Realizing Linear Controllers for Quadruped Robots on Planetary Terrains

Aditya Shirwatkar*
Somnath Kumar*
Shishir Kolathaya
Indian Institute of Science
Bengaluru, India

Shamrao Garur
U R Rao Satellite Centre
Bengaluru, India

Vinod Kumar
Indian National Space Promotion and Authorization Center
Bengaluru, India

ABSTRACT

Until now, planetary exploration has been accomplished with wheeled vehicles, making movement in highly complex, sandy, and sloping terrain incredibly tough. On the other hand, legged robots have come a long way in the last decade and have reached a stage of development where practical applications appear to be possible. To collect critical scientific data, legged robots can overcome wheeled vehicles' difficulties when exploring harsh environments like impact craters. As a result, there is a need to develop simple, stable walking controllers given the limited power resources and reserve maximum onboard computing for scientific equipment while exploring such regions. This work proposes a walking controller for legged robots that is computationally efficient at runtime for traversing planetary terrains. We implement this walking controller on our custom-built quadruped, using learned linear feedback policies that modulate the end-foot trajectories. The proposed walking controller can traverse various planetary terrains such as flat, sloped, rugged, loose, and lower-than-Earth gravity conditions in simulation environments. Our controller outperforms the baseline open-loop controller on planetary landscapes by reducing slippage and increasing stability. We have also provided preliminary hardware testing results of our controller. In addition, video results can be found at: https://youtu.be/La3y-xhWm1U

CCS CONCEPTS

• **Computer systems organization** → **Robotic control**; • **Computing methodologies** → **Randomized search**; **Reinforcement learning**.

KEYWORDS

Quadrupedal Walking, Linear Policy, Planetary Terrains

ACM Reference Format:
Aditya Shirwatkar, Somnath Kumar, Shishir Kolathaya, Shamrao Garur, and Vinod Kumar. 2023. Realizing Linear Controllers for Quadruped Robots on Planetary Terrains. In *Advances In Robotics - 6th International Conference of The Robotics Society (AIR 2023), July 05–08, 2023, Ropar, India.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3610419.3610572

*Both authors contributed equally to this research.

AIR 2023, July 05–08, 2023, Ropar, India

ACM ISBN 978-1-4503-9980-7/23/07…$15.00
https://doi.org/10.1145/3610419.3610572

1 INTRODUCTION

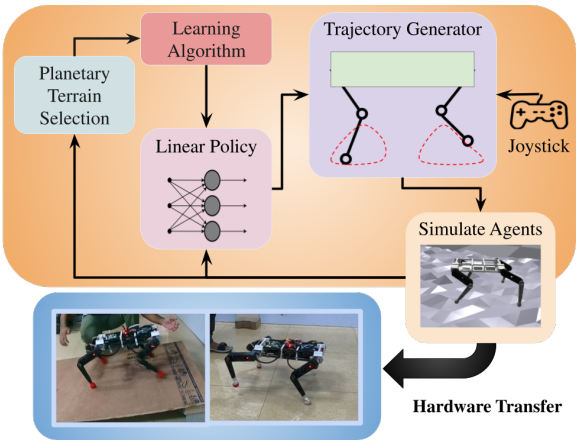

Figure 1: Overview of Complete Pipeline

Future robotic exploration missions into our solar system will involve navigation over a diverse array of terrains including those seen on craters or along cliffs. These types of habitats provide vital information about the planet's geological history. Due to the unpredictability of the encountered soil and the requirement to traverse mountainous terrains, it is impossible for classical wheel-based rovers to navigate over these geological locations. Due to the recent advancement in locomotion with quadrupedal walking robots [4, 8, 12], there is a growing interest to deploy leg-based rovers for planetary exploration.

Over the decades, many methods have been proposed for realizing robust-legged locomotion on unstructured terrains. Classical works such as the spring-loaded inverted pendulum (SLIP) with Raibert's heuristic controller [16] and the Zero Moment Point (ZMP) [5] approach have been used to generate robust walking behaviors. Convex Model Predictive Control (MPC) [6] approach has shown dynamic motion behaviors using only the centroidal dynamics model. Nevertheless, these optimization-based methods suffer from the drawback of requiring heavy onboard compute resources and relying on knowing the system's dynamics model.

Concurrently, due to the advances in deep learning techniques, development of legged locomotion controllers based on Reinforcement Learning (RL) have made significant progress over the decade. RL combined with the centroidal dynamics model of quadruped has been shown to solve stepping-stone locomotion, two-legged in-place balance, and balance beam locomotion with a good sim-to-real transfer [21]. There have also been approaches that use Deep

RL policies to parameterize end-foot trajectories [12]. Although these have been shown to work in challenging outdoor terrains, the policy requires thousands of parameters for inference and millions of samples to train. This makes the overall training and deployment process computationally expensive

It is worth mentioning that work on legged rovers for planetary exploration is not a recent development, and robots like ATHLETE [20], Scorpian [7], SpaceClimber [2], and SpaceBok [1] have been explicitly developed for such use cases. ATHLETE, Scorpian, and SpaceClimber offer good stability while traversing rough terrain but have limitations on hardware that acts as a bottleneck for traversability, speed and agility. SpaceBok, on the other hand, offers excellent dynamic maneuvers, which have been shown to work in low gravity conditions [17] and sandy terrains [9]. However, considering the limited resources available for space exploration missions, having computationally efficient and robust controllers is critical, as most of the compute should be accessible to the scientific equipment. Linear feedback policies have shown promising alternatives to the classical model-based and Deep RL approaches [13, 15]. These linear policy-based approaches allow one to realize an efficient and robust controller for walking robots.

Motivated by these findings, we propose a control framework consisting of linear feedback policies to generate walking trajectories on flat, sloped, rough, loose (soiled/sandy), and lower-than-Earth gravity environments such as Mars and Moon. The following points summarizes our main contributions in this work:

1. Learning linear policies to control a highly non-linear system such as a quadruped in various planetary terrains and gravity conditions. Our approach significantly differs from [9] and [17] as we use neither Deep Neural Network (DNN) policy nor optimization-based control methods. This allows us to not only have computationally efficiency, but also an intuitively interpretable controller.
2. Large-scale approximate terramechanics subroutine for quadruped robots in simulation. To the best of our knowledge, there is no large-scale real-time solution for terramechanics simulation which can be readily integrated with learning algorithm pipelines. As a result, we utilized Nvidia's IsaacGym [10] in conjunction with a Bekker model's subroutine [3] to mimic sand-soil contact interactions. It is worth mentioning that we did not attempt to focus on sinkage but rather on slippage effects.
3. Preliminary hardware testing to showcase the capabilities of our controller on loose soil. We specifically demonstrate hardware transfer of the learned linear policy, showing an improvement over open-loop controllers in terms of robustness.

The paper is organized as follows: Section 2 will describe the robot model, notations, and hardware specifications. Section 3 will describe the linear policy, the walking controller, and details on the terramechanics subroutine. Section 4 will describe the training and evaluation methods used. Section 5 will provide simulation results, assessments, comparison with the baseline open-loop controller, and preliminary hardware tests. Finally, Section 6 will provide conclusions.

2 ROBOT DESCRIPTION

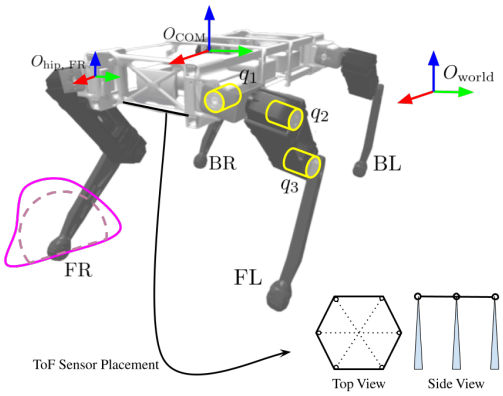

Figure 2: Figure showing quadruped's kinematic description where dashed curve shows the ideal end-foot trajectory generated by the open-loop controller, and bold curve in pink shows the modulation of the ideal trajectory by the linear policy

This section will briefly explain the quadruped robot used, and the accompanying actuator-sensor, terrain orientation estimation, and kinematics framework we intend to employ for locomotion. The quadruped, as shown in Fig. 2, is a custom-built low-cost walking robot developed for rapid prototyping of learning-based controllers. The symbols FL, FR, BL, BR, represent the front-left, front-right, back-left, and back-right legs respectively. Also, $O_{\text{world}}, O_{\text{COM}} \in \mathbb{R}^3$ represent the world and Centre of Mass (COM) frames respectively. For simplicity, only the FR frame at the hip joint $O_{\text{hip, FR}} \in \mathbb{R}^3$ is shown. It is worth noting that, unlike SpaceBok [1], the quadruped is not explicitly built for planetary exploration applications, as in this work, we want to focus on building the model-free control framework that is easily transferable to any quadruped system.

Actuation: Overall, the robot model consists of 6 floating and 12 actuated Degrees of Freedom (DoF). The 12 DoF are actuated using B3M Smart Servo motors. The motors can measure joint angles (hip and knee) and torques via joint encoders and motor current sensors. The stall torque provided by the motors is in the range of 4.1 N m.

Onboard Computation: We use a Raspberry PI 3b microcomputer for running all the high-level controllers. It consists of 2 GB RAM and includes four high-performance ARM Cortex-A53 processing cores running at 1.2 GHz. An STM32 microcontroller board is also used for low-level communication between a Raspberry Pi and the servo motors using Serial Peripheral Interface (SPI) communication.

Sensors: Accurate feedback is necessary for any controller to drive the system to the desired state. For this, we use an Xsens MTi-610 Inertial Measurement Unit (IMU), which provides calibrated data on the 3D orientation, angular velocities, acceleration, and magnetic field.

Local Terrain Slope Estimation: We require the local terrain slope as feedback to the controller. Hence we use a similar technique as [13] to estimate it with the help of joint encoders in the motors and six Time of Flight (ToF) sensors placed in a hexagonal

configuration below the torso as shown in Fig. 2. Note ToF sensors are only used to estimate the readings on hardware as the current version of IsaacGym does not support range sensors. This ToF sensor placement is required as currently, we lack any foot force sensing capabilities for contact detection and thus are planned for the future versions of the robot.

Kinematics: We treat each leg independently to derive analytical relations for forward and inverse kinematics. Here q_1, q_2, and q_3 represent abduction, hip, and knee joints and form a serial-3R kinematic chain as shown in Fig. 2. This reduces the runtime overhead present in the iterative solvers, as it is one of the critical components that help us map end-foot trajectories to joint space.

3 METHODOLOGY

This section provides an overview of the control architecture, i.e., how end-foot trajectories are generated and tracked in real-time to walk in various environments. This structure is also depicted graphically in Fig. 1.

3.1 Reinforcement Learning

Quadrupedal locomotion in this work is treated as an RL problem. We parameterize our RL policy with end-foot trajectories with sinusoidal height variation to accelerate training. Our method is similar to [13, 15], which uses a feedback mechanism to dynamically alter these trajectories based on body and local terrain slope. At a high level, the RL policy infers the parameters of the walking trajectories and, therefore, robot motion. We limit our policy to being linear, as it then requires low computation, allowing our policy to be executed on an onboard embedded system in real-time. Section 4 contains further information on the training algorithm used to learn these linear policies.

Observation Space: The observation space, which is the input to the policy, is in $\mathbb{R}^{18\times 1}$ in our formulation. It consists of robot walking height h, body orientation in roll α, pitch β, yaw γ, the position of foot $r_i \in \mathbb{R}^3$ for each leg $i \in \{\text{FL, FR, BL, BR}\}$ with respect to the center of mass (COM), local terrain slope in roll α_s and pitch β_s.

Action Space: The action space, i.e., the output of the policy, is in $\mathbb{R}^{8\times 1}$, which represents the instantaneous shifts. The shifts $\xi_i \in \mathbb{R}^3$ are x, y, z translational transforms for the leg trajectories of the robot in the body frame. These instantaneous shifts result in reactive behavior, as seen in Fig. 2.

Linear Policy: We choose the policy to be $\pi(\mathbf{s}) := M(\Theta)\mathbf{s}$, where $M \in \mathbb{R}^{8\times 18}$, is a matrix that maps the observations $\mathbf{s}$ to actions, and Θ represents the learnable parameters, i.e., the policy matrix elements. To simplify the problem, we consider M to be a sparse matrix by accounting for the intuitive contribution of each element in the observation space to the elements of the action space. For example, the shift in x for the leg is only affected by the body height, body pitch, slope pitch, and x coordinate of the legs. So the column element for the respective terms will be learned, with the rest always being zero. Such structures also showcase one of the advantages of having linear feedback policies as they are easier to analyze and impose intuitive heuristics than DNNs.

3.2 Walking Controller

Walking over varied terrains such as flat, inclined, rough, and loose brings unique problems that cannot be acquired straight from the open-loop walking controller (our framework without the linear policy feedback). As mentioned earlier, we use the linear policy to alter the end-foot trajectories in real-time based on the observations to ensure steady walking. For the scope of this work, we focused on mainly realizing two types of gait behaviors: trot and crawl.

The trajectory generator takes the desired linear velocity command $v_d \in \mathbb{R}^2$ from the joystick along with gait parameters (gait type, swing time T_{sw}, stance time T_{st}, swing height h_{sw}, walking height h_0) to calculate the touch-down point of the swing leg. The gait parameters are chosen according to the literature. For simplicity, we will show the trajectory generation for a single i^{th} leg ($i \in \{\text{FL, FR, BL, BR}\}$), which can be extended to other legs easily. The foot placement for the swing leg is determined using Raibert's heuristic [16], and the instantaneous shifts ξ_i as follows,

$$\rho_{xy} = \frac{v_d T_{st}}{2} + \xi_{i,xy}, \tag{1}$$

where $\rho_{xy} \in \mathbb{R}^2$ is the desired step location on ground plane, $\xi_{i,xy} \in \mathbb{R}^2$ are the x, y components of the shifts for the i^{th} leg. From the classical viewpoint, shifts are similar to the capture-point feedback term [14] used for push recovery. However, this capture point requires certain model assumptions and only depends on velocity feedback, which may not always hold in all scenarios; hence the motivation to learn the shifts that depend on multiple robot states.

The leg trajectories for every time step are generated as follows,

$$\begin{aligned}
\Delta r_{xy} &= \begin{cases} \frac{\rho_{xy} - r_{xy}}{T_{sw}} \frac{\pi}{\pi - \phi}\, \Delta t & \text{, when in swing} \\ -v_d\, \Delta t & \text{, when in stance} \end{cases} \\
r_{xy,d} &= r_{xy} + \Delta r_{xy} \\
r_{z,d} &= \begin{cases} h_{sw} \sin\phi + h_0 + \xi_{i,z} & \text{, when in swing} \\ h_0 + \xi_z^i & \text{, when in stance} \end{cases}
\end{aligned} \tag{2}$$

where $\phi \in [0, \pi]$ is the i^{th} leg phase indicating the percentage completion of swing time, Δt is the control time-step, $\xi_{i,z}$ is z component of the shift for the i^{th} leg, r_{xy} represent the x, y components of the current foot position, and $r_{xy,d}$ and $r_{z,d}$ represent the desired foot positions for the x, y components and z component respectively.

Thus the entire leg trajectory gets modulated by the predicted action values of the linear policy. We choose to have a decoupled structure in x, y and z for the end foot trajectory because it allows us to take twist inputs and instantaneous shifts seamlessly from the joystick and the policy respectively. In theory, the z component variation can consist of any function that generates a swinging profile; however, we used sinusoidal variation to fulfill our requirements. An inverse kinematics function then gives the command joint angles for each leg described earlier.

3.3 Details of Terramechanics Subroutine

We need controllers that do not destabilize the quadruped while walking over loose terrains made up of soil and sand. Thus, policies must be explicitly trained in such environments for seamless sim-to-real transfer. Current physics simulators like Chrono [19]

and Adams-Based Rover Terramechanics and Mobility Simulator (ARTEMIS) [22] lack the ability to perform real-time computations required for fast realization of learning-based controllers. A simple alternative is to create an approximate terramechanics subroutine of the Bekker model [3] in an existing fast and large scale simulator for Robot Learning such as Nvidia IsaacGym [10]. We only focus on the shear forces acting at the contact, which causes slippage in the total distance traveled. To account for the effects of high sinkage is beyond the scope of this work; hence, it is left for future works.

Shear Stress at the contact: Consider the scenario shown in Fig. 4 where the contact of the leg produces a vertical load W, with a being the projected contact area of cross-section AB. For simplicity, we neglect the effects due to the curvature of the leg and treat the contact area as a flat surface. Then the pressure P, produced at the contact surface on the soil, is given by $P = W/a$. This motion creates shear stress at the cross-section AB. The Mohr-Coulomb failure criterion estimates the soil failure, which is given by, $\tau_{max} = C + P \tan(\phi_c)$, where τ_{max} is the failure stress, C is the soil cohesion, and ϕ_c is the angle of friction. The actual shear stress generated is then derived as, $\tau = \tau_{max}\ (1 - e^{-j/G})$ and $F_s = \tau\ a$. Here τ is the actual shear stress generated at the contact surface, j is the shear displacement, and G is the shear modulus. Once τ is calculated at the contact surface, one can say that a shear force F_s is acting on the body. This force then induces a slippage at contact, which means that the actual velocity of the body is reduced as it is a reaction force. The above equations are then used to apply an external force at the next time step of the simulation. Finally, different gravity conditions can be simulated by changing the parameters of the physics engine solver.

4 POLICY TRAINING AND EVALUATION

We have used the Augmented Random Search algorithm [11] for policy search. The algorithm is comparable to other model-free RL algorithms when searching linear deterministic policies. Given the problem setup, the algorithm's goal is to determine the parameters Θ of the matrix M that yield the best rewards, which leads to the best locomotion on different planetary terrains. To exploit the parallelization capabilities of Isaac Gym, we implemented a batched version of ARS.

4.1 Domain Randomization

We employ two-stage domain randomization to reduce the sim-to-real gap and produce robust learned policies. The first stage consists of randomizing the robot orientation and spawning height at the start of each episode. This ensures that the policy can recover from unstable configurations. The second stage consists of changing the terrain of locomotion using curriculum learning similar to [13]. Initially, the curriculum consisted of easier terrains like flat ground and lower slope values. After every m iteration of the learning algorithm, where m is a hyper-parameter, the difficulty of terrain is gradually increased by training on higher slope values and rough terrains. This ensures that the policy has seen all the different terrain types. While learning the linear policies, we also randomly sample the soil/sand parameters for every rollout to avoid overfitting. The properties used for our experiment are obtained from [18].

4.2 Reward Function

Obtaining a good reward function is critical to reduce the training time and ensure that the learned policy makes optimal predictions for the walking controller. We choose our reward function R to be,

$$R = K_{w_1,u_1}(\|v_{cmd} - v_{curr}\|) + K_{w_2,u_2}(\alpha - \alpha_s) + K_{w_3,u_3}(\beta - \beta_s) + K_{w_4,u_4}(\|\omega_{x,y}\|) + K_{w_5,u_5}(P_o). \quad (3)$$

In the above equation, the function $K : \mathbb{R} \rightarrow [0, 1]$ is a Gaussian kernel and is given by $K_{w_j,u_j}(x) = w_j\ e^{-u_j x^2}$, where $j \in \{1, 2, 3, 4, 5\}$, w_j and u_j are scalar weights. Whereas, $v_{curr} \in \mathbb{R}^2$ and $\omega_{xy} \in \mathbb{R}^2$ are current 2D linear velocity and current angular velocity in x, y respectively. The reward can be broken into five terms, encouraging the robot to minimize the error in commanded and current linear velocity, maximizing the stability by aligning the body roll and pitch to the local terrain slope, minimizing the variation in current angular velocities, and consuming less power P_o.

We have trained our robot to walk on flat, sloping grounds of up to 15° and rough terrains with an average undulation of 5 cm. The robot was commanded to track a given linear velocity in a fixed direction from emulated joystick inputs with respect to the world frame. The hyper-parameters of ARS used in training were: Learning rate = 0.05, noise = 0.03, Number of directions = 20, Top-performing directions = 4, and an episode length of 500 epochs. Null policy (zero matrices) are initialized and iteratively improved in batches, reducing the computational time required during rollouts.

5 RESULTS

This section contains the simulation results, comparison of the proposed controller with the open-loop walking controller, and the hardware tests performed. As stated earlier, we use Nvidia Isaac Gym with the subroutine mentioned in 3.3 to validate our framework. We made a custom gym environment to train and evaluate the linear policies. Each policy update took on average 12 s, and thus the total training time was about 1 h 30 min. We have learned different policies for different gravity conditions, and the walking controller was commanded to track a velocity of 0.5 m/s for Earth, 0.15 m/s for Mars and Moon. The control loop frequency for the trajectory generation and motor control was 100 Hz, and the simulation timestep was 0.01 s.

As mentioned earlier, the Bekker model subroutine introduces a slip in the total distance traveled. Despite the slip, the policy can keep the robot stable due to the shear forces occurring at the contact foot. Tables **??** and **??** show the average distance traveled and the slip occurring, which should be accounted for by the high-level path planners. As one can see, the linear policy controller outperforms the open-loop controller. The slip is calculated according to the relation $s_l = 100|(d_o - d_l)|/d_o$, where d_l and d_o is the distance travelled on loose and rigid terrains respectively. It is also evident that power consumption is higher as higher torques are required to produce the same motion whenever the robot traverses on loose terrains.

Fig. 5 shows the comparison of the performance of various gaits in different gravity conditions for a simulation time of 15 s. We prefer to use trot gait in higher gravity conditions to track higher velocity commands. In comparison, gaits like crawl perform better

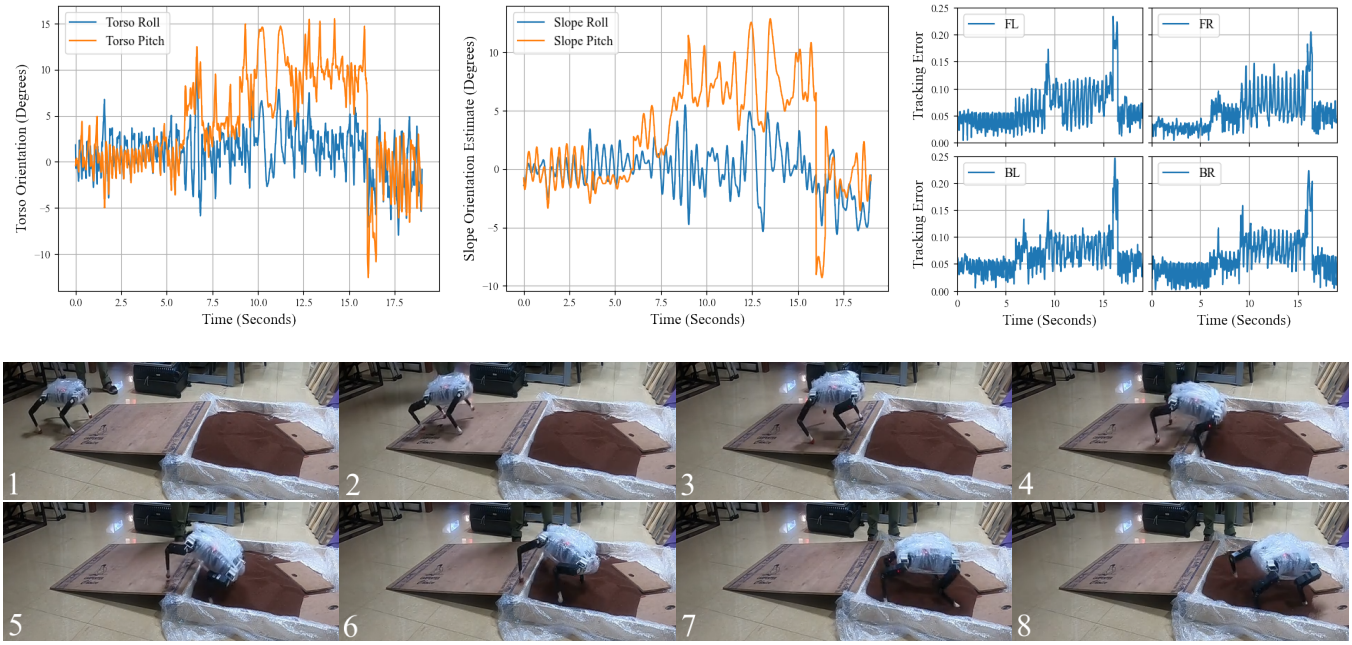

Figure 3: Top figures shown are the plots of variation in torso orientation, slope orientation, and tracking errors of foot trajectory. The bottom figures show the keyframes of the hardware experiment performed on a slope of 9° and loose terrain made of artificial sand.

	Loose Terrain	Distance Travelled (m)	Power Consumed (kW)	Slip (%)
Earth	No	4.991	12.996	56.30
	Yes	2.181	15.765	
Mars	No	1.776	3.292	18.91
	Yes	1.44	3.444	
Moon	No	1.762	2.959	21.66
	Yes	1.38	3.098	

(a)Performance of the linear policy controller in different gravity conditions on flat ground for a simulation time of 15 s

	Loose Terrain	Distance Travelled (m)	Power Consumed (kW)	Slip (%)
Earth	No	5.711	13.867	35.56
	Yes	3.847	15.765	
Mars	No	1.77	3.229	15.19
	Yes	1.501	3.267	
Moon	No	1.842	2.981	17.48
	Yes	1.520	3.151	

(b)Performance of the linear policy controller in different gravity conditions on flat ground for a simulation time of 15 s

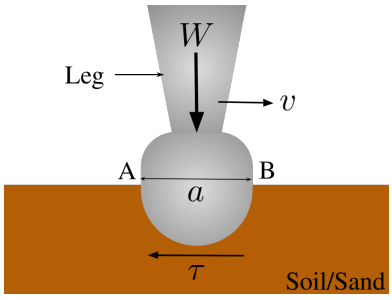

Figure 4: Figure showing an interaction between soiled/sandy terrain and a robot leg

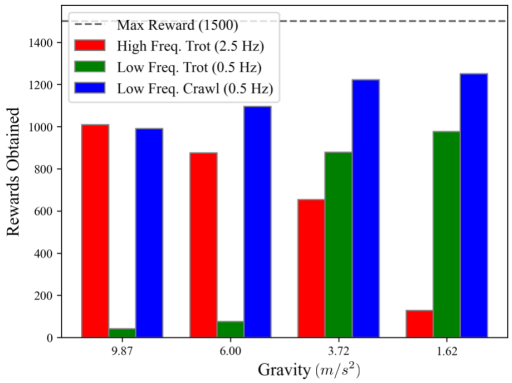

Figure 5: Comparison of different gaits in different gravity conditions for a simulation time of 15 s

in lower gravity conditions. Such performance difference is because higher frequency gaits tend to generate higher impact forces when the leg reaches the touchdown phase. This is evident mainly due to

position control methods employed. We plan to use force control methods in future work since it is possible to lower the impact forces during the touchdown or develop stable bounding behaviors irrespective of the underlying terrain type.

We have also performed preliminary hardware experiments on our proposed controller. Fig. 3 shows the keyframes of the robot traversing slope of 9° along with loose terrain made of artificial dry sand. The robot can keep itself stable when transitioning between different terrains without toppling over. Finally a more diverse set of results can be found at: https://youtu.be/La3y-xhWm1U

6 CONCLUSION

This work successfully showed the development of a linear policy-based walking controller capable of generating robust quadrupedal motion on planetary terrains such as flat, sloped, rugged, loose, and low-than-Earth gravity terrains. The end-foot trajectory modulating policy has been demonstrated to be transferrable across all terrain transitions. The proposed technique will provide a single framework for rapidly constructing linear feedback control policies for any multi-legged robot, thereby significantly simplifying the controller design and deployment process for planetary exploration missions.

Even though nonlinear policy parameterizations might result in better performance, they will have significantly higher computational requirements. Hence linear policies were considered in our framework as they have the smallest number of parameters while showing good performance. The simulation results showed that our method outperforms the baseline open-loop controller by reducing slippage and offering high stability. We also observed that gaits like crawl become more stable with our framework in low gravity conditions. Preliminary results on our hardware platform Stochlite are shown to validate our framework. Future work will include deploying the robot on Lunar and Martian testbeds, performing sinkage analysis, extending the framework to force control methods, and studying more dynamic gaits such as bounding.

REFERENCES

[1] Philip Arm, Radek Zenkl, Patrick Barton, Lars Beglinger, Alex Dietsche, Luca Ferrazzini, Elias Hampp, Jan Hinder, Camille Huber, David Schaufelberger, Felix Schmitt, Benjamin Sun, Boris Stolz, Hendrik Kolvenbach, and Marco Hutter. 2019. SpaceBok: A Dynamic Legged Robot for Space Exploration. In *2019 International Conference on Robotics and Automation (ICRA)*. 6288–6294. https://doi.org/10.1109/ICRA.2019.8794136

[2] Sebastian Bartsch, Timo Birnschein, Florian Cordes, Daniel Kuehn, Peter Kampmann, Jens Hilljegerdes, Steffen Planthaber, Malte Roemmermann, and Frank Kirchner. 2010. SpaceClimber: Development of a Six-Legged Climbing Robot for Space Exploration. In *ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*. 1–8.

[3] M. G. Bekker. 1969. *Introduction to terrain-vehicle systems*. University of Michigan Press.

[4] Gerardo Bledt, Matthew J. Powell, Benjamin Katz, Jared Di Carlo, Patrick M. Wensing, and Sangbae Kim. 2018. MIT Cheetah 3: Design and Control of a Robust, Dynamic Quadruped Robot. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2245–2252. https://doi.org/10.1109/IROS.2018.8593885

[5] Katie Byl, Alexander C. Shkolnik, Sam Prentice, Nicholas Roy, and Russ Tedrake. 2008. Reliable Dynamic Motions for a Stiff Quadruped. In *ISER*.

[6] Jared Di Carlo, Patrick M. Wensing, Benjamin Katz, Gerardo Bledt, and Sangbae Kim. 2018. Dynamic Locomotion in the MIT Cheetah 3 Through Convex Model-Predictive Control. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 1–9. https://doi.org/10.1109/IROS.2018.8594448

[7] Spenneberg Dirk and Kirchner Frank. 2007. The Bio-Inspired SCORPION Robot: Design, Control & Lessons Learned.

[8] Marco Hutter, Christian Gehring, Andreas Lauber, Fabian Günther, Dario Bellicoso, Vassilios Tsounis, Péter Fankhauser, Remo Diethelm, Samuel Bachmann, Michael Blösch, Hendrik Kolvenbach, Marko Bjelonic, Linus Isler, and Konrad Meyer. 2017. ANYmal - toward legged robots for harsh environments. *Advanced Robotics* 31 (2017), 918 – 931.

[9] Hendrik Kolvenbach, Philip Arm, Elias Hampp, Alexander Dietsche, Valentin Bickel, Benjamin Sun, Christoph Meyer, and Marco Hutter. 2021. Traversing Steep and Granular Martian Analog Slopes With a Dynamic Quadrupedal Robot. arXiv:2106.01974 [cs.RO]

[10] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. 2021. Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning. arXiv:2108.10470 [cs.RO]

[11] Horia Mania, Aurelia Guy, and Benjamin Recht. 2018. Simple random search provides a competitive approach to reinforcement learning. arXiv:1803.07055 [cs.LG]

[12] Takahiro Miki, Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. 2022. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science Robotics* 7, 62 (2022), eabk2822. https://doi.org/10.1126/scirobotics.abk2822 arXiv:https://www.science.org/doi/pdf/10.1126/scirobotics.abk2822

[13] Kartik Paigwar, Lokesh Krishna, Sashank Tirumala, Naman Khetan, Aditya Sagi, Ashish Joglekar, Shalabh Bhatnagar, Ashitava Ghosal, Bharadwaj Amrutur, and Shishir Kolathaya. 2020. Robust Quadrupedal Locomotion on Sloped Terrains: A Linear Policy Approach. arXiv:2010.16342 [cs.RO]

[14] Jerry Pratt, John Carff, Sergey Drakunov, and Ambarish Goswami. 2006. Capture Point: A Step toward Humanoid Push Recovery. In *2006 6th IEEE-RAS International Conference on Humanoid Robots*. 200–207. https://doi.org/10.1109/ICHR.2006.321385

[15] Maurice Rahme, Ian Abraham, Matthew L. Elwin, and Todd D. Murphey. 2020. Dynamics and Domain Randomized Gait Modulation with Bezier Curves for Sim-to-Real Legged Locomotion. arXiv:2010.12070 [cs.RO]

[16] Marc H. Raibert. 1986. *Legged Robots That Balance*. Massachusetts Institute of Technology, USA.

[17] Nikita Rudin, Hendrik Kolvenbach, Vassilios Tsounis, and Marco Hutter. 2022. Cat-Like Jumping and Landing of Legged Robots in Low Gravity Using Deep Reinforcement Learning. *IEEE Transactions on Robotics* 38, 1 (Feb 2022), 317–328. https://doi.org/10.1109/tro.2021.3084374

[18] H. Shibly, K. Iagnemma, and S. Dubowsky. 2005. An equivalent soil mechanics formulation for rigid wheels in deformable terrain, with application to planetary exploration rovers. *Journal of Terramechanics* 42, 1 (2005), 1–13. https://doi.org/10.1016/j.jterra.2004.05.002

[19] Alessandro Tasora, Radu Serban, Hammad Mazhar, Arman Pazouki, Daniel Melanz, Jonathan A. Fleischmann, Michael Taylor, Hiroyuki Sugiyama, and Dan Negrut. 2015. Chrono: An Open Source Multi-physics Dynamics Engine. In *HPCSE*.

[20] Brian Wilcox, Todd Litwin, Jeff Biesiadecki, Jaret Matthews, Matt Heverly, Jack Morrison, Julie Townsend, Norman Ahmad, Allen Sirota, and Brian Cooper. 2007. ATHLETE: A cargo handling and manipulation robot for the moon. *Journal of Field Robotics* 24 (05 2007), 421–434. https://doi.org/10.1002/rob.20193

[21] Zhaoming Xie, Xingye Da, Buck Babich, Animesh Garg, and Michiel van de Panne. 2021. GLiDE: Generalizable Quadrupedal Locomotion in Diverse Environments with a Centroidal Model. arXiv:2104.09771 [cs.RO]

[22] F Zhou, RE Arvidson, K Bennett, K Iagnemma, C Senatore, R Lindemann, B Trease, P Bellutta, and S Maxwell. 2013. Simulating Mars Exploration Rover Opportunity Drives Using Artemis. In *44th Annual Lunar and Planetary Science Conference*.