Selecting Mechanical Parameters of a Monopode Jumping System with Reinforcement Learning

Andrew Albright[1] and Joshua Vaughan[2] — should affiliation show up somewhere?

***Abstract*— Legged systems have many advantages when compared to their wheeled counterparts. For example, they can more easily navigate extreme, uneven terrain. However, there are disadvantages as well, including difficulties seen in modeling the nonlinearities of the system. Research has shown that using flexible components is advantageous in terms of both efficiency and legged locomotive performance. Because of the difficulties encountered in modeling flexible systems, control methods such as reinforcement learning can be used to define control strategies. Furthermore, reinforcement learning can be tasked with learning mechanical parameters of a system to match a control input. It is shown in this work that by deploying reinforcement learning to solve for the optimal spring constant and damping ratio for a pogo-stick jumping system, designs which the agent defines are shown to be higher performing in terms of jumping height.**

I. INTRODUCTION

The use of flexible components within legged locomotive systems has proved useful for both reducing power consumption and increasing performance [1], [2], [3]. However, designing controllers for these systems is difficult as the flexibility of the system generates nonlinear models. As such, employing series-elastic-actuators (SEA) instead of flexible links is an attractive and popular solution, since the models of the systems become more manageable [2], [4], [5]. Still, the use of SEAs do not represent the full capability of flexible systems. As a result, other methods that use flexible tendon-like materials meant to emulate more organic designs have been proposed [6]. These, however, are still not representative of fully flexible links, which have been shown to drastically improve locomotive performance measures such as running speed [7].

Control methods have been developed that work well for flexible systems like the ones mentioned [8], [9]. However, as the systems increase in dimensionality, effects such as dynamic coupling between members make such methods challenging to implement. As such, work has been done ~~which~~ that uses neural networks and methods such as reinforcement learning (RL) to develop controllers for flexible systems [10], [11]. For example, RL has been used to create faster running control strategies for flexible-legged locomotive systems that also are robust to different design parameters [12].

In addition to the work done using RL to develop controllers for flexible systems, work has been completed which shows that this technique can be used to concurrently design the mechanical aspects of a system and a controller to match said system [13]. These techniques have even been used to define mechanical parameters and control strategies where the resulting controller and hardware were deployed in a sim-to-real process, validating the usability of the technique [14]. Using this technique for legged-locomotion has also been studied, but thus far has been limited to the case of rigid systems [15].

As such, this paper starts the discovery of using RL for concurrent design of flexible-legged locomotive systems. A simplified flexible jumping system was used where, for the initial work, the control input was held fixed so that the RL algorithm was tasked with only learning optimized mechanical parameters. The rest of the paper is organized such that in the next section, similar work will be discussed. In Section III, the pogo-stick environment details will be defined. Next, in Section IV, the input used during training will be explained. Then, in Section V, the algorithm used along with the method of the experiments will be explained. The performance of the learned design is presented in Section VI.

II. RELATED WORK

A. Flexible Locomotive Systems

switch order

The use of flexible components within locomotive robotics systems has shown improvements in performance measures such as movement speed and jumping height [3], [1]. Previous work has shown that the use of flexible components in the legs of legged locomotion systems increase performance while decreasing power consumption [7]. Related to work on robotics systems employing flexible links, work has been done showing the uses of series-elastic-actuators for locomotive systems [5]. In much of this work, human interaction with the robotic systems is considered where rigidity is not ideal [4]. The studies of flexible systems are challenging however, as the models which represent them are often nonlinear and therefore difficult to develop control systems for. As such, there is a need for solutions which can be deployed to develop controllers for these nonlinear systems.

B. Controlling Flexile Systems Using RL

Control methods developed for flexible linked systems have been shown to be effective for position control and vibration reduction [8], [16]. Because of the challenges seen in scaling the controllers, methods utilizing reinforcement learning are of interest. This method has been used in simple planar cases, where it is compared to a PD control strategy for vibration suppression and proves to be a higher performing method [17]. Additionally, it has also been shown to be effective at defining control strategies for flexible-legged locomotion. The use of actor-critic algorithms such move to this column

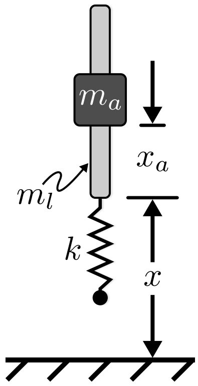


Fig. 1. Pogo-stick System

TABLE I
POGO-STICK MODEL PARAMETERS

Model Parameter	Value
Mass of Leg, m_l	0.175 kg
Mass of Actuator, m_a	1.003 kg
Spring Constant, $k_{nominal}$	5760 N/m
Natural Frequency, ω_n	$\sqrt{\frac{k}{m_l+m_a}}$
Damping Ratio, $\zeta_{nominal}$	1e-2 & 7.5e-2 $\frac{\mathrm{N}}{\mathrm{m/s}}$
Damping Constant, c	$2\,\zeta\,\omega_n m$
Actuator Stroke, $(x_a)_{\max}$	0.008 m
Max. Actuator Velocity, $(\dot{x}_a)_{\max}$	1.0 m/s
Max. Actuator Accel., $(\ddot{x}_a)_{\max}$	10.0 m/s^2

as Deep Deterministic Policy Gradient [18] have been used to train running strategies for a flexible legged quadruped [12]. Much of the research is based in simulation, however, and often the controllers are not deployed ~~in a sim-to-real fashion~~, which leads to the question on whether or not these are useful techniques in practice.

on physical systems

C. Concurrent Design

Defining an optimal controller for a system can be difficult due to challenges such as mechanical and electrical design limits. This is especially true when the system is flexible and the model is nonlinear. A solution to this challenge is to concurrently design a system with the controllers so that the two are jointly optimized. This is has been researched in previous work as a strategy to develop better performing mechatronics systems [19]. More recent work has been completed which used advanced methods such as evolutionary strategies to define robot design parameters [20]. In addition to evolutionary strategies, reinforcement learning has been shown to be a viable solution for concurrent design of 2D simulated locomotive systems [13]. This is further shown to be a viable method by demonstrating more complex morphology modifications in 3D reaching and locomotive tasks [15]. However, these techniques have not yet been applied to flexible systems for locomotive tasks.

III. POGO-STICK MODEL

The pogo-stick model show in Figure 1 has been shown to be useful as a representation of several different running and jumping gaits [21]. As such, it is used in this work to demonstrate the ability of reinforcement learning for the initial steps of concurrent design. The model parameters used in the simulations in this paper are summarized in Table I. The variable m_a represents the mass of the actuator, which moves along the rod with mass m_l. A nonlinear spring with constant k is used as the representation of flexibility. A damper (not shown in Figure 1), is parallel to the spring. Variables x and x_a represent the system's vertical position with respect to the ground and the actuator's position along the rod, respectively. The system is additionally constrained such that it only moves vertically, so the reinforcement agent is not required to balance the system.

a little awkward

The equations of motion describing the system are:

$$\ddot{x} = \alpha\left(\frac{k}{m_t}x^3 + \frac{c}{m_t}\dot{x}\right) - \frac{m_a}{m_t}\ddot{x}_a - g \tag{1}$$

Did we never go to $F_{sp} = k_1 x + k_3 x^3$ design of the nonlinear spring? I know we discussed it.

where x and x_a are position and velocity of the rod respectively, the acceleration of the actuator, x_a, is the control input, and m_t is the mass of the complete system. Ground contact determines the value of α, so that the spring and damper do not supply force while the leg is airborne:

$$\alpha = \begin{cases} -1, & x \leq 0 \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

add a \hphantom{-}

Does this work include compression constraint on the spring/damper? If so, you need to include that here.

IV. JUMPING COMMAND DESIGN

Bang-bang derived jumping commands like the one shown in Figure 2 are likely to result in a maximized jump height. why? For this command, the actuator mass travels at maximum acceleration within its allowable range, pauses, then accelerates in the opposite direction. Commands designed to complete this motion are bang-bang in each direction, with a selectable delay between them. The resulting motion of the actuator along the rod is shown in Figure 3. Starting from an initial position, $(x_a)_0$, it moves through a motion of stroke length Δ_1, pauses there for δ_t, then moves a distance Δ_2 during the second portion of the acceleration input.

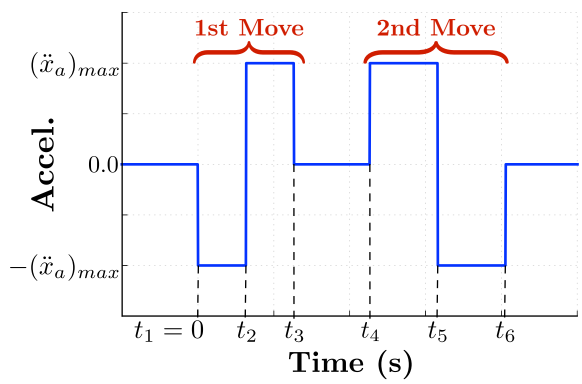


Fig. 2. Jumping Command

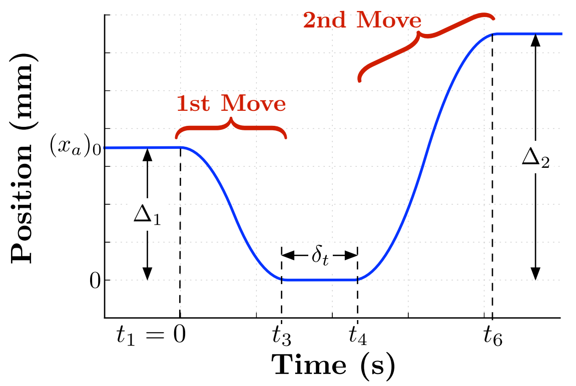


Fig. 3. Resulting Actuator Motion

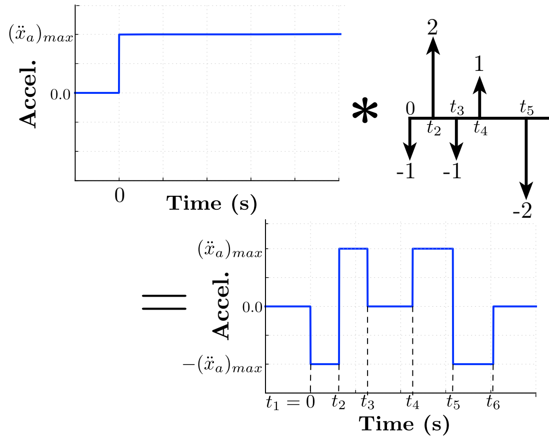


Fig. 4. Decomposition of the Jump Command into a Step Convolved with an Impulse Sequence

This bang-bang-based profile can be represented as a step command convolved with a series of impulses, as shown in Figure 4 [22]. Using this decomposition, input-shaping principles and tools can be used to design the impulse sequence [23], [24]. For the bang-bang-based jumping command, the amplitudes of the resulting impulse sequence are fixed, $A_i = [-1, 2, -1, 1, -2, 1]$. However, the impulse times, t_i, can be varied and optimal selection of them can lead to a maximized jump height of the pogo-stick system [25]. Commands of this form will often result in a stutter jump like that in Figure 5, where the small initial jump allows the system to compress the spring to store ~~farther storing more~~ energy in the spring to be used in the final jump. This jumping command type was used as the input for the pogo-stick during the simulation phase of training.

V. Learning Spring Constant and Damping Ratio

A. Reinforcement Learning Algorithm

The algorithm used for this work was Twin Delayed Deep Deterministic Policy Gradient (TD3) [26]. This is an actor-critic algorithm wherein there exists two main neural networks and a set of twin trailing networks. The first main network is the actor, which determines the action of the agent. This network takes in the systems state, $\mathcal{S}$, and outputs the action, $\mathcal{A}$, based on the state. The critic is an estimator of the value of being in a state and is used to determine the difference between expected and estimated value used to update the actor network during training. It takes in the systems state, $\mathcal{S}$, and outputs the expected future reward, $\mathbb{R}$, from being in that state. The twin trailing networks are used to find the temporal difference error against the critic network which is used to update the critic network. a little awkward

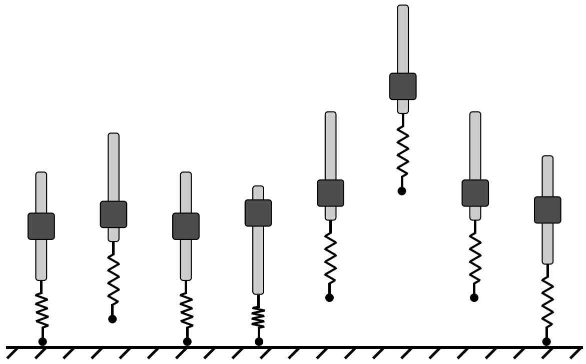


Fig. 5. Example Stutter Jump

TABLE II
TD3 Training Hyperparameters

Hyperameter	Value
Learning Rate, α	0.001
Learning Starts	100 Steps
Batch Size	100 Transitions
Tau, τ	0.005
Gamma, γ	0.99
Training Frequency	1:Episode
Gradient Steps	$\propto$ Training Frequency
Action Noise, ϵ	None
Policy Delay	1 : 2 Q-Function Updates
Target Policy Noise, ϵ	0.2
Target Policy Clip, c	0.5
Seed	100 Random Seeds

The training hyperparameters were selected based on experimental findings and are highlighted in Table II. Many of them were standard in comparison to what is implemented through Stable Baselines and what is suggested by the authors of the algorithm [26]. The rollout setting (Learning Starts) is key, however, as it saves to the replay buffer a set of randomly explored design selections allowing the agent to learn quickly. So, how did you choose it?

B. Training Environment Design

To allow the agent to find a mechanical design, a reinforcement learning environment conforming to the OpenAI Gym standard [27] was created for the pogo-stick model described in Section III, including a fixed controller input based on the algorithm described in the previous section. Unlike the common use case for RL, which is tasking the agent with finding a control input to match a design, the agent in this work was tasked with finding mechanical parameters to match a control input.

Better to use a sequential colormap for these plots, instead of a divergent one.

The mechanical parameters the agent was tasked with optimizing were the spring constant and the damping ratio of the pogo-stick system. At each time step during training, the agent selected a set of design parameters from a distribution of available designs, that were used to simulate the design using the input described in the previous section. The actions applied, $\mathcal{A}$, and transitions saved, $\mathcal{S}$, from the environment were defined as follows:

Should the k and ζ here be subscripted... they are just the "nominal" values, right?

$$\mathcal{A} = \{\{k \in \mathbb{R} : [-0.9k, 0.9k]\},\\ \{\zeta \in \mathbb{R} : [-0.9\zeta, 0.9\zeta]\}\} \tag{3}$$

$$\mathcal{S} = \left\{ \sum_{t=0}^{t_f} x_t,\ \sum_{t=0}^{t_f} \dot{x}_t,\ \sum_{t=0}^{t_f} x_{at},\ \sum_{t=0}^{t_f} \dot{x}_{at} \right\} \tag{4}$$

where k and ζ where the nominal? spring constant and damping ratio of the pogo-stick, respectively; x_t and $\dot{x}_t$ were the pogo-stick rod height and velocity steps, and x_{at} and $\dot{x}_{at}$ are the pogo-stick actuator position and velocity steps, all captured during simulation.

C. Reward Function Design

The RL algorithm was utilized to find designs for two different reward cases. Time series data was captured during the simulation phase of training and was used to evaluate the designs performance through these rewards. The first reward case used was:

$$\mathbb{R}_1 = \left(\sum_{t=0}^{t_f} x_t \right)_{max} \tag{5}$$

where x_t was ~~a step of~~ the pogo-stick's rod height ~~captured~~ at each step during simulation. The goal of the first reward was to find a design that, ~~when simulated,~~ would allow the pogo-stick to jump as high as possible. ¶ The reward for the second case was:

$$\mathbb{R}_2 = \frac{1}{\frac{|\mathbb{R}_1 - x_s|}{x_s} + 1} \tag{6}$$

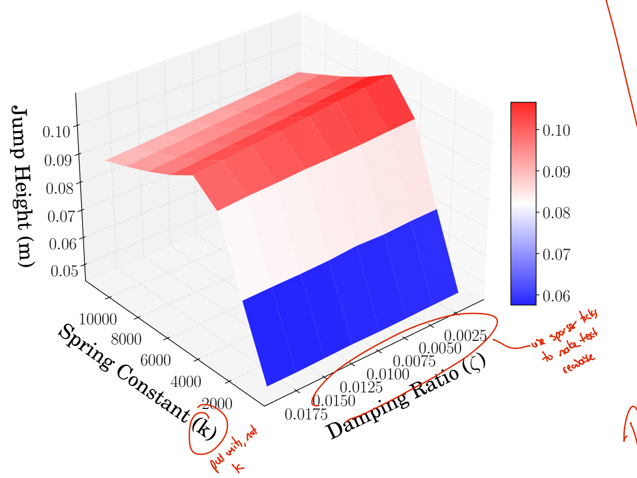
Awkward

where x_s was the ~~specified~~ desired jump height ~~the pogo-stick was needing to jump to. In this work,~~ x_s, which was set to 0.01 meters. ~~This was less than the maximum achievable height for the pogo-stick system, based on experimental observation.~~ The second case was utilized to test RL's ability to find a design that ~~when simulated,~~ minimized the error between the maximum height reached and the height specified.

D. Training Schedule

To evaluate the algorithms ability to robustly find design parameters meeting performance needs regardless of the neural network initializations, 100 different agents were trained with different network initialization seeds to find designs ~~that jumped to max heights and that jumped to specified heights~~. Additionally, it was of interest to verify if RL could be used in cases where high variances in performance were seen when selecting different design parameters. In other words, it was of interest to see if RL could overcome the requirement of finding design parameters that were close to optimal in the first place. Therefore, an additional set of 100 agents were

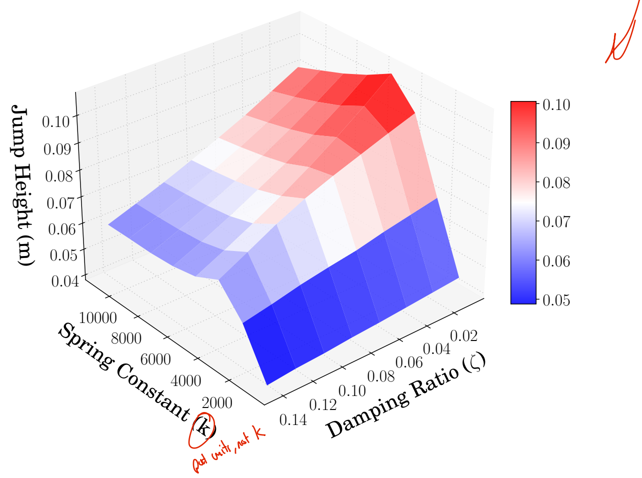
Let's clean this up a little



put units, not k

use sparser ticks to make text readable

Fig. 6. Height Jumped Given Close to Optimal Range of Design Parameters



put units, not k

Fig. 7. Height Jumped Given Non-close to Optimal Range of Design Parameters

trained where changes in both spring constant and damping ration made a significant effect of design performance.

All agents trained where done so for 900 learning steps preceding 100 random steps. This number of steps was chosen based on experimental observation of network updates needed for the reward to converge for the pogo-stick system. During the training process, the height reached during the simulation phase (per environment step) and the design parameters selected by the algorithm where collected to evaluate the learning process. The results are presented in the next section.

Awkward

?

VI. Jumping Height Reached

A. Evaluation Technique

In order to evaluate the ability of reinforcement for finding high performing design parameters, the performance of the design parameters the agents had access to are presented in Figures 6 and 7, one for each case previously mentioned. In the case presented in Figure 6, the design is close to what could be optimal, such that changes to the damping ratio

Awkward

We need to clean this up some... both in language and thoroughness of explanation. It's tough to follow as is.

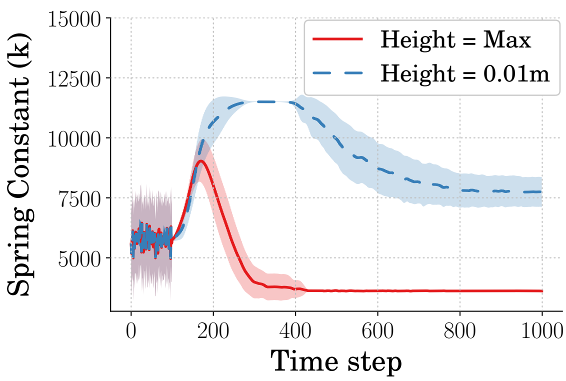


Fig. 8. Spring Constant Selected During Training

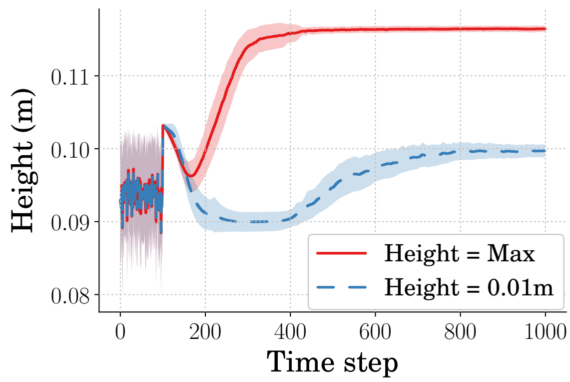


Fig. 10. Height Reached During Training

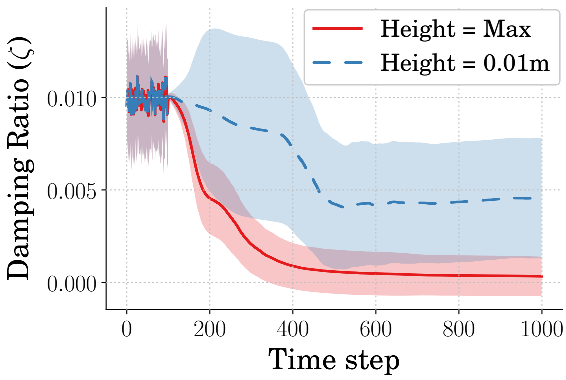


Fig. 9. Zeta Value Selected During Training

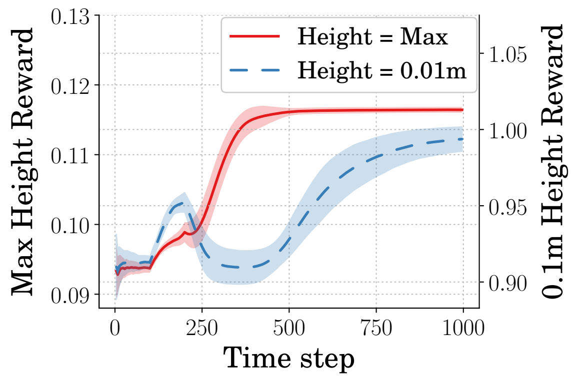


Fig. 11. Reward Received During Training

have a smaller effect. The second case then, presented in Figure 7, shows the performance of designs where changes to both the damping ratio and the spring constant have a large effect on performance.

B. Design Learned Given Close to Optimal Design Parameters

The average and standard deviation of the design parameters that the agent chose during training for simulation are displayed in Figures 8 and 9. These plots represents the learning curves for the agents seeking to define both high jumping designs and designs which jumped to 0.01 meters for the case where changes in damping ratio make less of a difference. The lack of change to jumping performance when altering damping ratio is supported in Figure 9 where, in the case of learning to jump to a specified height, the agent learned a variety of designs which closed in on satisfying the jumping height constraint. In the case of maximizing jump height, where there is only one design solution to accomplish the goal, there is significantly less variance.

Awkward

Stay formal

I think it'd be better to show these, then show k or ζ plots

Figures 10 and 11 show the height achieved by the learned designs, and the reward received during training for the same case just previously mentioned. Figure 10 can be used to compare the performance of the designs the agents learned throughout training with Figure 6 to see the designs are performing towards the maximum achievable performance regarding the agents learning designs to maximize jumping height. Additionally, looking at the agents learning designs to jump to the specified 0.01 meters, the data shows the designs, even being high in variances regarding design parameters, are low in variance regarding height jumped to. Figure 11 shows the rewards the agents are receiving during training and support both agent learning designs which have converged within the time trained.

Stopped reading here

C. Given Non-close to Optimal Design Parameters

The average and standard deviation of the design parameters that the agent chose during training for simulation are displayed in Figures 12 and 13. These plots represents the learning curves for the agents seeking to define both high jumping designs and designs which jumped to 0.01 meters for the case where changes in damping ratio make more of a difference. The increase in change to jumping performance when altering damping ratio is supported in Figure 13 where, in both cases, the agent learned a specific damping ratio. However, in comparing the data from these figures to that of Figure 6, the agents learning designs to maximize jump height seem to have done so in a way

Somet might not be right with Fig- ure 7. I am go- ing to look into it to make sure it

VII. Conclusion

Conclusion.

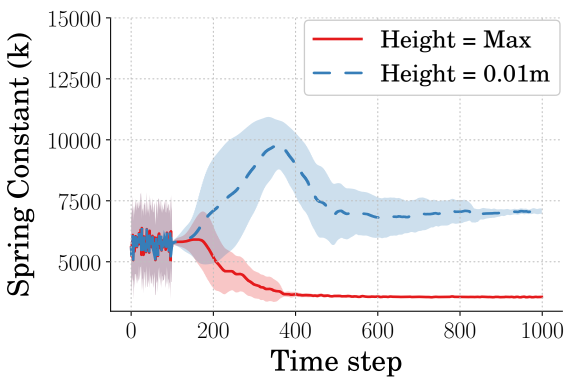


Fig. 12. Spring Constant Selected During Training

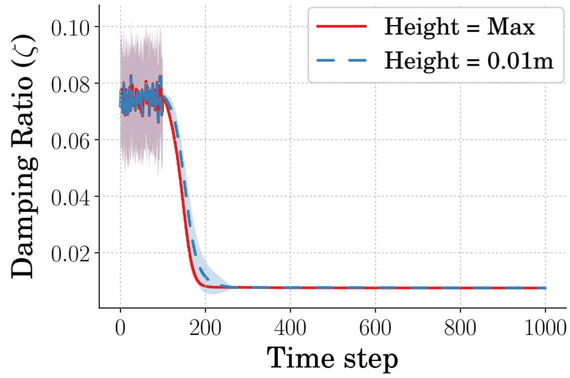


Fig. 13. Zeta Value Selected During Training

ACKNOWLEDGMENT

The authors would like to thank the Louisiana Crawfish Promotion and Research Board for their support of this work.

REFERENCES

[1] Y. Sugiyama and S. Hirai, "Crawling and jumping of deformable soft robot," *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 4, no. c, pp. 3276–3281, 2004.

[2] G. Buondonno, J. Carpentier, G. Saurel, N. Mansard, A. De Luca, and J. P. Laumond, "Actuator design of compliant walkers via optimal control," *IEEE International Conference on Intelligent Robots and Systems*, vol. 2017-Septe, pp. 705–711, 2017.

[3] J. Hurst, "The Role and Implementation of Compliance in Legged Locomotion," *The International Journal of Robotics Research*, vol. 25, no. 4, p. 110, 2008.

[4] T. Zhang and H. Huang, "Design and Control of a Series Elastic Actuator with Clutch for Hip Exoskeleton for Precise Assistive Magnitude and Timing Control and Improved Mechanical Safety," *IEEE/ASME Transactions on Mechatronics*, vol. 24, no. 5, pp. 2215–2226, 2019.

[5] G. A. Pratt and M. M. Williamson, "Series elastic actuators," *IEEE International Conference on Intelligent Robots and Systems*, vol. 1, pp. 399–406, 1995.

[6] F. Iida, G. Gómez, and R. Pfeifer, "Exploiting body dynamics for controlling a running quadruped robot," *2005 International Conference on Advanced Robotics, ICAR '05, Proceedings*, vol. 2005, pp. 229–235, 2005.

[7] U. Saranli, M. Buehler, and D. E. Koditschek, "RHex: A Simple and Highly Mobile Robot," *International Journal of Robotics Research*, vol. 20, no. 7, pp. 616–631, 2001.

[8] Z.-h. Luo, "Direct Strain Feedback Control of Flexible Robot Arms: New Theoretical and Experimental Results," vol. 38, no. 11, 1993.

?

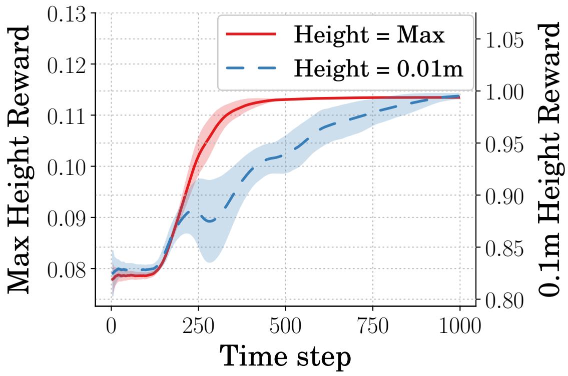


Fig. 14. Reward Received During Training

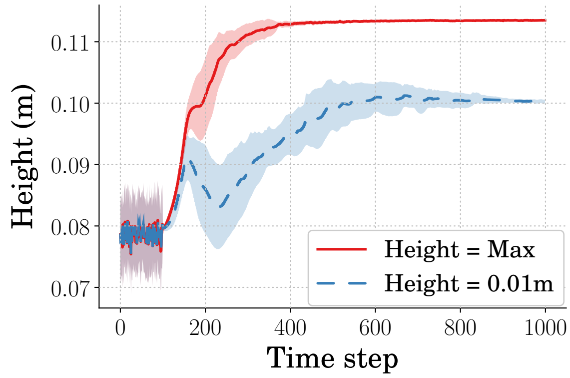


Fig. 15. Height Reached During Training

[9] A. M. Modeling, "Gain Adaptive Nonlinear Feedback Control of Flexible SCARA / Cartesian Robots," no. AIM, pp. 1423–1428, 2003.

[10] S. Bhagat, H. Banerjee, Z. T. H. Tse, and H. Ren, "Deep reinforcement learning for soft, flexible robots: Brief review with impending challenges," *Robotics*, vol. 8, no. 1, pp. 1–36, 2019.

[11] T. G. Thuruthel, E. Falotico, F. Renda, and C. Laschi, "Model Based Reinforcement Learning for Closed Loop Dynamic Control of Soft Robotic Manipulators," *IEEE Transactions on Robotics*, vol. 35, pp. 124–134, 2019.

[12] Z. Dwiel, M. Candadai, and M. Phielipp, "On Training Flexible Robots using Deep Reinforcement Learning," *IEEE International Conference on Intelligent Robots and Systems*, pp. 4666–4671, 2019.

[13] D. Ha, "Reinforcement learning for improving agent design," *Artificial Life*, vol. 25, no. 4, pp. 352–365, 2019.

[14] T. Chen, Z. He, and M. Ciocarlie, "Hardware as Policy: Mechanical and computational co-optimization using deep reinforcement learning," *arXiv*, no. CoRL, 2020.

[15] C. Schaff, D. Yunis, A. Chakrabarti, and M. R. Walter, "Jointly learning to construct and control agents using deep reinforcement learning," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2019-May, pp. 9798–9805, 2019.

[16] M. Ahmadi and M. Buehler, "Stable control of a simulated one-legged running robot with hip and leg compliance," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 1, pp. 96–104, 1997.

[17] W. He, H. Gao, C. Zhou, C. Yang, and Z. Li, "Reinforcement Learning Control of a Flexible Two-Link Manipulator: An Experimental Investigation," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–11, 2020.

[18] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 2016.

?

[19] Q. Li, W. J. Zhang, and L. Chen, "Design for control - A concurrent engineering approach for mechatronic systems design," *IEEE/ASME Transactions on Mechatronics*, vol. 6, no. 2, pp. 161–169, 2001.

[20] T. Wang, Y. Zhou, S. Fidler, and J. Ba, "Neural graph evolution: Towards efficient automatic robot design," *arXiv*, pp. 1–17, 2019.

[21] R. Blickhan and R. J. Full, "Similarity in multilegged locomotion: Bouncing like a monopode," *Journal of Comparative Physiology A*, vol. 173, no. 5, pp. 509–517, 1993.

[22] K. L. Sorensen and W. E. Singhose, "Command-induced vibration analysis using input shaping principles," *Automatica*, vol. 44, no. 9, pp. 2392 – 2397, 2008, analysis tools;Analytical tools;Command shaping;Deconvolution;Dynamic Systems;Induced vibrations;Input shaping;Key systems;Performance analyses;Phase plane;Residual vibrations;Simple methods;Vector diagram;Vibration analysis;Vibratory response;. [Online]. Available: http://dx.doi.org/10.1016/j.automatica.2008.01.029

[23] N. C. Singer and W. P. Seering, "Preshaping command inputs to reduce system vibration," *Journal of Dynamic Systems, Measurement, and Control*, vol. 112, pp. 76–82, March 1990.

[24] W. Singhose, W. Seering, and N. Singer, "Residual vibration reduction using vector diagrams to generate shaped inputs," *ASME J. of Mechanical Design*, vol. 116, pp. 654–659, June 1994.

[25] J. Vaughan, "Jumping Commands For Flexible-Legged Robots," 2013.

[26] S. Fujimoto, H. Van Hoof, and D. Meger, "Addressing Function Approximation Error in Actor-Critic Methods," *35th International Conference on Machine Learning, ICML 2018*, vol. 4, pp. 2587–2601, 2018.

[27] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," pp. 1–4, 2016. [Online]. Available: http://arxiv.org/abs/1606.01540