Realising the role of arms on improving the stability bipedal robots

Manan Tayal[1], Shishir Kolathaya[1]

***Abstract*— This paper shows the stabilizing effect of the upper body (especially the arms) on improving the stability of Bipedal robots. We demonstrate the efficacy on MuJoCo simulations of a Bipedal Robot Digit.**

***Index Terms*— Bipedal Walking, Hopping, Reinforcement Learning**

I. Introduction

Most of the research in the field of controls of Legged locomotion focuses more on the control of lower body while walking, hopping or running, which leaves the torso unstable. Swinging arms in an opposing direction with respect to the lower limb reduces the angular momentum of the body, balancing the rotational motion produced during walking and thereby reducing the disturbance in the torso. Moreover it also helps in reducing the energy spent during locomotion [1].

II. Robot model

In this section, we describe the robot model on which the proposed framework is tested. We also introduce the mathematical notations used throughout the paper.

A. Robot model and Notations

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. Each arm has 4 DoF corresponding to the shoulder roll, pitch and yaw joints (q_{sr}, q_{sp}, q_{sy}) and the elbow joint (q_e). Each leg consists of eight joints, including three actuated hip joints (hip roll, yaw, and pitch (q_{hr}, q_{hp}, q_{hy})), one actuated knee joint (q_k), two actuated ankle joints (toe pitch and roll (q_{tp}, q_{tr})), and three passive joints corresponding to shin-spring (q_{ss}), tarsus (q_t), and heel-spring joints (q_{hs}). To differentiate between left and right leg joints, we add the superscript L and R respectively to each of the joints. The position and orientation of the robot's base is denoted by:

$$\mathbf{q^b} = [p_x, p_y, p_z, \psi, \theta, \phi]^T, \tag{1}$$

where p_x, p_y, p_z correspond to the base translation and ψ, θ, ϕ correspond to the base orientation (roll, pitch and yaw angles) respectively. Therefore, the generalized coordinates of the robot are completely defined by:

$$\mathbf{q} = (\mathbf{q^b}, \mathbf{q^j}), \tag{2}$$

where $\mathbf{q^j}$ is defined by the robot joint angles:

$$\mathbf{q^j} = [q^L_{hr}, q^L_{hy}, q^L_{hp}, q^L_k, q^L_{ss}, q^L_t, q^L_{hs}, q^L_{tp}, q^L_{tr} q^L_{sr}, q^L_{sp}, q^L_{sy}, q^L_e, q^R_{hr}, q^R_{hy}, q^R_{hp}, q^R_k, q^R_{ss}, q^R_t, q^R_{hs}, q^R_{tp}, q^R_{tr}, q^R_{sr}, q^R_{sp}, q^R_{sy}, q^R_e]^T.$$

In this paper, we denote v_x, v_y, v_z as the torso velocity, $\dot{\psi}, \dot{\theta}, \dot{\phi}$ as the torso angular velocity about the roll, pitch and yaw axes, and the error as $e_\square = \square^d - \square$, where $\square^d$ is the desired value for that state.

1) Forward Kinematics: Given the generalized coordinates of the robot $\mathbf{q}$, forward kinematics (FK) can be used to compute the homogeneous transformation matrix $\mathbf{T} \in \mathbb{R}^{4\times4}$ of the robot's end-effector and center of mass (CoM). Several open-source packages can solve the FK by using the URDF model of the robot. We created a URDF of Digit from the XML model provided by the Agility Robotics, and used FROST [2] to obtain the symbolic expressions for $\mathbf{T}$.

For any homogeneous transformation:

$$\mathbf{T}_{\text{ac}} = \begin{bmatrix} \mathbf{R}_{\text{ac}} & \mathbf{p}_{\text{ac}} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix}, \tag{3}$$

where a and c denote any two frames of interest, $\mathbf{R}_{\text{ac}} \in \mathbb{R}^{3\times3}$ represent the rotation matrix, and $\mathbf{p}_{\text{ac}} \in \mathbb{R}^{3\times1}$ represents the relative position of the origin of frame c with respect to the origin of frame a. The orientation of the robot's feet with respect to the world is given through:

$$\mathbf{R}^{L/R}_{\text{wf}} = \mathbf{R}^{L/R}_{\text{wb}} \mathbf{R}^{L/R}_{\text{bf}}, \tag{4}$$

where w corresponds to the world fixed frame, f and b correspond to the robot's feet and base body frames, and L/R determines left or right side.

By using the FK described above, we can use the orientation of the stance foot to estimate the support plane roll (γ) and pitch (α) angles of the walking terrain by converting the rotation matrix $\mathbf{R}^{L/R}_{\text{wf}}$ to Euler angles.

2) Inverse Kinematics: We only consider the foot position with respect to the robot base to solve the IK problem. In addition, given the particular closed-chains structure of Digit's leg, we keep the yaw hip angle constant and use simple trigonometric computations to transform the foot Cartesian position into virtual leg length, pitch angle, and roll angle. The virtual leg is the imaginary line that connects the hip of the robot with the leg ankle. In this decoupled system, the virtual leg length and pitch angle are determined by the position of the hip pitch and knee joint. With these values, we then solve the "reduced" IK subject to the constraints imposed by the leg kinematic structure. Finally, by solving the IK problem for a sufficiently diverse set of desired foot positions, we obtain a closed form solution using nonlinear regression.

This work is supported by PMRF

[1]Cyber-Physical Systems, Indian Institute of Science (IISc), Bengaluru. {manantayal, shishirk}@iisc.ac.in .

3) Contact detection: To switch between the left and right stance during the walking gait, we use the contact information of the feet with the ground. To detect the contact event, we estimate the ground reaction force of the stance foot by using the spring deflection and the contact Jacobian of the stance foot, as shown in [3].

III. Method

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.

A. Using Arm Swing while Walking

In this section, we describe the control framework, and explain how the end-foot trajectories are modulated and tracked in real-time.

B. Overview of the Control Framework

We use a hierarchical structure with a high-level foot trajectory modulator and a low-level gait controller. The foot trajectory modulator comprises a linear policy that modulates a parameterized semi-elliptical foot trajectory. The trajectory thus generated is then fed into the gait controller, which uses a regulation based on the contact state of each of the legs. We chose to append an ankle regulation with the required joint targets obtained from the foot trajectory through inverse kinematics for the swing leg. If a leg is in the stance phase, we keep the ankle passive and activate the torso regulation to maintain the upper body attitude. Effectively, the policy is free to control the swing leg, whereas the stance leg is only used to stabilize the robot base. Since the stance ankle is passive and aligns the foot parallel to the support plane, we can accurately estimate the ground elevation using the forward kinematics. Ignoring the short-lived double support phase in every walking cycle, we only consider the single support phases and estimate the terrain for every walking step as discussed in section II-A.1. This estimate and the robot's torso states are used by the policy to modulate the properties of the swing leg trajectory.

C. Foot Trajectory Modulator

To modulate the foot trajectory, we propose to train a policy that uses only the relevant feedback deduced through physical insights from walking motion. For the given state space $\mathcal{S} \subset \mathbb{R}^n$ of dimension n, and action space $\mathcal{A} \subset \mathbb{R}^m$ of dimension m, we define our policy to be $\pi : \mathcal{S} \rightarrow \mathcal{A}$ as $\pi(s) := Ms$, where $M \in \mathbb{R}^{m \times n}$ is a matrix of learnable parameters. Our formulation drastically decreased the required control complexity, and hence a linear policy was sufficient to learn such a transformation. The observation and the action space choices are explained as follows.

Observation Space: In our prior work [4], we demonstrated the effectiveness of choosing a reduced observation space from all available robot states. However, to improve the terrain complexity that the policy could handle and develop command-controlled policies, we augmented the torso velocity error and the desired heading velocity to the observation space in [4], including the torso orientation, torso angular velocity, and support plane elevation. For heading direction control, we choose to send the error in heading yaw in the place of the current torso yaw. Thus, the observation space is a 12 dimensional state vector defined as $s_t = \{\psi, \theta, e_\phi, \dot{\psi}, \dot{\theta}, \dot{\phi}, \gamma, \alpha, e_{v_x}, e_{v_y}, e_{v_z}, v_x^d\}$,

Action Space: The semi-elliptical foot trajectory, is parameterized by the major axis (Step Length, ℓ), the orientation of the ellipse along the Z-axis of the hip frame (hip yaw, φ) and translational shifts along X, Y and Z directions, $\acute{x}, \acute{y}, \acute{z}$ (together shown as $\acute{O}$). Here, step length and hip yaw describe the walking motion, whereas the shifts are heavily utilized to balance the robot actively. The semi-ellipse is then generated in the hip frame of reference as shown in [4]. To preserve the symmetry of the trajectories, we remove all the asymmetric conventions between the legs, outside the policy and apply a mirrored transformation according to the leg. This enables us to learn to predict a single set of parameters irrespective of the leg. Thus, the action space is a 5-dimensional vector such that, a_t=$\{\ell, \varphi, \acute{x}, \acute{y}, \acute{z}\}$.

D. Gait Controller

The gait controller is responsible for keeping track of the gait parameters and tracking the generated foot trajectory. Based on the contact state of the leg (estimated as explained in Section II-A.3), the gait controller augments the ankle regulation followed by joint level PD tracking for the swing leg and just the torso regulation for the stance leg, as in [5]. A phase-variable $\tau, \tau \in [0, 1)$ which is used to track the semi-elliptical trajectory gets reset once every walking step or upon a premature foot contact. Hence for an ideal walking cycle, the phase variable iterates from 0 to 1 twice.

Ankle Regulation: Since the generated foot trajectory does not include the two DoF at the ankle (actuated joints), we require an explicit regulation to control the foot orientation. For bipeds, control of the foot is crucial as the swing leg's angle of attack directly affects the torso orientation. The swing foot is kept parallel to the underlying terrain elevation to ensure proper landing on the ground. The desired position of the swing foot is determined from the kinematics of the robot's leg [5] as follows:

$$q_{tr}^d = q_{hr} + S_f(0.366 + \psi + \gamma) \tag{5}$$

$$q_{tp}^d = q_{hp} + S_f(0.065 - \theta - \alpha), \tag{6}$$

where q_{tr}^d and q_{tp}^d are the target angles for the ankle roll and pitch joints. The value of S_f is defined as follows,

$$S_f = \begin{cases} -1 & \text{left leg in swing phase} \\ +1 & \text{right leg in swing phase} \end{cases}$$

Torso Regulation: The torso regulation is applied to ensure an upright torso, which is desired for a stable walking gait and, more importantly, to prevent the stance leg from sliding. The robot is assumed to have a rigid-body torso, and hence simple PD controllers defined below can be used

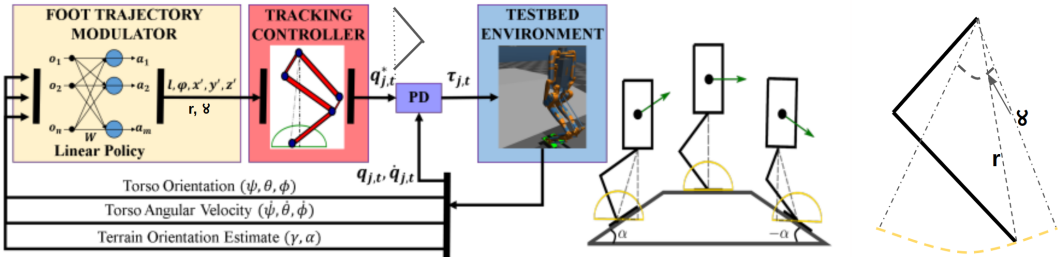

Fig. 1: Control framework (Walking)

for the the attitude control.

$$u_{hr} = P_{hr}(\psi_d - \psi) + D_{hr}(\dot{\psi_d} - \dot{\psi}) \tag{7}$$

$$u_{hp} = -S_f(P_{hp}(\theta_d - \theta) + D_{hp}(\dot{\theta_d} - \dot{\theta})), \tag{8}$$

where u_{hr}, u_{hp} are the torques applied the the hip roll and pitch of the stance leg and $P_{hr}, D_{hr}, P_{hp}, D_{hp}$ are hand tuned gains. The desired targets for the torso attitude $(\psi_d, \theta_d, \dot{\psi_d}, \dot{\theta_d})$ are all set to zero for normal walking.

E. Development of Heuristics

Being simple and interpretable, the linear policy allows us to manipulate the matrix elements based on physical insights. In [4] we developed a set of heuristic linear equations as a sub-optimal policy, hand-tuned the gains and deployed it as the initial policy for the training. This technique resulted in the policy converging to practical walking motions across different slopes. However, a lack of structure in the policy leads to the training algorithm making undesirable relations between the state and action variables. For example, the feedback of torso pitch (θ) or yaw (ϕ) is unnecessary for y-shift ($\acute{y}$), as it cannot control those DoF. The effect of these non-sparse terms in the matrix was insignificant in simulation but got amplified in our hardware trials, leading to the policy's failure. We hypothesize that the policy overfitting to the simulation dynamics through these non-zero stray terms affects the hardware performance due to the domain shift. To resolve this issue, we enforce a structure to the sparse matrix and only learn for the relevant terms. In this work, we intend to train separate policies for walking i) on arbitrary slopes, ii) on stairs, and iii) asper commands. Hence, we select certain terms common across all these matrices to ensure dynamic balance and several unique terms for each of them based on their task requirements.

Stabilization Heuristics: Irrespective of the task at hand, we require any policy to keep balance and walk forward. To this end, we define the following heuristic relations to stabilize the robot in each of the following planes.

In the sagittal plane,

- ℓ is to be used for correcting the disturbance in θ,
- $\acute{x}$ is to be used for correcting the error in v_x, i.e. e_{v_x},
- $\acute{z}$ is to be used for minimizing the torso oscillation along the z-direction, i.e. e_{v_z}

In the transverse plane,

- $\acute{y}$ is to be used for correcting the disturbance in ψ and the error in v_y, i.e. e_{v_y}

In the coronal plane,

- φ is to be used for correcting the error in heading direction, i.e. e_ϕ

Task-Specific Heuristics: Apart from the stabilization heuristics, we add additional terms to the policy matrix based on the nature of the task for each of the following cases,

Arbitrary Slope Policies: In this case, there should be a dependency of the actions $\acute{x}$ and $\acute{z}$ with the support plane estimates (γ, α), to alter the foot placements in the sagittal plane based on the underlying terrain. Deducing a feasible target velocity for an arbitrary terrain is not straightforward, and we are also not keen on velocity tracking compared to stable walking on this challenging terrain. Hence, we relate the action ℓ with the state e_{v_x}, expecting the policy to converge to nominal walking step size in accordance with the objective (refer Section III-F).

Command Controlled Policies: To learn a command controlled policy, we keep the same setup as for arbitrary slopes except for the step length (ℓ) to be related with the commanded heading velocity (v_x^d) directly.

Stair Policies: The primary strategy to walk on stairs blindly are i) have a high swing height and ii) increase the z-shift upon accidental stubbing with a step. For the first strategy, we explicitly choose a higher foot clearance. To incorporate the second strategy, we enable the term connecting the state e_{v_x} with the action $\acute{z}$. The intuition here is that when a foot collides with a step, a sudden change in the v_x can be observed, and the feedback from e_{v_x} can result in an increase in the $\acute{z}$.

F. Policy Training

In this section, we discuss the training procedure used for learning the linear policy. We start from a hand-tuned intial policy to provide a warm start for the training. We use Augmented Random Search (ARS) [6], owing to the minimal number of hyper-parameters to tune, ease of use, and its effectiveness towards solving continuous-control problems. A point worth noting is that, instead of using the generic ARS setup, where the search space is in $\mathbb{R}^{m \times n}$, having enforced a heuristic structure to the policy matrix, we only search a sub-space of this parameter space.

G. Reward Function:

Due to the ambiguity in finding a feasible target velocity for a given terrain type, we propose two different reward

functions for training the i) Terrain Policies and ii) Command Controlled Policies. For terrain policies (slope and stair policies), we use a reward function defined as,

$$r = G_{w_1}(\psi) + G_{w_2}(\theta) + G_{w_3}(e_\phi) + G_{w_4}(e_{p_z}) + W\Delta_x \tag{9}$$

where, e_{p_z} is the error in the robot's height, and Δ_x is the distance travelled along the heading direction in that time-step, weighted by W. The mapping $G : \mathbb{R} \to [0, 1]$ is the Gaussian kernel given by $G_{w_j}(x) = \exp(-w_j * x^2)$, $w_j > 0$. The objective here is to walk as far as possible while ensuring the stability of the torso. For training the command controlled policies, we remove the Δ_x term and substitute it with a velocity tracking term as shown in (10). This is because, we require the policy to learn to react to changes in the velocity commands.

$$r = G_{w_1}(\psi) + G_{w_2}(\theta) + G_{w_3}(e_\phi) + G_{w_4}(e_{p_z}) + G_{w_5}(e_v) \tag{10}$$

where e_v is the error in the heading velocity of the robot.

H. Training Setup

For terrain policies, we train on the variants of a given parameterized terrain type. A specific combination of terrain parameters is randomly chosen from a discrete set of that terrain's configurations at the beginning of an episode. The target heading velocity is kept to be a small positive value to prevent the policy from learning to walk in place (as $e_{v_x} \neq 0$). For the command-controlled policies, we only train on flat-ground and update the target velocity and desired yaw every three seconds. An episode is terminated when the robot topples, or if the robot's height decreases below a certain threshold or if the maximum episode length is reached.The ARS hyperparameters used for training are learning rate (β) = 0.03, noise (ν) = 0.04 and episode length = 15k simulation steps.

Swinging the arms helps in reducing the angular momentum of the body and reduces the disturbance in the torso. So to show that we will compare two cases: Digit robot walking with arm swing and without arm swing using Linear Policy as mentioned in [7]. For that we add 2 more rows in the policy for the radius of swing (r) and maximum swing angle (γ). The final control framework is shown in Fig. 1.

I. Using Arm Swing while Hopping

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 [7], 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.

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.

Now, in this case, we will compare two cases: Digit robot hopping with arm swing and without arm swing using Linear Policy as mentioned in [8]. In this case also we add 2 more rows in the policy for the radius of swing (r) and maximum swing angle (γ). The final control framework is shown in Fig. 2

J. Using Arm Swing for Push Recovery while Standing

To analyse the utility of arm swing in case of push recovery, we will compare the case of push recovery with active, passive and fixed arm movements. We will use LQR Controller to recover from external push on hip, knee and ankle joints in all the cases. In case of Active arm Linear MPC is used to control the arm movement, while keeping the control values as zero in case of passive arm and fixed control values (set by the initial control values) in case of fixed arms.

IV. Simulations

A. Using Arm Swing while Walking

The inferences from the experiment comparing the cases with and without arms (figs: 3 & 4) are as follows:

- Adding an Arm Swing in Stabilizing the Torso better
- It also helped in Tracking the velocity better
- It also helped in Walking at higher speeds
- Moreover, Swing the arms helped in reducing the CoT (Cost of Transmission) by 24%

B. Using Arm Swing while Hoping

The inferences from the experiment comparing the cases with and without arms (figs: 5, 6 & 7) are as follows:

- Adding an Arm Swing in Stabilizing the Torso better
- It also helped in Tracking the velocity better
- It also helped in Hopping at higher speeds
- However, the CoT (Cost of Transmission) in both the cases are almost similar
- The height of hop increases with increase in velocity

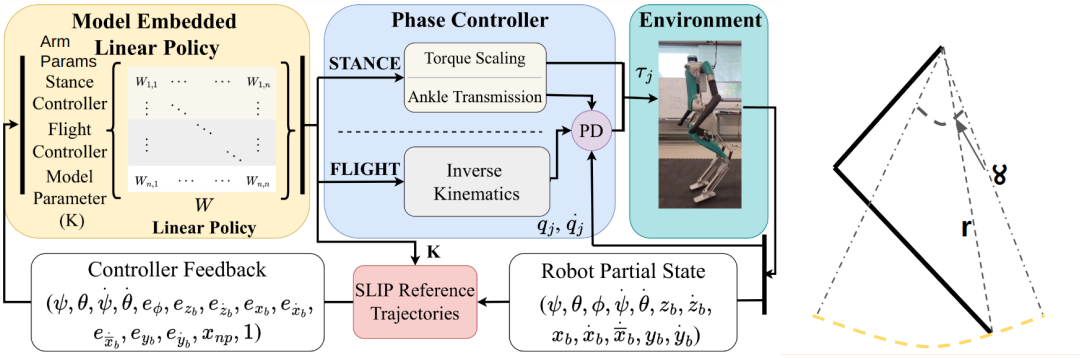

Fig. 2: Figure showing the control framework (Hopping)

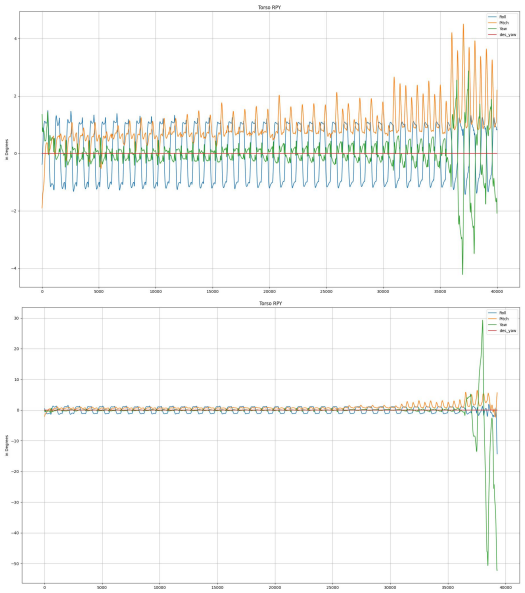

Fig. 3: Figure showing the RPY with (left) and w/o Arm swing (right) while Walking)

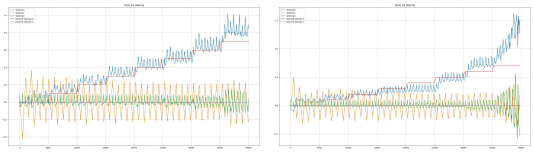

Fig. 4: Figure showing the Velocity Tracking with (left) and w/o Arm swing (right) while Walking

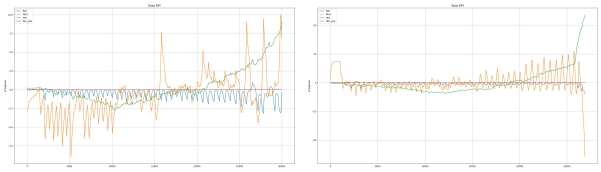

Fig. 5: Figure showing the RPY with (left) and w/o Arm swing (right) while Hopping

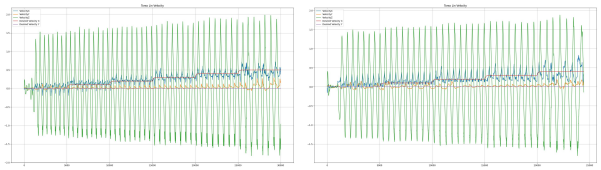

Fig. 6: Figure showing the Velocity Tracking with (left) and w/o Arm swing (right) while Hopping

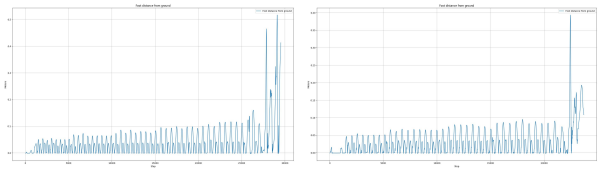

Fig. 7: Figure showing the Foot Distance from ground with (left) and w/o Arm swing (right) while Hopping

C. *Using Arm Swing for Push Recovery while Standing*

The inferences from the experiment comparing the cases active, fixed and passive arms (figs: 8) are as follows:

- Active arm has the least CoM deviation to push.
- The Power consumption in case of active arms is 4.7% less than fixed arms and 129% less than passive arm.
- In active arm case the oscillation of the torso after external push is least.

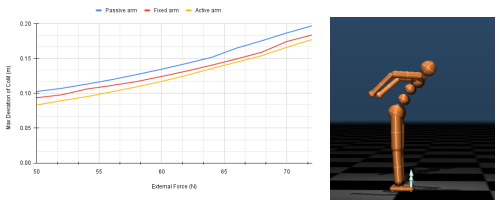

Fig. 8: Figure showing the max deviation of torso vs external perturbation(left)

V. Conclusions

In this paper, we successfully demonstrated stabilizing the walking, hoping and push recovery in the bipedal robot Digit in simulations.

REFERENCES

[1] S. M. Bruijn, O. G. Meijer, P. J. Beek, and J. H. Van Dieën, "The effects of arm swing on human gait stability," in *The Journal of experimental biology, 213, 3945–3952.*, 2010.

[2] A. Hereid and A. D. Ames, "Frost*: Fast robot optimization and simulation toolkit," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 719–726.

[3] Y. Gong, R. Hartley, X. Da, A. Hereid, O. Harib, J.-K. Huang, and J. Grizzle, "Feedback control of a cassie bipedal robot: Walking, standing, and riding a segway," in *2019 American Control Conference (ACC)*, 2019, pp. 4559–4566.

[4] L. Krishna, U. A. Mishra, G. A. Castillo, A. Hereid, and S. Kolathaya, "Learning linear policies for robust bipedal locomotion on terrains with varying slopes," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 5136–5141.

[5] G. A. Castillo, B. Weng, W. Zhang, and A. Hereid, "Robust feedback motion policy design using reinforcement learning on a 3d digit bipedal robot," 2021.

[6] H. Mania, A. Guy, and B. Recht, "Simple random search of static linear policies is competitive for reinforcement learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 1800–1809.

[7] L. Krishna, G. A. Castillo, U. A. Mishra, A. Hereid, and S. Kolathaya, "Linear policies are sufficient to realize robust bipedal walking on challenging terrains," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2047–2054, 2022.

[8] R. Soni, G. A. Castillo, L. Krishna, A. Hereid, and S. Kolathaya, "Melp: Model embedded linear policies for robust bipedal hopping," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 10 418–10 424.