\text{max}}\|^2 + w_{\dot{k}} \|(\mathbf{k}_t - \mathbf{k}_{t-1}) \oslash \mathbf{k}_{\text{max}}\|^2$ **[For gain magnitude and gain churn (change in gain)]**

Weights of Reward Function

Weight	Penalizes	Applied to	Value
w_x	Task tracking error	$\mathbf{e}_t^{\text{task}}$	20.0
w_q	Joint tracking error	$\mathbf{e}_t^{\text{joint}}$	1.0
w_v	Joint velocity	$\dot{\mathbf{q}}_t$	0.02
w_τ	Commanded effort	$\mathbf{u}_t$	0.02
$w_{\dot{\tau}}$	Effort rate	$\Delta \mathbf{u}_t$	0.0
$w_{\ddot{q}}$	Joint acceleration	$\ddot{\mathbf{q}}_t$	0.0
w_s	Near-saturation effort	$\max(\mathbf{u}_t - 0.98, 0)$	0.2
w_k	Gain magnitude	$\mathbf{k}_t \oslash \mathbf{k}_{\max}$	0.0
$w_{\dot{k}}$	Gain rate	$(\mathbf{k}_t - \mathbf{k}_{t-1}) \oslash \mathbf{k}_{\max}$	0.005

Evaluation

Goal: Demonstrate effectiveness & deterministic simulation behavior of approach

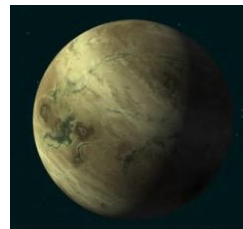
Three task benchmarks for a single 3 DOF leg of hexapod robot:

- Holding a fixed pose
- Following a sinusoidal trajectory
- Following a point-to-point trajectory



Three gravity environments:

- Moon: 1.625 m/s^2
- Earth: 9.81 m/s^2
- Kepler-452b: 12.164 m/s^2

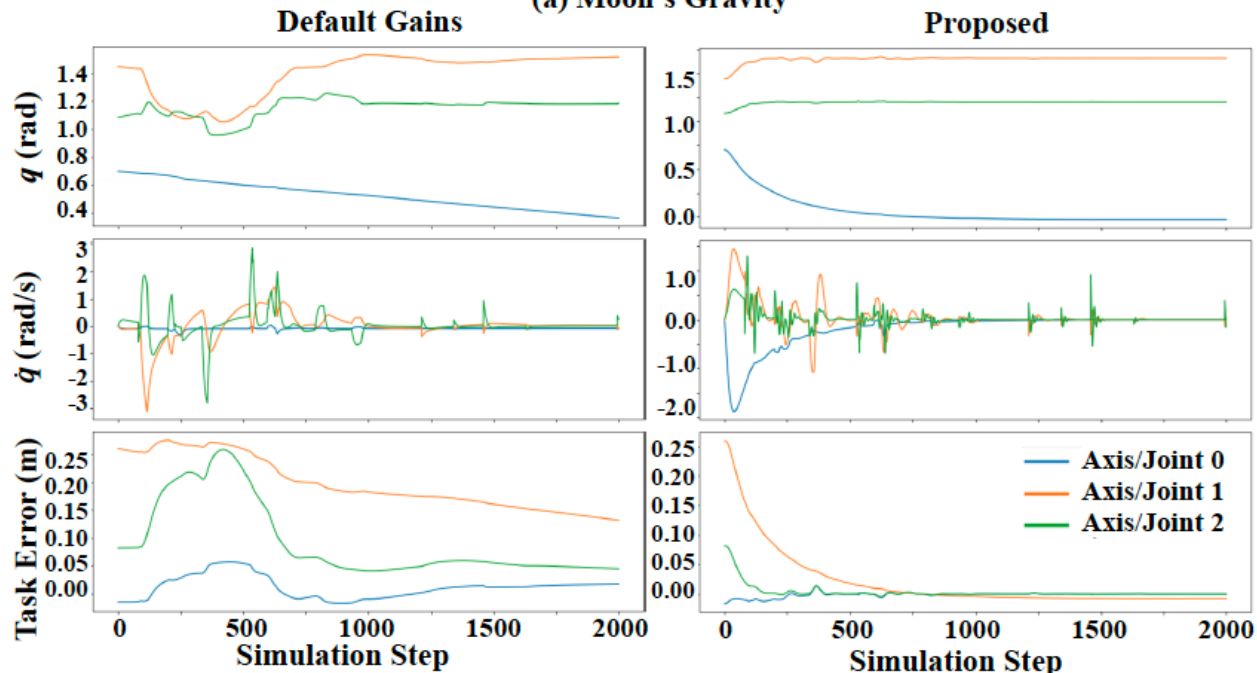


Compared against a fixed **default-gain** PID reference sharing the same inverse kinematic (IK) targets, PID structure, actuator limits, and leg model

Evaluation Results – Overall Behavior

For Pose-Holding Task

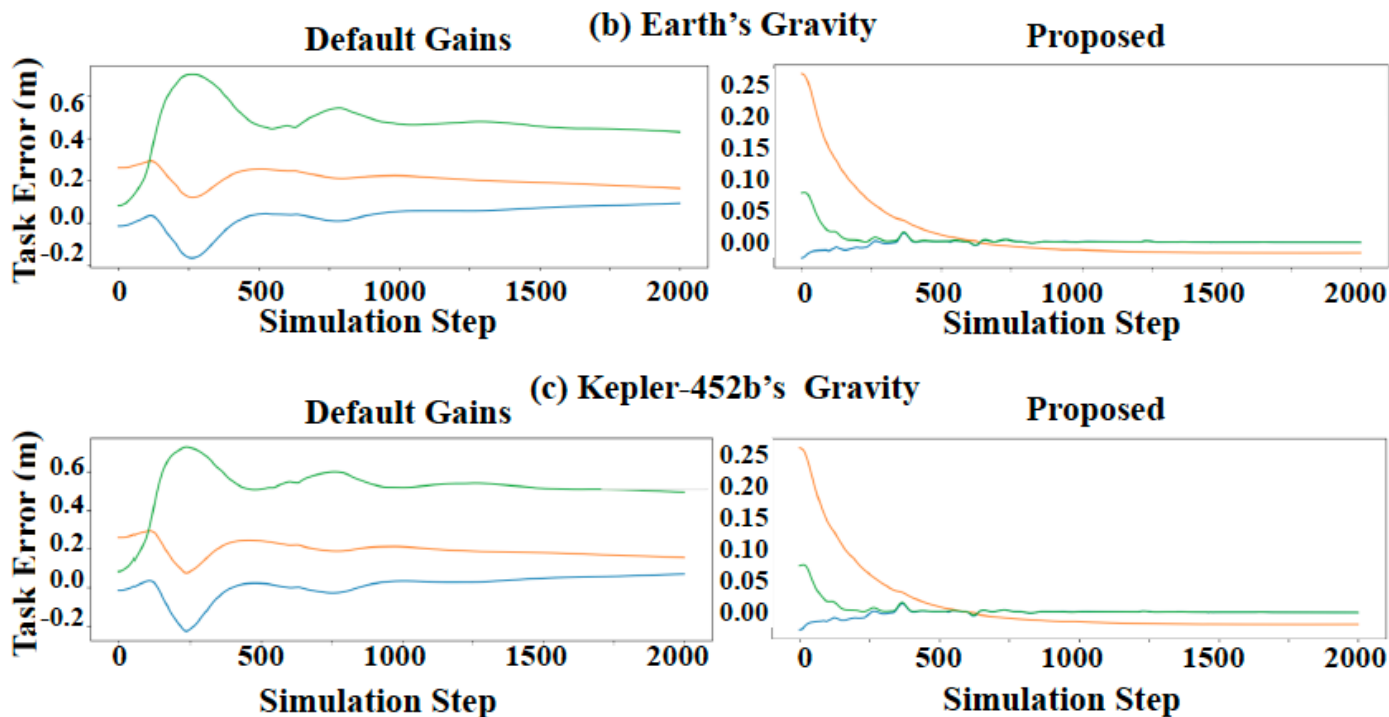
(a) Moon's Gravity



Note: All simulations are deterministic & repeatable for given initial conditions and configurations, allowing high fidelity of simulation and testability of the system model

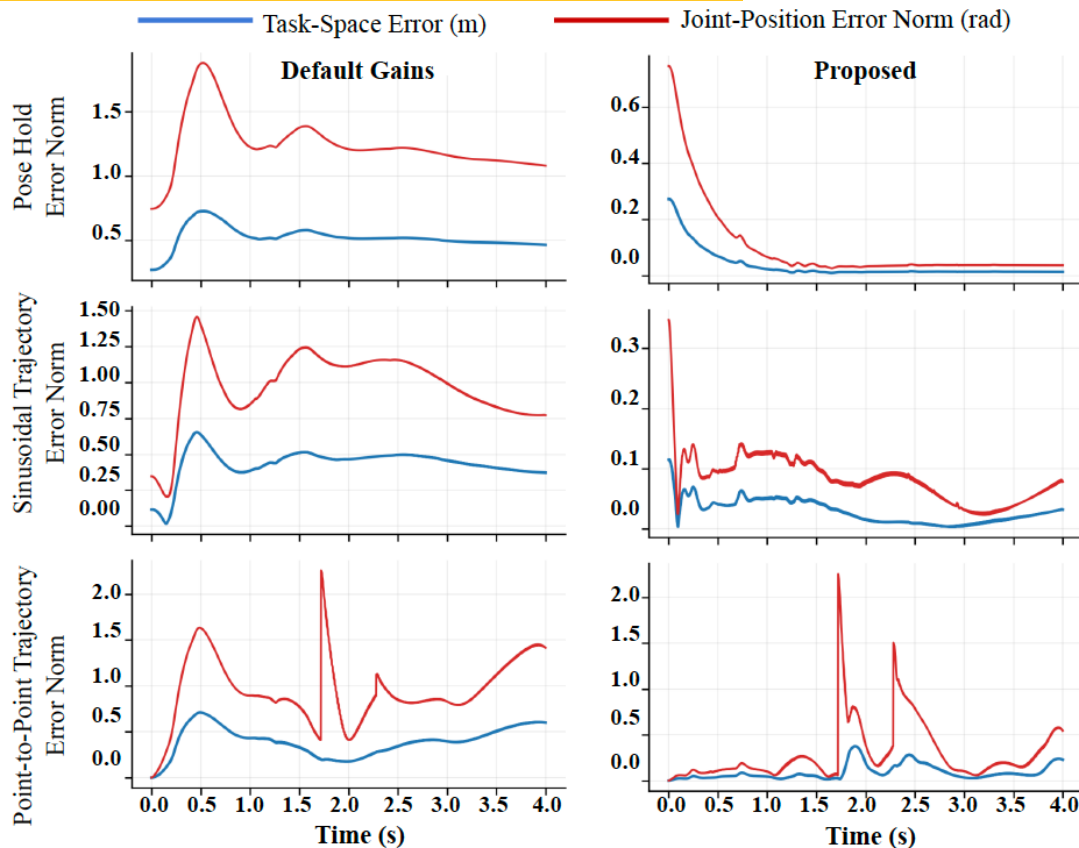
Evaluation Results – Overall Behavior

For Pose-Holding Task



Evaluation Results – For Each Task

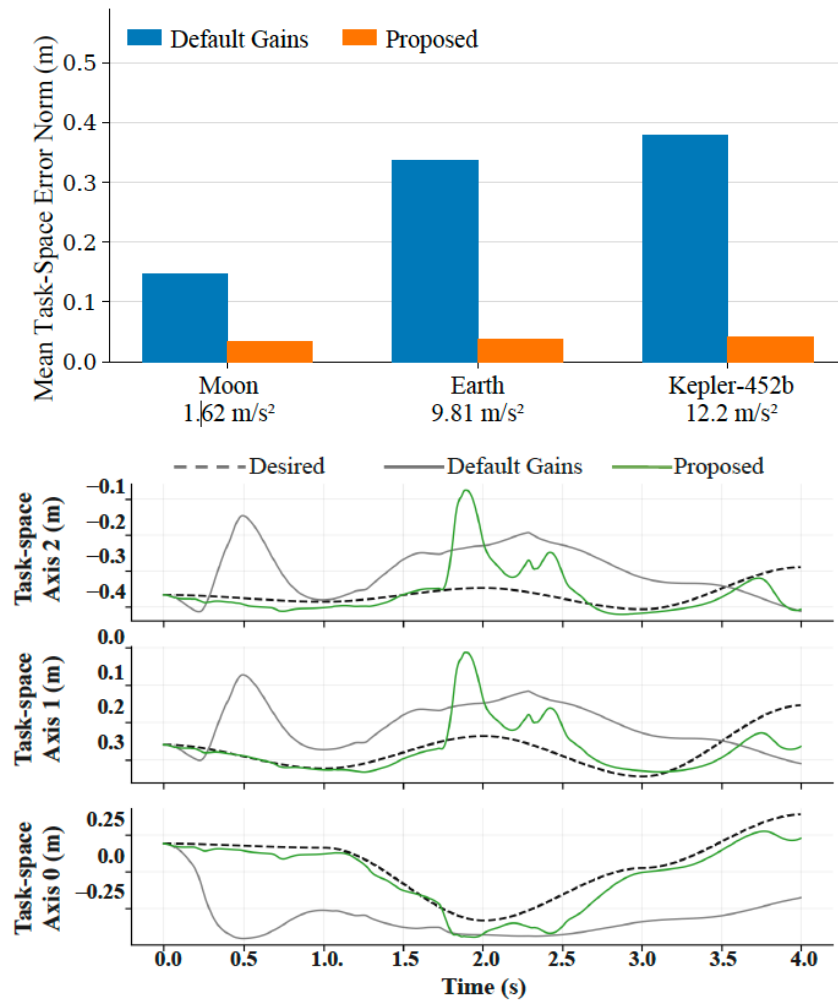
- Comparison of task- and joint-space errors for three task classes, normalized across all gravity environments



Evaluation Results

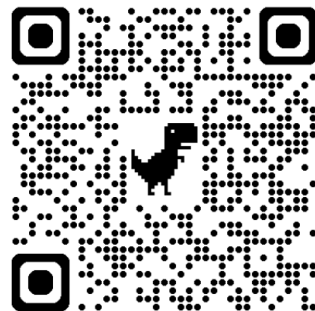
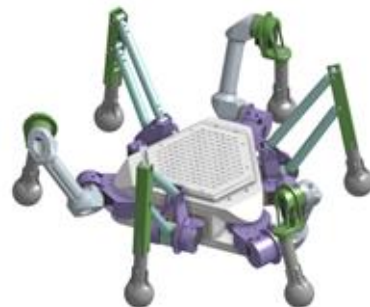
– Summary

- Summary of temporal mean task errors under three gravity settings
- Default gain reference shows larger deviations, whereas PPO-scheduled response shows smaller deviations
- **Limitation of evaluation:** Reference was not systematically tuned; these results do not determine whether a well-tuned controller could cover the same gravity range.



Demo (If Time Allows)

<https://github.com/asu-kim/mujoco-python-rl>



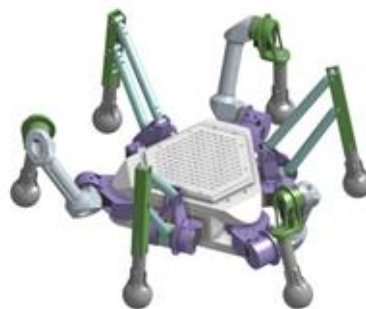
Steps

- Python virtual environment setup and installation of requirements
- Compilation (code generation) of the Lingua Franca program (`lfc src/Main.lf`)
- Execution of Lingua Franca program (simulation, `bin/Main`)
- Visualization of simulation results (`mjpython sim/run_sim.py`, on Mac OS X)

Conclusion and Future Work

- The PPO scheduler can adjust the joint-specific PID and anti-windup gains for a hexapod leg with 3 DOF
- The mean task error for the proposed controller was much less compared to the fixed default-gain reference
- Future work will compare the scheduler against fine-tuned fixed gains and classical gains and evaluate with multiple training seeds.

<https://github.com/asu-kim/mujoco-python-rl>



Contact: hokeun@asu.edu
<https://hokeun.github.io/>
<https://labs.engineering.asu.edu/kim/>



ASU KIM
KNOWLEDGEABLE &
INTERACTIVE MACHINES