MELP: Model Embedded Linear Policies for Robust Bipedal Hopping

Raghav Soni, Guillermo A. Castillo, Lokesh Krishna, Ayonga Hereid, Shishir Kolathaya

***Abstract*— Linear policies are the simplest class of policies that can achieve stable bipedal walking behaviors in both simulation and hardware. However, a significant challenge in deploying them widely is the difficulty in extending them to more dynamic behaviors like hopping and running. Therefore, in this work, we propose a new class of linear policies in which template models can be embedded. In particular, we show how to embed Spring Loaded Inverted Pendulum (SLIP) model in the policy class and realize perpetual hopping in arbitrary directions. The spring constant of the template model is learned in addition to the remaining parameters of the policy. Given this spring constant, the goal is to realize hopping trajectories using the SLIP model, which are then tracked by the bipedal robot using the linear policy. Continuous hopping with adjustable heading direction was achieved across different terrains in simulation with heading and lateral velocities of up to $0.5m/sec$ and $0.05m/sec$, respectively. The policy was then transferred to the hardware, and preliminary results (> 10 steps) of hopping were achieved.**

Keywords: *Humanoid and Bipedal Locomotion, Reinforcement Learning*

I. Introduction

Recent advances in actuation, computation, and sensing have made legged robots a viable technology for deploying in environments engineered for humans. These results are often conservative owing to the controlled settings of their deployed environments. However, the fullest potential of these platforms can be unleashed through their deployment as first responders in uncontrolled environments which require highly athletic and aggressive manoeuvres akin to their human counterparts, which is currently limited by existing control strategies. Established methods like Zero Moment Point (ZMP) [18], Linear Inverted Pendulum models [5] and Hybrid Zero Dynamics (HZD) [20] can provide robust walking but fail to extend to the expected aperiodic transient behaviours. Their success is also subject to rigorous tuning and gain scheduling [9] for their real-world deployment.

Sophisticated techniques from optimal control, like trajectory optimization (TO), model predictive control (MPC) and whole body impulse control (WBIC), show great promise in realizing dynamic behaviours like jumps and backflips [6]. Despite MPC's success, scaling it to acrobatic motions with longer aerial phases is challenging as their success and real-time deployment is conditional to the use of simplified dynamics [6], [5] and shorter planning horizons. TO provides a workaround to overcome this limitation by generating libraries of trajectories utilizing the full order dynamics that can then be tracked online [13] [12], [14] but take longer to compute and restrict these techniques to be offline. The stochasticity and non-stationarity of real-world conditions demand the replanning or adaptation of these trajectories, which at present can only be achieved through a highly-engineered synergy between the offline full order TO and online reduced order MPCs and WBIC's [4], which requires rigorous modeling and tuning.

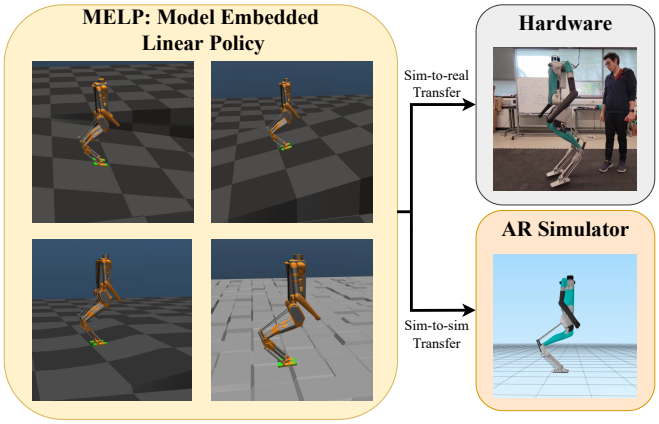

Fig. 1: *Figure showing generalisation of policy across different terrains and domains.*

On the other end of the spectrum, data-driven methods have shown compelling results in synthesizing controllers for legged robots [17], [2]. Their success can be attributed to their exposure to full-order dynamics, allowing them to learn unrestricted, efficient and novel behaviours, unlike their model-based equivalents. Additionally, they provide a more flexible framework to integrate and model the uncertainty observed in real-world conditions, thereby enhancing their sim-to-real transfer [16]. [10] recently showed that such policies for controlling a highly non-linear system typically parameterized as deep neural networks tend to linearize it such that a low-dimensional linear system can represent the entire high-dimensional closed-loop system, thus shining light on a major drawback of deep reinforcement learning based control, i.e. interpretability and safety. These findings complement our prior works [8], [7] that learnt linear control policies for robust blind bipedal walking in simulation and showed direct transfer to hardware. Despite the policy's success in bipedal walking, the use of a hand-picked foot trajectory limits extending the framework to more athletic

This work is supported in part by the Pratiksha Trust and the National Science Foundation under grant FRR-21441568.

R. Soni and L. Krishna are with the department of Electronics Engineering, Indian Institute of Technology (BHU) Varanasi, India.

G. A. Castillo and A. Hereid are with the department of Mechanical and Aerospace Engineering, Ohio State University, Columbus, OH, USA.

S. Kolathaya is with the department of Computer Science and Automation and the Centre for Cyber-Physical Systems, Indian Institute of Science, Bengaluru, India.

behaviours such as hopping. Selecting such task-specific trajectories is infeasible and hence calls for the utility of model-based planning. Thus, in this work, we propose to extend our design philosophy of learning linear control policies for realizing bipedal hopping through the integration of a dynamical model. Bipedal hopping is an ideal candidate for such a motion as it requires establishing an optimal balance of handling the body momentum and precise foot placements after significant flight phases. [15] demonstrates agile hopping but at the cost of design simplifications of the deployed hardware, and hence cannot be readily extended to full-order humanoids. Hence, we propose alleviating such design limitations by learning the controller in simulation. Our contribution is twofold.

1) Providing a control framework for integrating reduced order models in model-free learning and realizing robust bipedal hopping.

2) A simple yet effective policy structure to integrate any number of diverse control strategies to automate the cumbersome parameter tuning (as found in model-based control) through training in simulation.

The paper is structured as follows: In Section II, we present the control framework and meticulously explain our design choices. Section III showcases the results and analysis followed by the conclusion in Section IV.

II. Methodology

In this section, we describe our control framework and explain the working of the high-level linear policy in adherence with the low-level phase controller. The experiments were performed on the humanoid robot Digit. Digit is a 30 degrees of freedom (DoF) 3D biped developed by Agility Robotics, USA. The total weight of the robot is 48 kg, from which 22 kg corresponds to the upper body, and 13 kg to each leg. The kinematic structure of Digit is extensively discussed in our previous work [7].

A. Overview of the Control Framework

In [7], we developed a policy that enabled robust walking by modulating an elliptical trajectory for the feet using a linear policy. This approach limits the realisation of more athletic behaviours since elliptical trajectories are restrictive in nature. In our experiments, extending [7] to hopping, we found elliptical trajectories to be inadequate, causing the policies to fail. Gaits like hopping require more complex trajectories, which cannot be hand-picked. Therefore, we propose to integrate the linear policy with a model that plans the desired template motion online once every hopping cycle. This embedded model provides a reference trajectory to the linear policy for executing arbitrary motions, unlike fixed foot trajectory modulators with fixed gaits [8], thereby extending the framework. It is worth mentioning that the framework is independent of the chosen model and can be readily extended to diverse models as per the task and motion requirements. For instance, in our case, we use a SLIP model for hopping, whereas, for more complicated motions like backflips and parkour, we believe the approach could be extended to centroidal and full-order models. Thus the proposed hierarchical control framework, as shown in Fig. 2, has a model-embedded linear policy at the high level and a phase controller, transforming the policy's actions at the low level. The learnt linear policy generates unscaled motor commands for the leg joints during the stance phase and desired foot placement in the flight phase. The policy's outputs are then transformed to the appropriate scale or target joint angles through inverse kinematics by the phase controller in accordance with the given phase. These controllers are further explained in detail as follows.

B. High-Level Model Embedded Linear Policy

To generate motions with significant flight phases, we integrate the learnable linear policy with a *Spring-Loaded Inverted Pendulum* (SLIP) model as it is the fastest and simplest model that can generate trajectories for athletic behaviours like hopping and running in real-time [3]. A new SLIP trajectory is generated at every touchdown with Digit's reduced current state as initial conditions. This trajectory is then used as a reference by the linear policy for the upcoming stance and flight phase. The holistic linear policy is parameterized by a single matrix with different rows pertaining to different outputs for stance and flight phases. By being a part of the same policy, different feedback terms are trained together and converge to an optimal consensus between trajectory tracking and torso balance.

1) Trajectory Generation: The SLIP model assumes a point mass m and a springy leg of length l_0 with spring constant k. The dynamics in polar coordinates for the SLIP model during stance, with the foot anchored at the origin, are given as:

$$m\ddot{r} - mr\dot{\theta} + mg\cos(\theta) - k(l_0 - r) = 0 \tag{1}$$

$$mr^2\ddot{\theta} + 2mr\dot{r}\dot{\theta} - mgr\sin(\theta) = 0 \tag{2}$$

Owing to the low inertia legs of Digit, the centre of mass can be reasonably approximated to be the base frame defined at the hip, as most of the mass is concentrated around the pelvis. Hence, we use the robot's base state as the state of the COM for SLIP. At every touchdown, the initial condition for the SLIP model is given as:

$$\begin{bmatrix} x & \dot{x} & z & \dot{z} \end{bmatrix}^T = \begin{bmatrix} X_b & \dot{X}_b^d & Z_b & \dot{Z}_b^d \end{bmatrix}^T \tag{3}$$

where X_b and Z_b are x and z coordinates of the base with respect to the feet, and $\dot{X}_b^d$ and $\dot{Z}_b^d$ are the desired longitudinal and vertical velocities, respectively. Due to the symmetry between the liftoff and touchdown velocities, we initialise the trajectories with the desired heading velocities. $\dot{Z}_b^d$ is calculated based on the desired maximum jump height. The dynamics equations are numerically integrated over time to obtain a trajectory for the stance phase. It is worth noting that the above equation is for a 2D SLIP model where the motion is constrained to the sagittal plane, unlike the unconstrained real system in 3D. However, since the

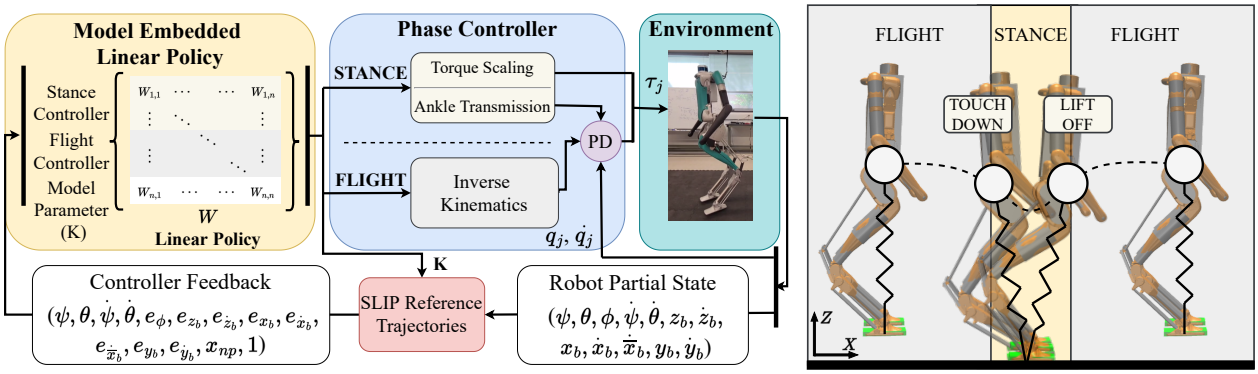

Fig. 2: *Figure showing the control framework (left) and the SLIP trajectory (right) with flight and stance phases separated by touchdown and liftoff*

policy is trained to learn stabilizing behaviours for the full-order system in simulation, it is unaffected by the fidelity of the template model, thereby proving its robustness and generality.

*2) **Observation Space**:* In [7], the effectiveness of using a reduced observation space was exhibited. Keeping up with the same philosophy, we formulate the observation space with torso orientation and trajectory tracking errors, allowing the linear policy to marginally deviate from the reference trajectories and trade-off accurate tracking for torso balance. Thus observation space is a 13-dimensional state vector defined as-

$$s_t = [\psi, \theta, \dot{\psi}, \dot{\theta}, e_\phi, e_{x_b}, e_{y_b}, e_{z_b}, e_{\dot{x}_b}, e_{\dot{y}_b}, e_{\dot{z}_b}, e_{\bar{\dot{x}}_b}, x_{np}, 1]^T \tag{4}$$

Where ψ and θ are the torso roll and pitch, and $\dot{\psi}$ and $\dot{\theta}$ are the corresponding angular velocities. $e_{\square_b} = \square^d - \square$, where $\square \in \{\phi, x_b, y_b, z_b, \dot{x}_b, \dot{y}_b, \dot{z}_b\}$, are the current heading yaw, base positions and linear velocities with respect to the feet and $\square^d \in \{\phi^d, x_b^d, y_b^d, z_b^d, \dot{x}_b^d, \dot{y}_b^d, \dot{z}_b^d\}$, are the corresponding desired terms which are obtained from the SLIP trajectory, and user commands. $e_{\bar{\dot{x}}_b}$ is the error between the mean velocity of a hopping step and the desired velocity of the base along the heading direction. Akin to [15], x_{np} is the nominal foot position calculated as $x_{np} = 0.5\dot{\bar{x}}_b T_s$, where T_s is the time period for the last stance phase. However, by providing $e_{\bar{\dot{x}}_b}$ and x_{np} as inputs to the policy, we learn the relative weighting between the nominal and feedback terms through training and thereby enabling it to handle early contacts. A normalised constant (1) is passed for estimating the stiffness constant k for the SLIP model as it is independent of any of the system's state variables.

*3) **Action Space**:* The action space of the policy can be broadly classified into four parts:

a) Torque actions for stance: During stance, the policy outputs unscaled torques for the hip (hr, hp) and knee (kp) motors. Since there is a significant difference in the magnitude of torques produced by position and velocity feedback, denoted by $a_\square^p$ and $a_\square^v$ respectively, where $\square \in \{hr, hp, kp\}$, we assign separate dedicated rows in the policy to output the corresponding torques and thereby keeping the matrix elements normalised. These torques are then appropriately scaled and added together to form the final joint torques. An additional action a_{kp}^l is also inferred from the policy for maintaining lateral stability. This action is added to the knee joint of one leg and subtracted from the other. The resulting torque difference accounts for nullifying the lateral disturbances.

b) Ankle transmission for stance: For the ankles, the policy provides the target roll and pitch values with respect to the current roll and pitch of the ankle. These actions are denoted as $\acute{q}_{ap}^d$ and $\acute{q}_{ar}^d$.

c) Target foot placement during flight: The policy estimates a fixed target foot position for the touchdown with dynamic adjustments to correct for external disturbances. The fixed target position is denoted as x_f and the dynamic offsets along the x and y directions are denoted as $\acute{x_f}$ and $\acute{y_f}$. A target joint angle for the hip yaw motor, denoted by q_{hy}^d, is also estimated to control the desired heading yaw.

d) Model parameter for SLIP: The policy also estimates the stiffness constant k for the SLIP model. This is done to identify a spring constant that would closely mimic the dynamics in the template motion.

Hence, the action space is a 12-dimensional vector defined as-

$$a_t = [a_{hr}^p, a_{hr}^v, a_{hp}^p, a_{hp}^v, a_{kp}^p, a_{kp}^v, a_{kp}^l, \acute{q}_{ap}^d, \acute{q}_{ar}^d, x_f, \acute{x_f}, \acute{y_f}, q_{hy}^d, k]^T \tag{5}$$

*4) **Policy Matrix Sparsification**:* With [7], we observed that sparsification of the matrix could help reduce the overfitting of the policy to the simulation dynamics, speed up the training, and enhance the transfer to hardware. In our current work, the sparsification becomes even more critical as we combine actions for different phases in the same policy. For example, the flight phase touchdown position commands are not related to the stance phase's trajectory errors. Therefore, we learn only the relevant parameters that correlate an observation to action and set all the other terms to 0, as seen in Fig. 3. We give a warm start to the training by initialising it with a sub-optimal policy that is able to hop for a step or two before falling.

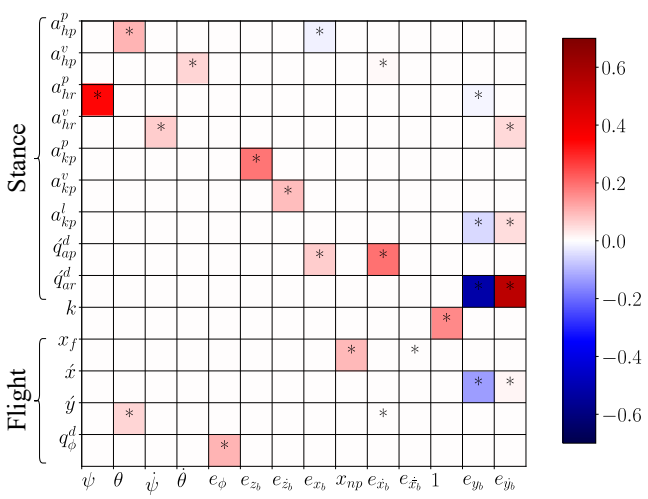

Fig. 3: *Figure showing the trained policy matrix with values denoted as a color map. The symbol '*' marks the non-zero terms that were optimised through ARS.*

C. Low-Level Phase Controller

The hopping gait has two phases: *stance and flight*, which are separated by the events of touchdown and lift-off. At the low level, the phase controller is responsible for switching the control received from the policy between the different phases and processing the actions accordingly.

While landing from the flight, the touchdown is detected through the compression in a passive spring attached to the shin of the Digit's leg. When Digit reaches back to a height equal to that of the touchdown, we assume that the lift-off has occurred in accordance with the template model. The phase controller processes the actions from the policy depending upon the phase as follows.

Stance: During stance, the gait controller performs torque scaling and ankle regulation.

a) Torque Scaling: As discussed before, we scale the unscaled actions and do a weighted sum of the position, and velocity feedback torques as follows,

$$\tau_i = S^p_i a^p_i + S^v_i a^v_i \tag{6}$$

where τ_i is the motor torque, S^p_i and S^v_i are the scaling factors for position and velocity components, respectively. $i \,\epsilon\, \{hr, hp\}$. For the knee pitch, we have:

$$\tau_{kp} = S^p_{kp}(a^p_{kp} + \mathcal{F} a^l_{kp}) + S^v_{kp} a^v_{kp} \tag{7}$$

The value of $\mathcal{F}$ is defined as, $\mathcal{F} = 1$ for the right leg, and $\mathcal{F} = -1$ for the left leg. Hence, the left and the right knee can have a difference in torque, allowing for auxiliary and asymmetric lateral stability.

b) Ankle Transmission: As the generated SLIP trajectories assume a point-foot and do not consider 2 DoF at the ankle, we need an explicit transmission for the ankle due to the non-linear relation between the desired angles and the ankle commands [7]. The actuation at the ankle is crucial in maintaining Digit's balance during stance. The target roll and pitch for the ankle are obtained as follows.

$$q^d_{ar} = q_{ar} + \mathcal{F}(\acute{q}^d_{ar} - Q^{off}_{ar}) \tag{8}$$

$$q^d_{ap} = q_{ap} + \mathcal{F}(\acute{q}^d_{ap} - Q^{off}_{ap}), \tag{9}$$

where q^d_{ar} and q^d_{ap} are the desired angles and $Q^{off}_{ar} = 0.366$ rad and $Q^{off}_{ap} = 0.04$ rad are the joint offsets in accordance to the kinematic assembly for the ankle joints. The desired roll and pitch commands for the ankles are transformed into the motor commands through ankle kinematics approximated using non-linear regression.

Flight: During the flight phase, the phase controller receives the target x foot position from the policy. We consider a straight-line trajectory for the feet from the position at the liftoff to the target touchdown. This trajectory is dynamically altered at each control step with foot placement residues, $\acute{x_f}$ and $\acute{y_f}$. Ideally, the trajectory is to be tracked in time T_F, which is the average time of flight of the gait. At time t during the flight phase, the target foot positon is calculated as follows.

$$x^d_f = \begin{cases} S_{x_f} x_f + S_{\acute{x_f}} \acute{x_f}, & \text{if } t > T_F \\ \frac{(S_{x_f} x_f - x^i_f)t}{T_F} + x^i_f + S_{\acute{x_f}} \acute{x_f}, & \text{if } t \leq T_F \end{cases} \tag{10}$$

$$y^d_f = \begin{cases} S_{\acute{y_f}} \acute{y_f} + Y^{off}_f, \text{ for left leg} \\ S_{\acute{y_f}} \acute{y_f} - Y^{off}_f, \text{ for right leg} \end{cases} \tag{11}$$

$$z^d_f = Z^{off}_f \tag{12}$$

where $S_{x_f} = 20$, $S_{\acute{x_f}} = 10$ and $S_{\acute{y_f}} = 20$ are the scaling factors for the actions from the policy, x^i_f is the feet position with respect to the base at liftoff, and $Y^{off}_f = 0.1m$ and $Z^{off}_f = -1.0m$ are the offsets for the feet with respect to the base.

The desired joint positions are then obtained using *Inverse Kinematics* (IK) and tracked with a joint-level PD controller. The ankle joints are kept passive and the target angle for hip yaw motors (q^d_{hy}), is directly tracked with a PD controller. It is to be noted that all the scaling factors were constant for different simulation and hardware settings.

D. Policy Training

Similar to our previous work, we use Augmented Random Search (ARS) for training the linear policy. Unlike generic ARS, we only search in a sub-space of the parameter space, $\mathbb{R}^{m \times n}$. The linear policies trained using ARS are found to be effective for solving various continuous-control problems and require minimal tuning of hyper-parameters [11].

1) Reward Function: The reward function is formulated with the objective to keep the torso upright while having sufficiently long flight phases and simultaneously tracking the commanded heading velocities. It is defined as,

$$r = G_{w_1}(\psi) + G_{w_2}(\theta) + G_{w_3}(e_\phi) + G_{w_4}(e_{\dot{x_b}}) + f_r \tag{13}$$

where $f_r = 0.2$ in the flight phase and $f_r = 0.0$ otherwise. The mapping $G : \mathbb{R} \to [0, 1]$ is the Gaussian kernel given by $G_{w_j}(u) = \exp(-w_j \times u^2), w_j > 0$.

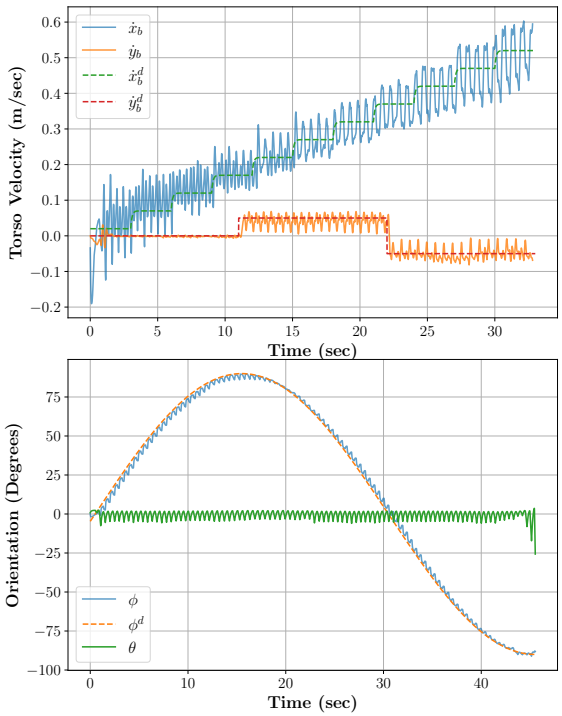

Fig. 4: *Figure showing command tracking in MuJoCo simulation. The longitudinal and lateral velocities, and the heading yaw are presented.*

2) Training Curriculum: We only train on flat ground across arbitrary heading velocity commands. The commanded heading velocity is uniformly sampled from the range $[0.0, 0.2]$ and either remains constant throughout the episode or increases with a step of 0.1 m/sec up to 0.2 m/sec every 5 seconds. Alongside, we also train for commanded lateral velocities sampled from $[-0.05, 0.05]$ m/sec and commanded heading yaw varied randomly in a quasi-static way. The length of each episode is 15 seconds. An episode is terminated if the robot topples or the robot's height decreases below a certain threshold, or if the maximum episode length is reached. Additionally, we introduce noise in the observation, randomise the scaling factors in the phase controller and arbitrarily reduce the saturation limits for torques to further robustify the policy.

III. Results

This section presents the simulation results, sim-to-sim transfer of the policies, and the preliminary hardware experiments conducted. We use a custom gym environment with the MuJoCo physics engine [19] for training our policies in simulation. We show the sim-to-sim transfer to the Agility Robotics (AR) Simulator, a simulation platform proprietary to AR. We also show preliminary sim-to-real transfer of the policies to the hardware.

A. MuJoCo Simulation Results

1) Performance Analysis: We train the policies on flat terrain for varied forward and lateral velocities, and yaw control. A 2D SLIP model is used to produce trajectories in the sagittal plane and the template motion is extended

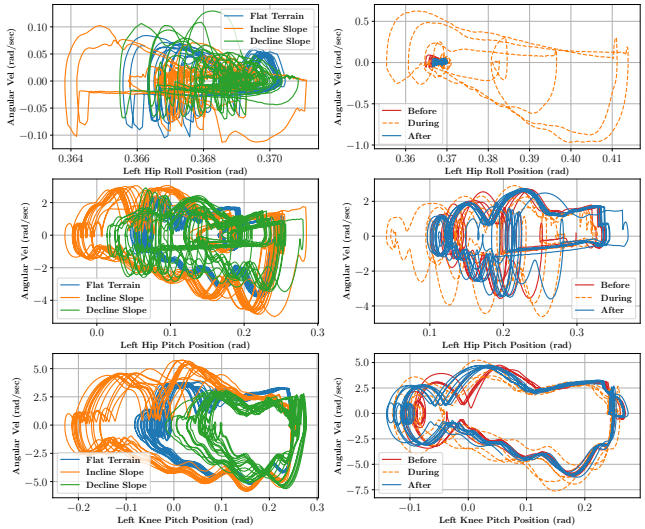

Fig. 5: *Figure showing how the limit cycles for the phase portraits of mentioned joints differ for different slopes (left) and phase portraits settling into a stable limit cycle after disturbance from the uneven terrain (right)*

to a complex 3D system like Digit, which is capable of performing lateral hops and adjusting the heading angle, by virtue of the policy design. The policy predicts actions for a single leg. As a result of the symmetric structure of Digit and most humanoid robots, the commands for one leg can be mirrored to the other leg.

We are able to achieve accurate tracking of heading velocities from 0 m/sec up to 0.52 m/sec and lateral velocities from -0.05 m/sec up to 0.05 m/sec as shown in Fig. 4. The heading yaw can be controlled freely, allowing the coverage of the complete 3D space with an omnidirectional heading. Maximum jumping height is increased with increasing longitudinal velocities to provide long enough flight phases for the feet to transition to the touchdown position. The maximum jumping height varies from 0.04 meters at 0 m/sec to 0.15 meters at 0.52 m/sec heading velocity. The main challenges in increasing the maximum jumping height were torso pitch disturbances and motor torque saturation at lower and higher longitudinal velocities, respectively. In adherence to the well-trained policy, the torso's oscillations were minimal and contained within the following ranges : $\psi \in (-1°, 1°), \theta \in (-5°, +7°), \phi \in (-1°, 1°)$. It can also be observed from Fig. 6 how the trained policy roughly establishes a proportional relationship between the heading velocity and the target foot placement during the flight phase.

2) Policy Generalisation and Robustness: We tested the policy trained on flat terrain under different terrain conditions, and it was found to be insusceptible to minor changes in the terrain. The policy performed well on the slopes ranging from $-11°$ to $11°$ without any terrain feedback or explicit training. The changes in the phase portraits for the slopes as compared to the flat terrain is shown in Fig. 5 (left). The policy successfully handled step disturbances in the terrain of up to 7 cm in height. The limit cycles of the

	Raibert's Controller	SLIP Controller	Initial Policy	Trained Policy
Max Horizontal Velocity (With perpetual hopping)	0.3 m/sec	0.3 m/sec	0.2 m/sec	0.5 m/sec
Time to destabilise (When a 40N forward force is applied continuously during every stance phase)	3 sec	5 sec	1 sec	Hops Perpetually
Time to destabilise (When a 30N sideways force is applied continuously during every stance phase)	2 sec	2 sec	3 sec	14 sec
Max Pitch Disturbance (For velocities up to 0.2 m/sec)	6 Deg	6 Deg	8 Deg	2.5 Deg
Hopping on Inclined Surface (Max Slope)	2 Deg	7 Deg	5 Deg	11 Deg
Hopping on Declined Surface (Max Slope)	4 Deg	7 Deg	7 Deg	11 Deg

TABLE I: Trained policy comparison with initial policy, Raibert's controller and task-space SLIP trajectory tracking controller on different metrics.

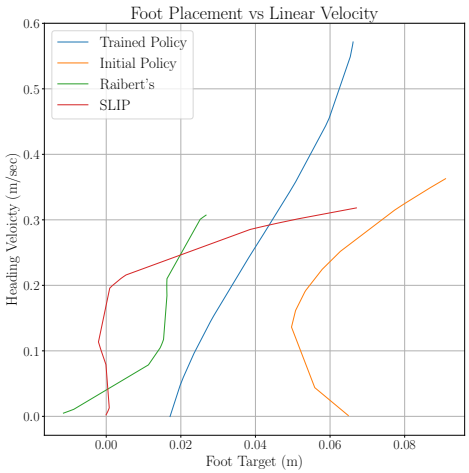

Fig. 6: Target Foot Placement during Flight phase for different controllers.

leg joints can be seen reacting to the terrain disturbances in Fig. 5 (right). The policy also rejected longitudinal forces ranging from -40 N to 40 N and lateral forces ranging from -30 N to 30 N applied periodically during the stance phase. All the mentioned results are presented in the accompanying video submission.

3) Comparison with the Baseline: To further evaluate the performance of the trained policy, we compare it against two manually tuned controllers. 1) Raibert's Linear Hopping with foot placement and attitude correction. 2) Task-space SLIP trajectory tracking with look-up table for foot placement.

Both controllers work only with active ankle and torso regulations which are manually tuned. For the Raibert's hopping controller, as the natural structure of Digit doesn't resemble a single springy leg, we assume legs to function like SLIP template model and perform energy shaping to jump to the desired height. We compare the controllers on different metrics which act as indicators of robustness and reliability. The results are shown in I. It can be observed that the trained policy performs substantially better on all the metrics.

B. Sim-to-sim transfer

To further evaluate the generalisation of the policy to different dynamic properties of the robot and the environment, we tested the policy in the AR simulator with no additional tuning to the policy parameters. The AR simulator is a high-fidelity real-time simulator that includes parameters corresponding to the actual hardware such as joint friction, communication delays, state estimation, etc.

In addition, testing the policy in the AR simulator results in an additional layer of safety that is important in the sim-to-real process to prevent non-expected behaviors on the real hardware.

We perform tests for velocity tracking and control of the heading angle, and robustness to challenging terrains. The performance of the learned policy for velocity tracking and control of heading angle are presented in Fig. 7. The policy effectively tracked longitudinal velocities from 0 m/sec up to 0.3 m/sec and lateral velocities from -0.05 m/sec up to 0.05 m/sec. We notice that the maximum velocity reached by the policy in the AR simulator is lower than the one obtained in the custom MuJoco simulator, which is a common effect in sim-to-sim and sim-to-real transfer, specially for dynamic maneuvers [1]. The combination of this commands allow the robot to jump in diagonal directions. In addition, the policy successfully tracked the continuous variations of the desired heading angle while jumping forward. Finally, we also tested the policy under different terrain conditions. The controller successfully traversed through challenging terrains, including slopes in a range of $[-10, 10]$ degrees. These results can be seen in the accompanying video submission.

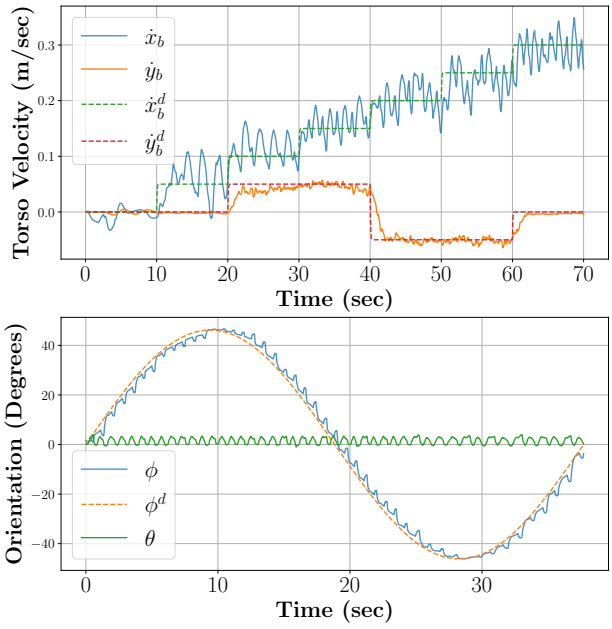

Fig. 7: *Figure showing velocity and heading angle tracking in the AR simulator.*

C. Preliminary hardware tests

We present some preliminary results of the testing of the leaned policy in the hardware of the Digit robot. Given the highly dynamic behaviors performed by the policy, there exists critical components that affect the performance of the policy in the real hardware, being one of them the state estimation algorithm used to compute the position and velocity of the robot.

Unfortunately, during the hardware tests, we found that the default state estimation algorithm included in the closed-source low-level API of the robot is sensitive to the takeoff and touchdown events. In particular, during the flight-phase and landing, there is a significant drift on the estimation of the vertical position and velocity of the robot's base. This estimation errors introduce significant disturbances to the policy input, causing high variance in the policy outputs, which results in instability of the whole system.

To address this problem, we use Forward Kinematics (FK) to compute the height of the robot's base during the double support phase and compensate for the drifting in the state estimation. Although the resulting motion is not highly dynamic like the one obtained in simulation, the robot is able to take jumps with prolonged phases of double support, in which the FK-based estimation of the robot's height is used to compensate for the drifting in the state estimator with the addition of a balancing controller to improve the stability of the stance phase.

Future work will focus on improving the state estimation algorithm to obtain reliable measurements of the robot's state that fully enables the deployment of the learned policy. The preliminary results for the hardware tests can be seen in the accompanying video submission.

IV. Conclusion

In summary, we presented a linear policy integrated with the SLIP template model, capable of performing robust and continuous hopping motion in 3D space. We exhibit the generalisation of the linear policy through successful trials on adverse simulation conditions, sim-to-sim transfer to a high-fidelity real-time simulator and preliminary experiments on the hardware. The proposed control framework presents a generic approach for performing agile and dynamic manoeuvres. Although, in this work we use SLIP model to demonstrate a particular case of hopping, we believe our approach can be readily extended to more complex behaviours.

Future research directions include extensive testing on the hardware with improved state estimation and integration of more complex models in the linear policy for more dynamic behaviours like running. The preliminary results for the hardware tests can be seen in the accompanying video submission. An extended version of the video with additional experiments can be seen here: https://youtu.be/iSK2JeBEPLw

References

[1] Ryan Batke, Fangzhou Yu, Jeremy Dao, Jonathan Hurst, Ross L. Hatton, Alan Fern, and Kevin Green. Optimizing bipedal maneuvers of single rigid-body models for reinforcement learning, 2022.

[2] Miroslav Bogdanovic, Majid Khadiv, and Ludovic Righetti. Model-free reinforcement learning for robust locomotion using demonstrations from trajectory optimization, 2021.

[3] Yu-Ming Chen and Michael Posa. Optimal reduced-order modeling of bipedal locomotion. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8753–8760, 2020.

[4] Matthew Chignoli, Donghyun Kim, Elijah Stanger-Jones, and Sangbae Kim. The mit humanoid robot: Design, motion planning, and control for acrobatic behaviors. In *2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids)*, pages 1–8, 2021.

[5] Yukai Gong and Jessy Grizzle. One-step ahead prediction of angular momentum about the contact point for control of bipedal locomotion: Validation in a lip-inspired controller. In *International Conference on Robotics and Automation (ICRA)*, 2021.

[6] Benjamin Katz, Jared Di Carlo, and Sangbae Kim. Mini cheetah: A platform for pushing the limits of dynamic quadruped control. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6295–6301, 2019.

[7] Lokesh Krishna, Guillermo A. Castillo, Utkarsh A. Mishra, Ayonga Hereid, and Shishir Kolathaya. Linear policies are sufficient to realize robust bipedal walking on challenging terrains. *IEEE Robotics and Automation Letters*, 7(2):2047–2054, 2022.

[8] Lokesh Krishna, Utkarsh A. Mishra, Guillermo A. Castillo, Ayonga Hereid, and Shishir Kolathaya. Learning linear policies for robust bipedal locomotion on terrains with varying slopes. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5136–5141, 2021.

[9] Zhongyu Li, Xuxin Cheng, Xue Bin Peng, Pieter Abbeel, Sergey Levine, Glen Berseth, and Koushil Sreenath. Reinforcement learning for robust parameterized locomotion control of bipedal robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, Xi'an, China, June 2021.

[10] Zhongyu Li, Jun Zeng, Akshay Thirugnanam, and Koushil Sreenath. Bridging model-based safety and model-free reinforcement learning through system identification of low dimensional linear models. 2022.

[11] Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search of static linear policies is competitive for reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 1800–1809, 2018.

[12] A. Marco, D. Baumann, M. Khadiv, P. Hennig, L. Righetti, and S. Trimpe. Robot learning with crash constraints. *IEEE Robotics and Automation Letters*, 6(2):1439–1446, February 2021.

[13] Quan Nguyen, Matthew J. Powell, Benjamin Katz, Jared Di Carlo, and Sangbae Kim. Optimized jumping on the mit cheetah 3 robot. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7448–7454, 2019.

[14] Brahayam Ponton, Majid Khadiv, Avadesh Meduri, and Ludovic Righetti. Efficient multi-contact pattern generation with sequential convex approximations of the centroidal dynamics. *IEEE Transactions on Robotics*, Early access:1–19, February 2021.

[15] Marc H Raibert. Legged robots. *Communications of the ACM*, 29(6):499–514, 1986.

[16] Jonah Siekmann, Kevin Green, John Warila, Alan Fern, and Jonathan Hurst. Blind bipedal stair traversal via sim-to-real reinforcement learning, 2021.

[17] Jonah Siekmann, Srikar Valluri, Jeremy Dao, Lorenzo Bermillo, Helei Duan, Alan Fern, and Jonathan Hurst. Learning memory-based control for human-scale bipedal locomotion. In *Robotics Science and Systems*, 2020.

[18] R. Tedrake, T. W. Zhang, and H. S. Seung. Stochastic policy gradient reinforcement learning on a simple 3d biped. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pages 2849–2854 vol.3, 2004.

[19] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012.

[20] E. R. Westervelt, J. W. Grizzle, and D. E. Koditschek. Hybrid zero dynamics of planar biped walkers. *IEEE Transactions on Automatic Control*, 48(1):42–56, Jan 2003.