Mechanical Design and Control of Flexible-Legged Jumping Robots

A Thesis

Presented to the

Graduate Faculty of the

University of Louisiana at Lafayette

In Partial Fulfillment of the

Requirements for the Degree

Masters of Science

Andrew Albright

Spring 2022

Mechanical Design and Control of Flexible-Legged Jumping Robots

Andrew Albright

APPROVED:

Joshua E. Vaughan, Chair Assistant Professor of Mechanical Engineering	Anthony S. Maida Associate Professor of Computer Science
Alan A. Barhorst Department Head of Mechanical Engineering	Mary Farmer-Kaiser Dean of the Graduate School

To all the poor souls using Word, one day you will see the light that is LaTeX.

"Before we work on artificial intelligence why don't we do something about natural stupidity?"

— Steve Polyak

Acknowledgments

I would like to firstly thank my advisor Dr. Joshua Vaughan for his leadership, both academic and personal, during my time here at the university. His support has been invaluable in helping me to understand the underlying ideas behind this material. Additionally, his constant drive to produce high quality material has kept me motivated in my attempts to do the same. I would not be here without him.

Additionally, I would like to thank my committee members, Dr. Alan Barhorst, Dr. Anthony Maida and Dr. Brian Post for their input in regards to this work. Along with my lab members during my time in the C.R.A.W.L.A.B., Gerald Eaglin, Adam Smith, Y (Eve) Dang, Brennan Moeller and Darcy LaFont. Their conversations and advice have greatly assisted me in my efforts to complete this work.

Lastly, I would like to thank the Louisiana Crawfish Board for their financial support in the form of a grant from the University of Louisiana at Lafayette. This grant has allowed me to work knowing my fiscal security was in tack.

Table of Contents

List of Tables

List of Figures

I Introduction and System Description

A legged locomotive robot can have many advantages over a wheeled or tracked one, particularly in regards to their ability to navigate uneven and unpredictable terrain [1–3]. They can achieve this advantage because of the numerous movement types they can deploy. Abilities such as independently placing their feet within highly rigid terrain and jumping or bounding over obstacles have been shown to be effective ways of locomoting []. These advantages do not come at no cost however. Legged systems are traditionally power inefficient compared to wheeled vehicles making them a less attractive option for applications where power conservation is required. Research has been conducted showing the usefulness of adding flexible components, like the legs seen on the robot in Fig. 1, for combating efficiency and other issues [3–6]. The addition of these components in legged robots has been shown to increase system performance measures such as running speed, jumping capability and power efficiency []. However, the addition of flexible components creates a system that is highly non-linear, and thus requires a more complex control system to be designed.

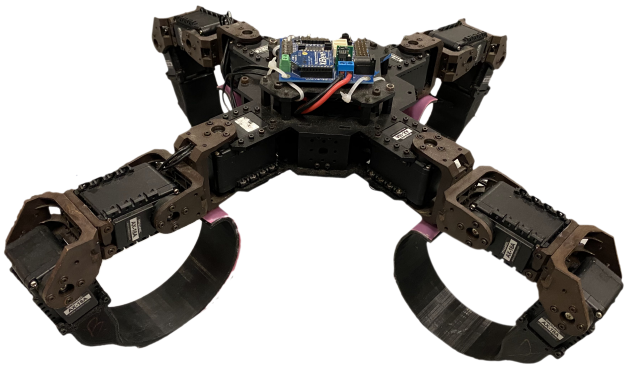


Figure 1. Flexible Robotics System

1.1 Improving Performance with Flexible Components

The addition of flexible components within robotic systems has been shown to be an effective way of improving performance metrics such as movement velocity and power efficiency [3, 6]. Of the different techniques that have be deployed the use of series elastic actuators (SEAs) has been shown to be very effective for increasing energy efficiency [7, 8]. The addition of flexible joints is not the only technique that has been used to improve performance however; utilizing tendon like elastic members to connect actuators to links has also been shown to be an effective way of improving efficiency [9]. The use of tendons, being an example of replicating what is found in nature, is a common method of finding unique mechanical designs that perform well in the real world. Research has been conducted finding the usefulness of including flexibility in the the spine of of 2D running robots where the velocity of the robot was drastically increased [10].

1.2 Controlling Flexible Systems

Finding an optimal control strategy for flexible-legged jumping systems can be a challenge. ...

Examples of traditional methods for controlling flexible systems.

1.3 Monopode Jumping System

The intent of the work completed in this thesis is to validate the use of reinforcement learning to concurrently design a system/controller architecture for flexible legged locomotive systems. To evaluate the methods used, a monopode system like the one shown in Fig. 2 was used to represent a flexible jumping system. This system has been studied and has been proven to be an effective base for modeling the jumping gaits for many different animals [11]. It will be referenced throughout the document along with the parameters assigned to the system.

The monopode is controlled by accelerating the actuator mass, m_a, along the

rod mass, m_l, causing a hopping like motion. A non-linear spring is modeled and represented by the variable α in the figure. Also included in the model is a damper parallel with the spring, having a damping coefficient of c, though it is not shown in the figure. Variables x and x_a represent the rod's global position and the actuator's local

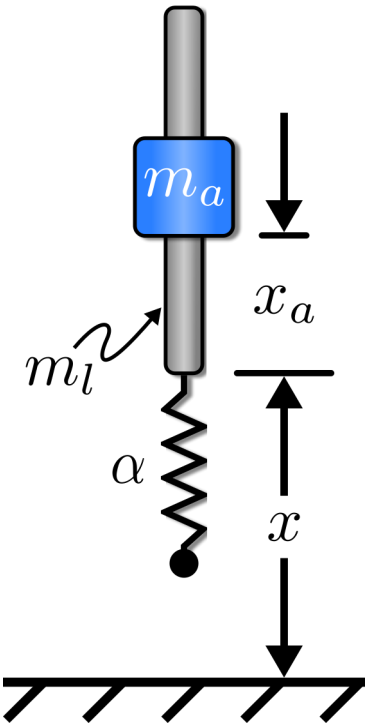


Figure 2. Monopode Jumping System

Table 1. Monopode Model Parameters

Model Parameter	Value
Mass of Leg, m_l	0.175 kg
Mass of Actuator, m_a	1.003 kg
Spring Constant, $\alpha_{nominal}$	5760 N/m
Natural Frequency, ω_n	$\sqrt{\frac{\alpha}{m_l+m_a}}$
Damping Ratio, $\zeta_{nominal}$	1e-2 $\frac{\text{N}}{\text{m/s}}$
Gravity, g	9.81 m/s^2
Actuator Stroke, $(x_a)_{\max}$	0.008 m
Max. Actuator Velocity, $(\dot{x}_a)_{\max}$	1.0 m/s
Max. Actuator Acceleration, $(\ddot{x}_a)_{\max}$	10.0 m/s^2

position with respect to the rod, respectively. The equations of motion for the system are: ...

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

Should these be in the appendix?

where x and $\dot{x}$ are position and velocity of the rod, respectively, the acceleration of the actuator, $\ddot{x}_a$, is the control input, and m_t is the mass of the complete system. Constants α and c represent the nonlinear spring and damping coefficient, respectively, and constant β is set to $1e8$. Ground contact determines the value of γ, so that the spring and damper do not supply force while the leg is airborne:

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

Additionally, the spring compression limit, or the systems position in the negative x direction, is limited to 0.008m. Additionally, the system is confined to move only vertically in regards to Fig. 2 so that controlling ballance is not required.

II Learning Efficient Jumping Strategies for the Monopode System

Utilizing reinforcement learning to (RL) train a neural network based controller has been shown to be useful for controlling many robotic systems [12, 13]. Successful control of rigid legged robots both in simulation and in real life has been shown to be highly effective [14, 15]. However, the use of RL to train controllers for flexible systems is limited, particularly in regards to legged locomotive systems.

There are many algorithms used to tain a network based controller in a reinforcement learning application, some of which have shown their ability to learn high performing control strategies for robotics systems []. Of the different algorithms used in research today, the one selected and tested in this work is Twin Delayed Deep Deterministic Policy Gradient (TD3) [16]. This is of the actor-critic type architecture and is further described in the Appendix. ...

In this work, RL is used to find control strategies for a flexible legged robotic system. The monopode described in Ch. 1 is used as a test system to validate the effectiveness of the RL approach.

Should these be there? Or should we have a section in a chapter?

2.1 Efficient Control Strategies

Efficient control of a robotic system is often one of the most important aspects of a controllers design. Applications where a robotic system is deployed and relies on a

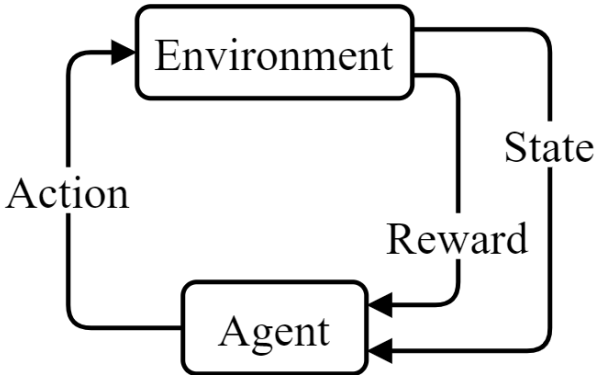


Figure 3. Reinforcement Learning Process
Work this into the text

limited power source such as a mobile walking robot will ofter require an efficient control strategy. Modern traditional methods such as model predictive control have been shown to produce energy efficient locomotion strategies for wheeled and legged systems [17, 18]. It is of interest in this work to utilize RL, a modern neural network based control method, to find strategies which are designed with power efficiency as the primary objective.

2.2 Learning to Jump Efficiently

Two different reward functions were designed to accomplish the task of determining how well RL learns efficient jumping strategies. The first reward function was one that ignored power usage all together and focused solely on the height of the jump:

$$R = x_t \tag{3}$$

where x_t was the height of the monopode system at any given time step. The second reward function was one that was defined to accomplish the same task, but also consider power consumption and was defined as:

$$R = \frac{x_t}{\sum_{t=0}^{t} P_t} \tag{4}$$

where P_t was the power consumption of the monopode system at any given time step defined mechanically as the product of the actuator's acceleration, velocity and mass:

$$P_t = m_a \, \dot{x_a} \, \ddot{x_a} \tag{5}$$

where m_a was the mass of the actuator, and $\dot{x_a}$ and $\ddot{x_a}$ where the actuators velocity and acceleration, respectively.

The purpose of defining two different reward functions was to compare the input commands and resulting jumping shapes of the two controller types to determine if the efficient controller was learning to conserve power.

2.3 Input Complexity

Two different jumping types were analyzed in this work. The first was referred to as a single jump command, and the second a stutter jump command. The intent of utilizing two different jumping commands was to determine if a RL algorithm was more or less effective in learning differing strategies depending on the complexity of the desired command.

An example single jump can be seen in Fig. 4. The intended command from the learned controller would be one that would jump the monopode once. This type of command would ideally compress the spring/damper by accelerating the actuator in the positive direction and then decompressing and jumping the system by accelerating in the negative direction.

An example stutter jump can be seen in Fig. 5. The intended command from the learned controller would be one that would jump the monopode twice. This type of command would ideally complete an optimal single jump, but then would decompress the spring and jump the system a second time achieving a higher jump.

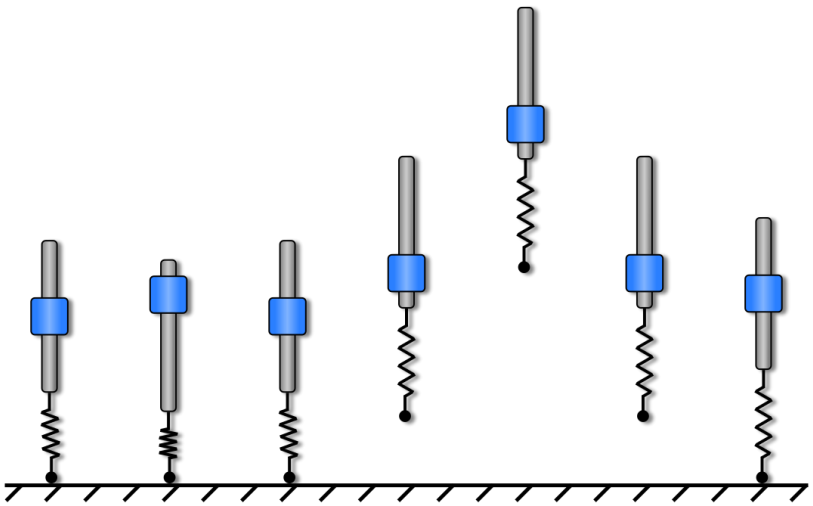


Figure 4. Example Single Jump

2.4 Average Performance of Network Controller

Display the data. . . .

Describe what is shown in the data.

. . .

The figures stacked side by side need larger font or they need to not be stacked.

2.5 Optimal Performance of Network Controller

Display the data. . . .

Describe what is shown in the data.

. . .

The figures stacked side by side need larger font or they need to not be stacked.

2.6 Conclusion

The conclusion.

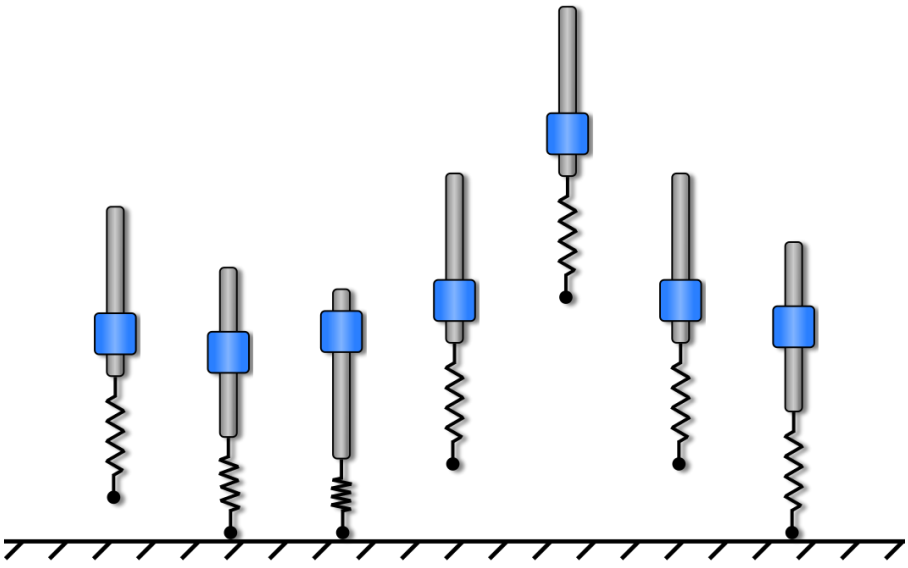
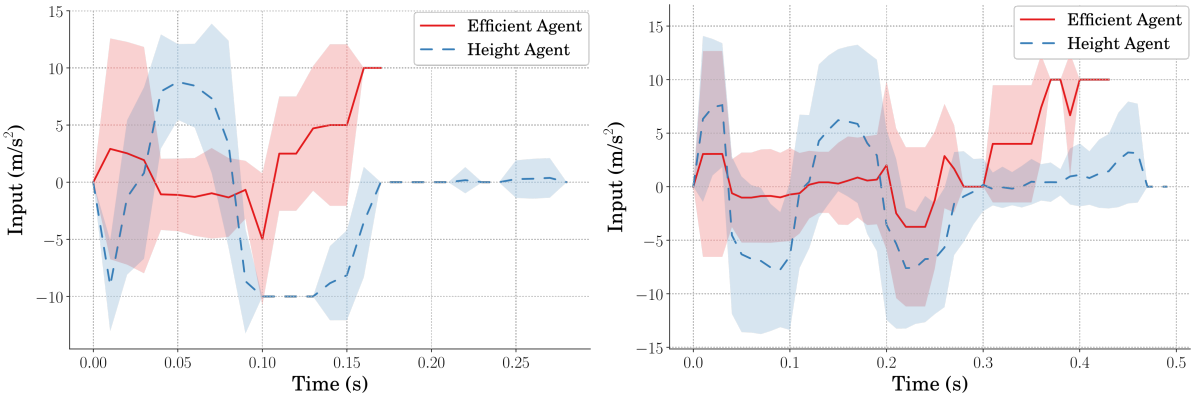
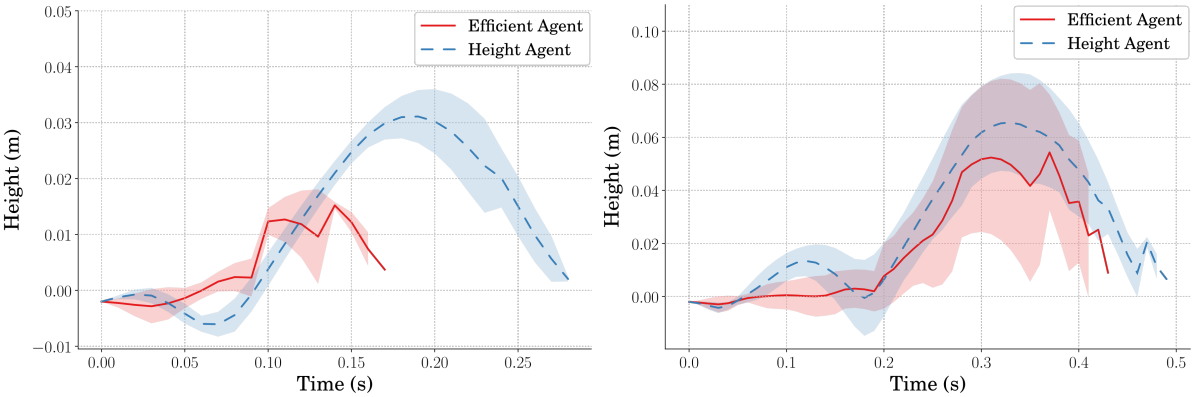


Figure 5. Example Stutter Jump



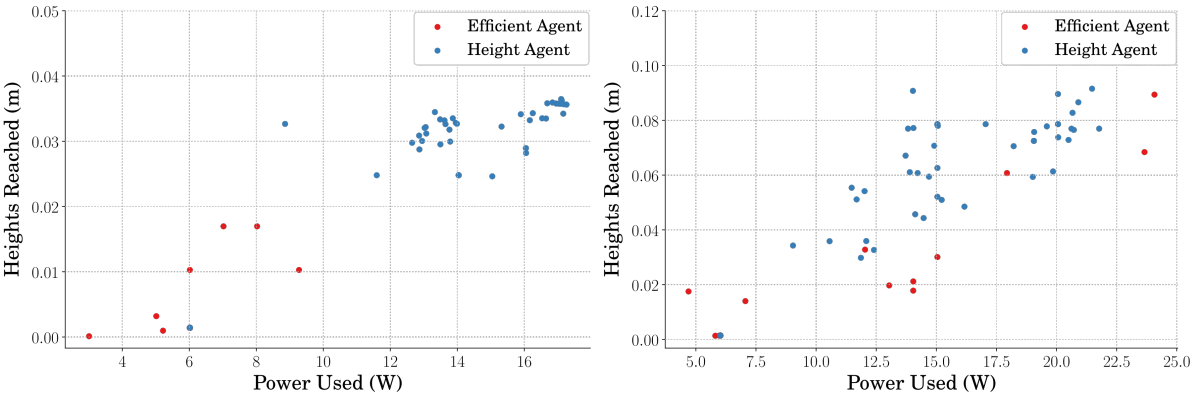
(a) Single Jumping Input **(b)** Stutter Jumping Input

Figure 6. Average and Standard Deviation Inputs to monopode



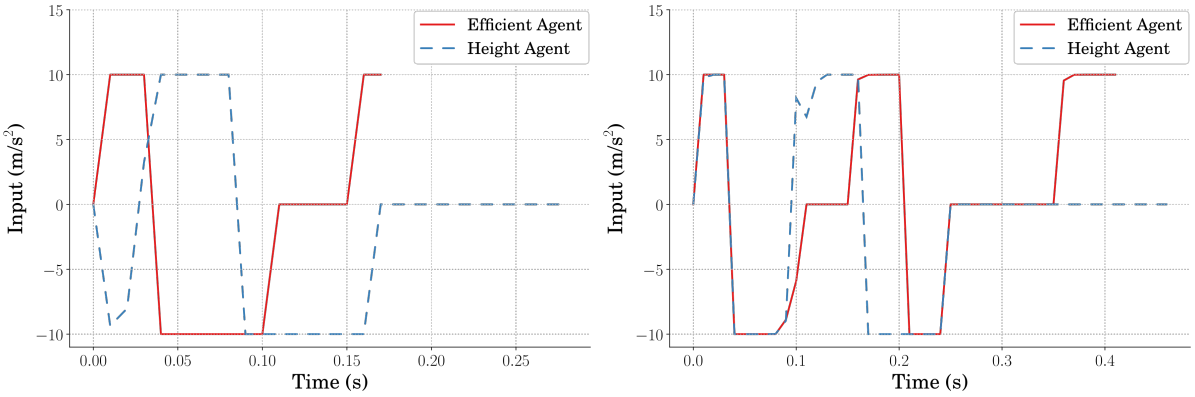
(a) Single Jumping Height **(b)** Stutter Jumping Height

Figure 7. Average and Standard Deviation Heights of monopode



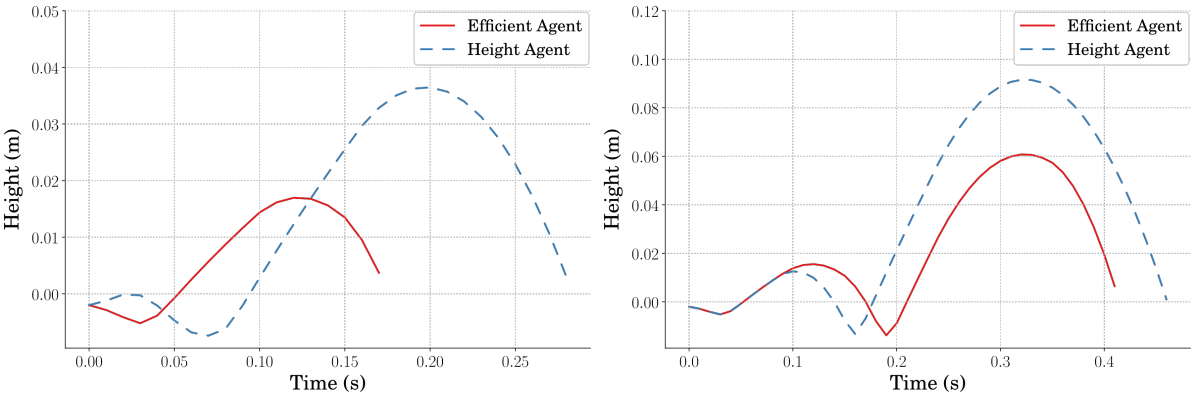
(a) Single Jumping Efficiency **(b)** Stutter Jumping Efficiency

Figure 8. Height Reached vs Power Consumed of monopode



(a) Single Jumping Input **(b)** Stutter Jumping Input

Figure 9. Optimal Inputs to monopode



(a) Single Jumping Height **(b)** Stutter Jumping Height

Figure 10. Optimal Heights of monopode

III Using Input Shaping to Validate RL Controller

Discussion on the use of input shaping to find optimal jumping strategies for jumping systems. How can one use these techniques to validate RL controllers? Do RL algorithms generate robust controllers when trained robustly.

3.1 Input Shaping Controller Input

Information from the paper DV wrote and some other resources found from the sources in that paper. Will need some resources from DV to fill in some input shaping information.

3.2 RL Controller Input

Discussion and figures from the inputs defined by the RL algorithms for the monopode system.

3.3 Training a Robust Controller

How are robust controllers trained using RL algorithms? How did we do ours? Show the results.

3.4 Conclusion

Discuss the results.

IV Mechanical Design of a the Monopode Jumping System

Often it is the goal of a controls engineer to design a controller to accommodate and manipulate systems according to the system description provided. However, research has been conducted showing the value of studying the manipulation of mechanical design parameters in order to achieve a desired system behavior []. In this chapter, reinforcement learning is shown to be useful as a tool to learn mechanical designs given a predefined system controller for the monopode jumping system.

4.1 Controller Input

Bang-bang based jumping commands like the one shown in Fig. 11 are likely to result in a maximized jump height [19]. 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 Fig. 12. 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

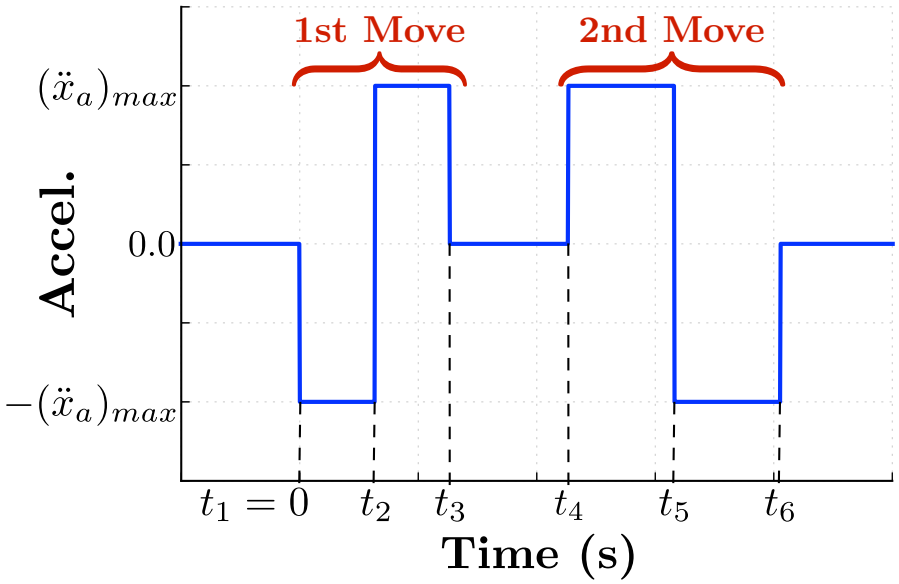


Figure 11. Jumping Command

during the second portion of the acceleration input.

This bang-bang-based profile can be represented as a step command convolved with a series of impulses, as shown in Fig. 13 [20]. Using this decomposition,

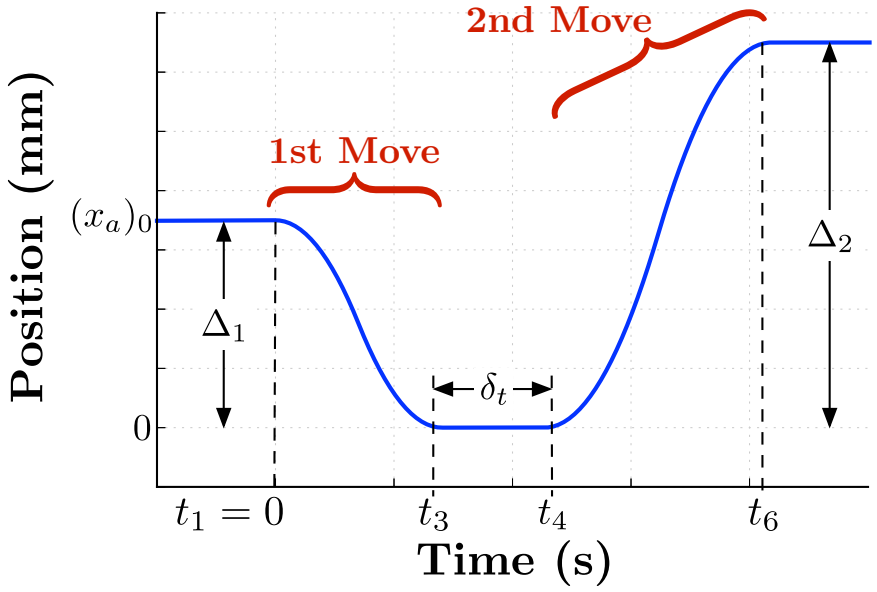


Figure 12. Resulting Actuator Motion

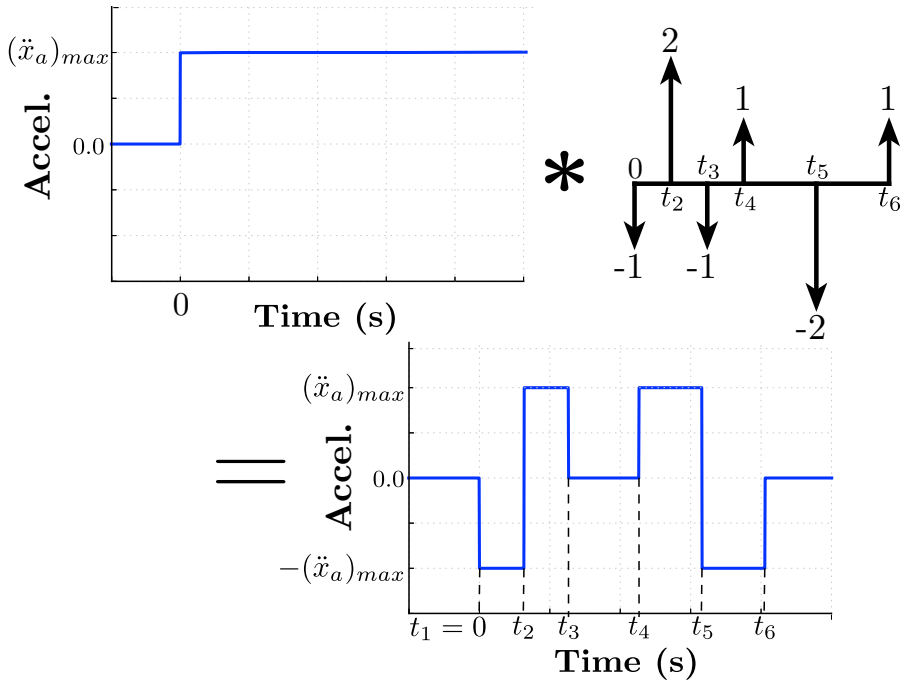


Figure 13. Decomposition of the Jump Command into a Step Convolved with an Impulse Sequence

input-shaping principles and tools can be used to design the impulse sequence [21, 22]. For the bang-bang-based jumping command, the amplitudes of the resulting impulse sequence are fixed, $A_i = [-1, 2, -1, 1, -2, 1]$. The impulse times, t_i, can be varied and optimal selection of them can lead to a maximized jump height of the monopode system [19]. Commands of this form will often result in a stutter jump like what was shown in Fig. 5, where the small initial jump allows the system to compress the spring to store energy to be used in the final jump. This jumping command type was used as the input for the monopode during the simulation phase of training.

4.2 Learning a Design

4.2.1 Algorithm for Learning a Design. The algorithm used for this work was Twin Delayed Deep Deterministic Policy Gradient (TD3) []. The training hyperparameters were selected based on TD3's author recommendations and Stable Baselines3 [] experimental findings and are displayed in Tab 2. All of the hyperparameters, with the exception of the rollout (Learning Starts) and the replay buffer, were set according to Stable Baselines3 standards. The rollout setting was defined such that the agent could search the design space at random, filling the replay

Table 2. 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

buffer with enough experience to prevent the agent from converging to a design space that was not optimal. The replay buffer was sized proportional to the number of training steps due to system memory constraints.

...

Maybe put in the algorithm description with references to the machancial design here?

4.2.2 Environment for Learning a Design. To allow the agent to find a mechanical design, a reinforcement learning environment conforming to the OpenAI Gym standard [] was created for the monopode model described in Chapter 1, including a fixed controller input based on the algorithm described in section 4.1. 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. The mechanical parameters the agent was tasked with optimizing were the spring constant and the damping ratio of the monopode system. At each episode during training, the agent selected a set of design parameters from a distribution of available designs. The actions applied, $\mathcal{A}$, and transitions saved, $\mathcal{S}$, from the environment were defined as follows:

$$\mathcal{A} = \{\{a_\alpha \in \mathbb{R} : [-0.9\alpha, 0.9\alpha]\},\ \{a_\zeta \in \mathbb{R} : [-0.9\zeta, 0.9\zeta]\}\} \tag{6}$$

$$\mathcal{S} = \left\{ \sum_{t=0}^{t_f} x_t,\ \sum_{t=0}^{t_f} \dot{x}_t,\ \sum_{t=0}^{t_f} x_{at},\ \sum_{t=0}^{t_f} \dot{x}_{at} \right\} \tag{7}$$

where α and ζ are the nominal spring constant and damping ratio of the monopode, respectively; x_t and $\dot{x}_t$ are the monopode's rod height and velocity steps, and x_{a_t} and $\dot{x}_{a_t}$ are the monopode's actuator position and velocity steps, all captured during simulation.

4.2.3 Rewards for Learning Designs. 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:

$$\mathbb{R}_1 = \left(\sum_{t=0}^{t_f} x_t \right)_{max} \tag{8}$$

where x_t was the monopode's rod height at each step during simulation. The goal of the first reward was to find a design that would cause the monopode to jump as high as possible.

The reward for the second case was:

$$\mathbb{R}_2 = \frac{1}{\frac{|\mathbb{R}_1 - x_s|}{x_s} + 1} \tag{9}$$

where x_s was the desired jump height, which was set to 0.01 m. The second case was utilized to test RL's ability to find a design that minimized the error between the maximum height reached and the desired maximum height to reach.

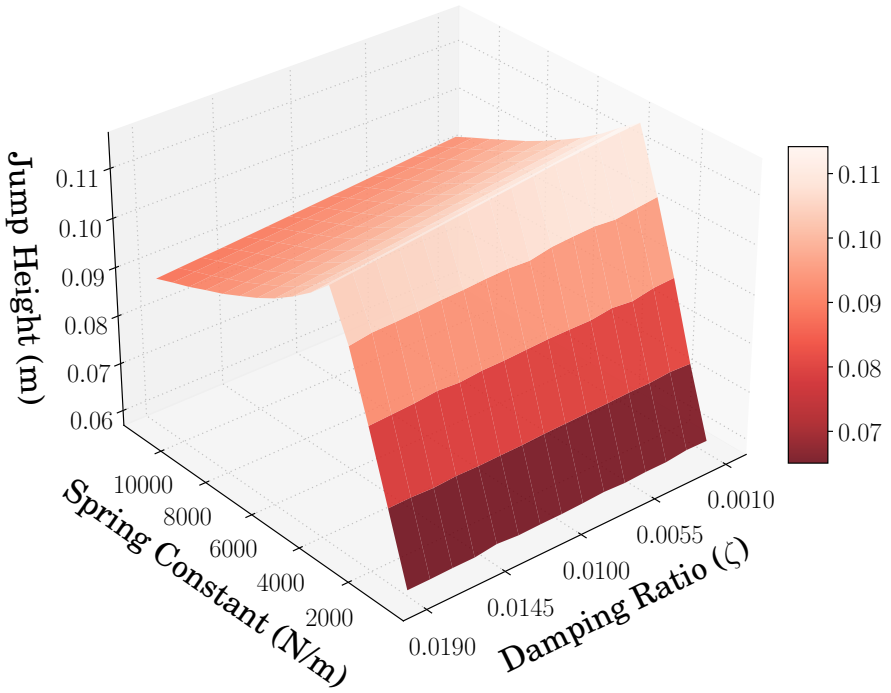


Figure 14. Jumping Performance of Narrow Design Space

4.2.4 Ability to Learn a Design.

Figures 14 and 15 represent the heights the monopode could reach for two different design spaces. The design space provided for the first case, shown in Fig. 14, represents a space where the allowable damping ratio was limited to a fairly narrow range. This limits the solution space, making it less likely that the agent will settle to a locally optimal value. The design space provided for the second case, shown in Fig. 15, represents a space where a wider range of damping ratios are allowed. This wider range of possible values makes it more likely that the agent will settle to a local maxima.

4.3 Jumping Performance

4.3.1 Narrow Design Space.

Figure 16 shows the height achieved by the learned designs for the agents given the narrow range of possible damping ratio values. For the agents learning designs to maximize jump height, Fig. 16 can be compared with Fig. 14 showing that the agent learned a design nearing one which would achieve

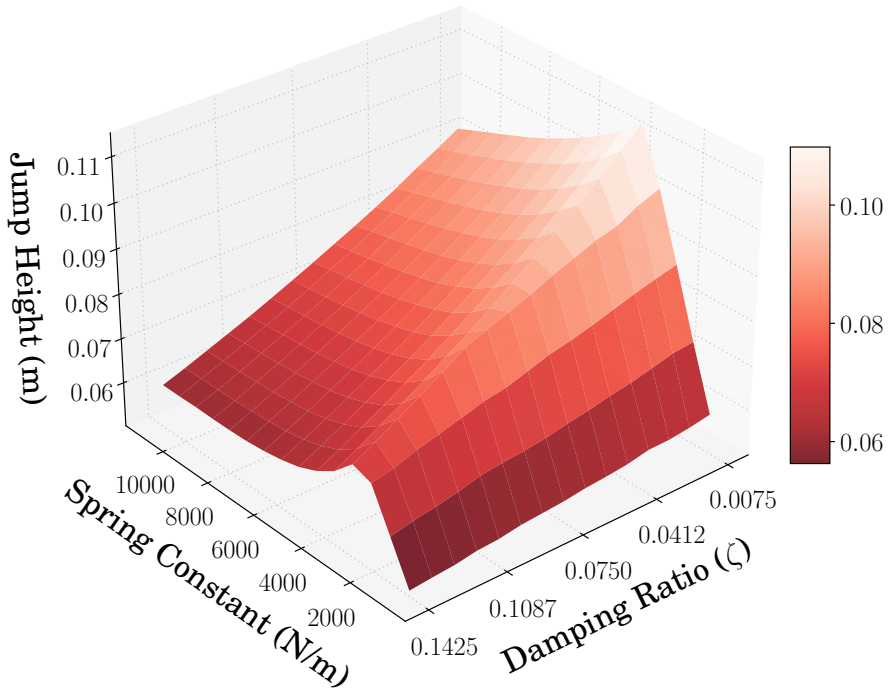


Figure 15. Jumping Performance of Broad Design Space

maximum performance. Additionally, looking at the agents learning designs to jump to the specified 0.01 m, the designs learned accomplish this with slightly more variance than that of the maximum height case. Figure 17 shows the rewards the agents received during training and support that both agent types learned designs which converged.

The average and standard deviation of the spring constant and damping ratio

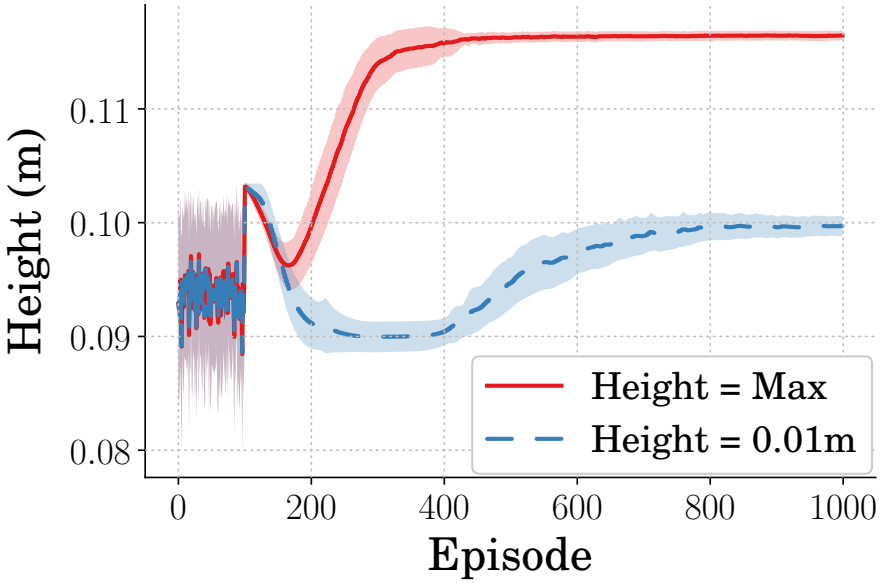


Figure 16. Height Reached During Training

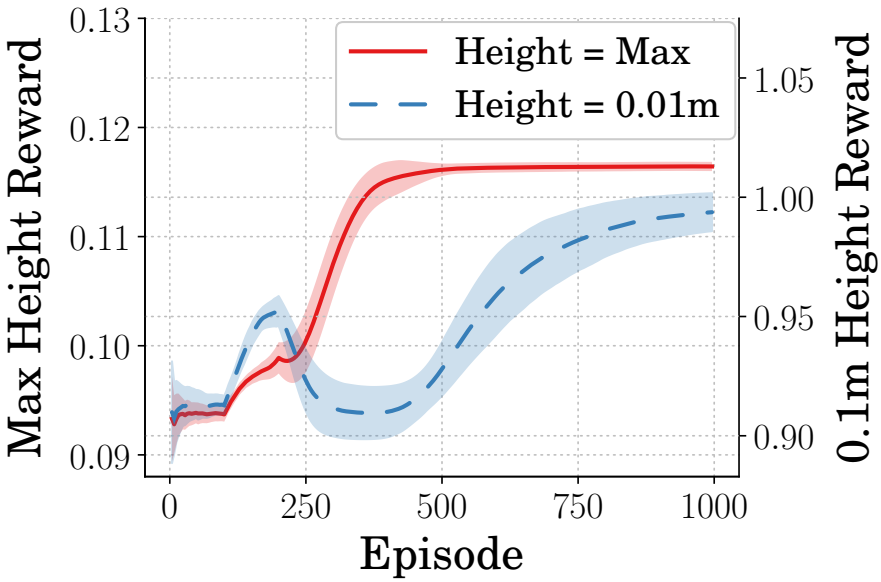


Figure 17. Reward Received During Training

design parameters the agents selected during training are shown in Figs. 18 and 19. These plots represent the learning curves for the agents learning design parameters to maximize jump height and the agents learning design parameters to jump to 0.01 m. There is a high variance in both the spring constant and the damping ratio found for the agents that learned designs to jump to a specified height. The agents which were learning designs which maximized height found designs with very little variance in terms of spring constant and significantly less variances in terms of damping ratio.

4.3.2 Wide Design Space. Figure 20 shows the height achieved by the learned designs for the agents given a wider range of damping ratios. For the agents learning designs to maximize jump height, Figure 20 can be compared with Figure 15 showing that the agents learned a design nearing one which would achieve maximum performance. Additionally, looking at the agents learning designs to jump to the specified 0.01 m, the designs learned accomplish this, only with slightly more variance than what is seen in the maximum height agents. Figure 21 shows the rewards the agents received during training and support that both agent types learned designs

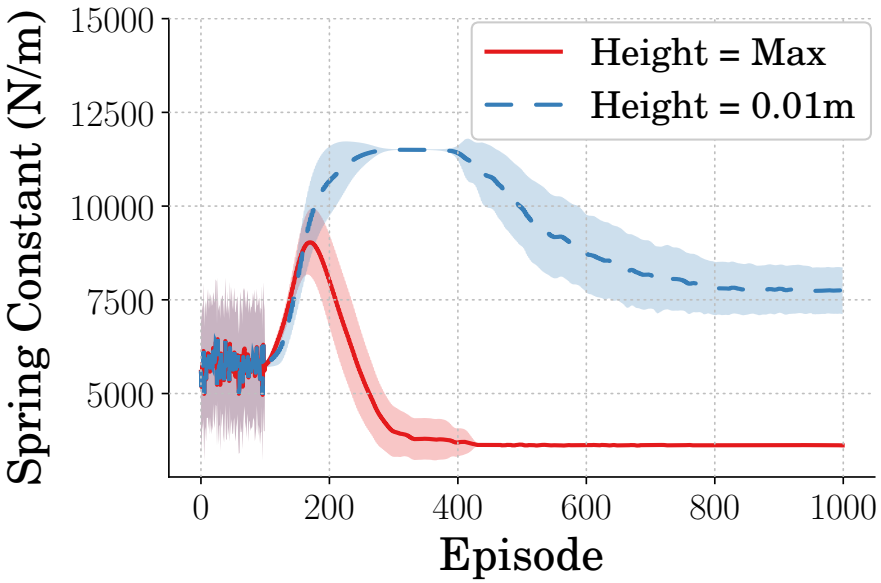


Figure 18. Spring Constant Selected During Training

which converged. In this case though, the agent learning a design to jump to a specific height requires more learning steps to converge.

The average and standard deviation of the spring constant and damping ratio design parameters the agents selected during training are shown in Figures 22 and 23. For the agents that learned designs to jump to a specified height, it can be seen that

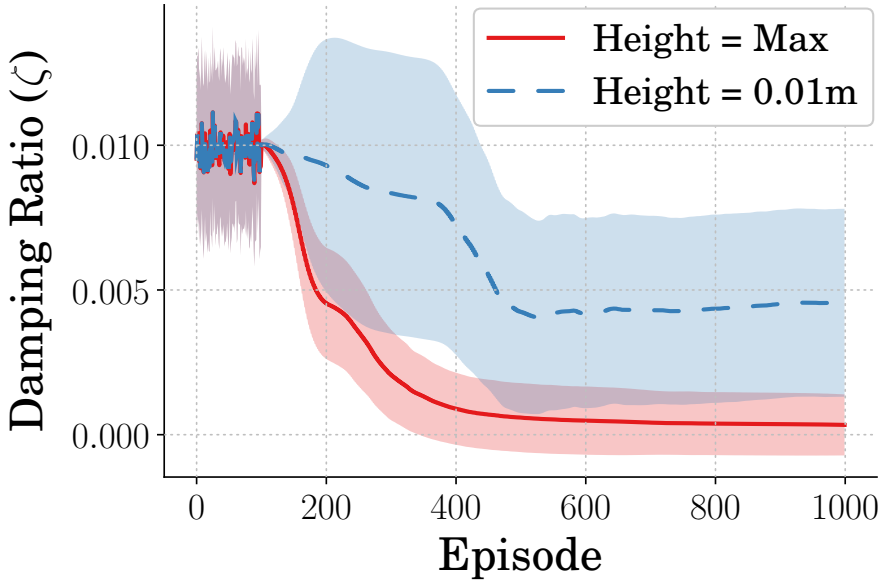


Figure 19. Damping Ratio Selected During Training

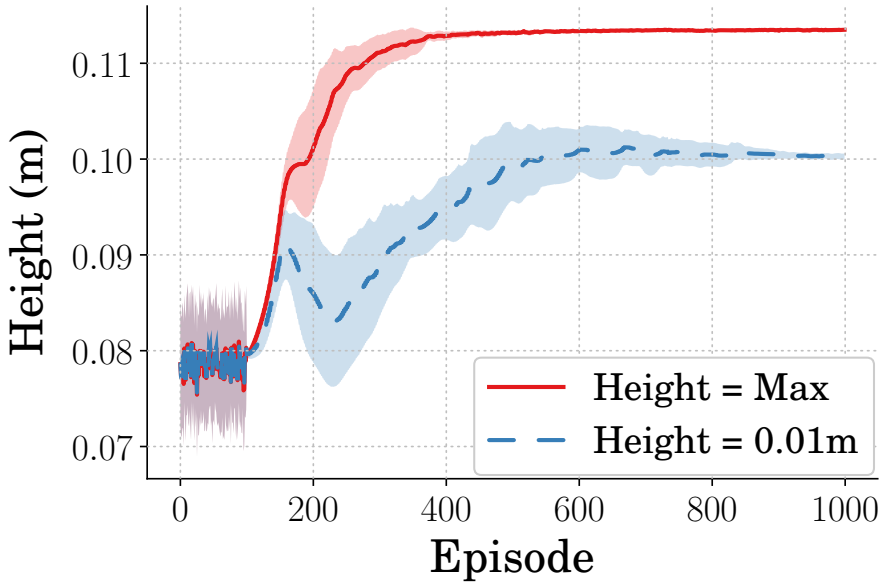


Figure 20. Height Reached During Training

there is a high variance in spring constant throughout training. However, the majority of agents converge to a specific design, lowering the variance. The same can be seen in the damping ratio; however, the variance is mitigated significantly earlier in training. The agents which were learning designs that maximized height found them with very little variance in terms of spring constant and damping ratio.

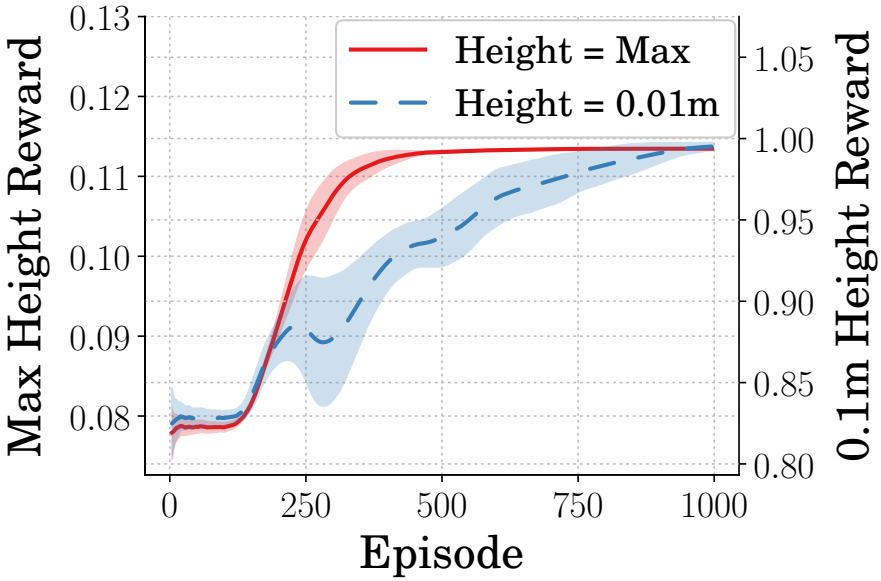


Figure 21. Reward Received During Training

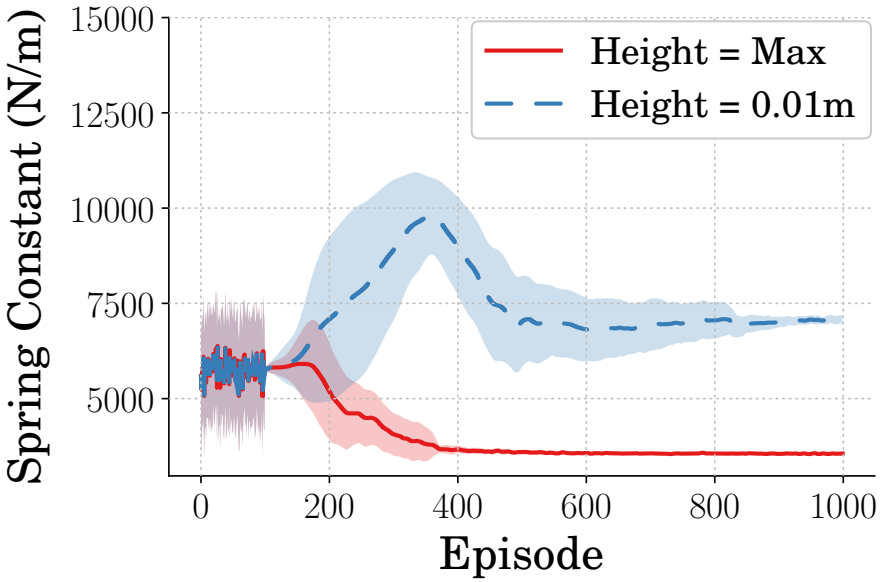


Figure 22. Spring Constant Selected During Training

4.3.3 Average Design Performance. The final mean and standard deviation of the design parameters for the two different cases are presented in Table 3. Figure 24 shows the jumping performance of the mean designs learned for both cases tested. The agents tasked with finding designs to jump to the specified 0.01 m, did so

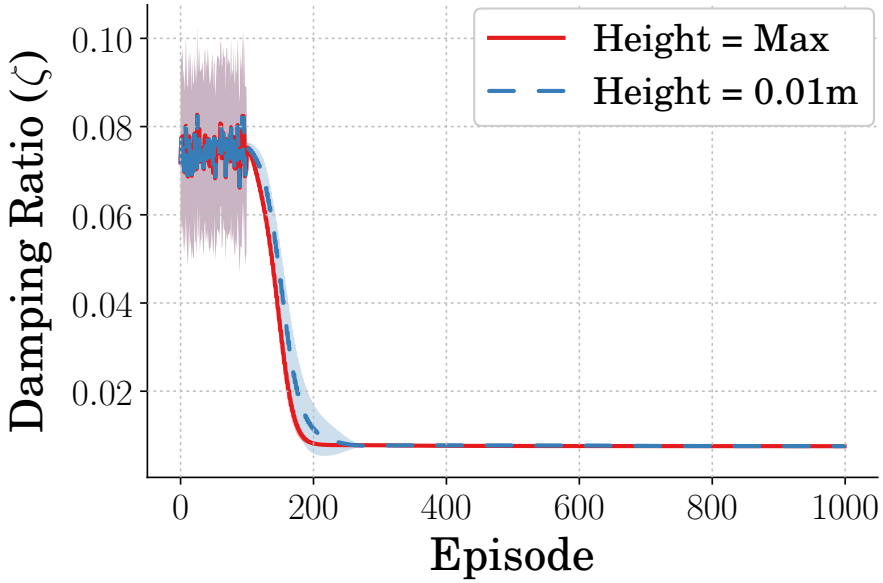


Figure 23. Damping Ratio Selected During Training

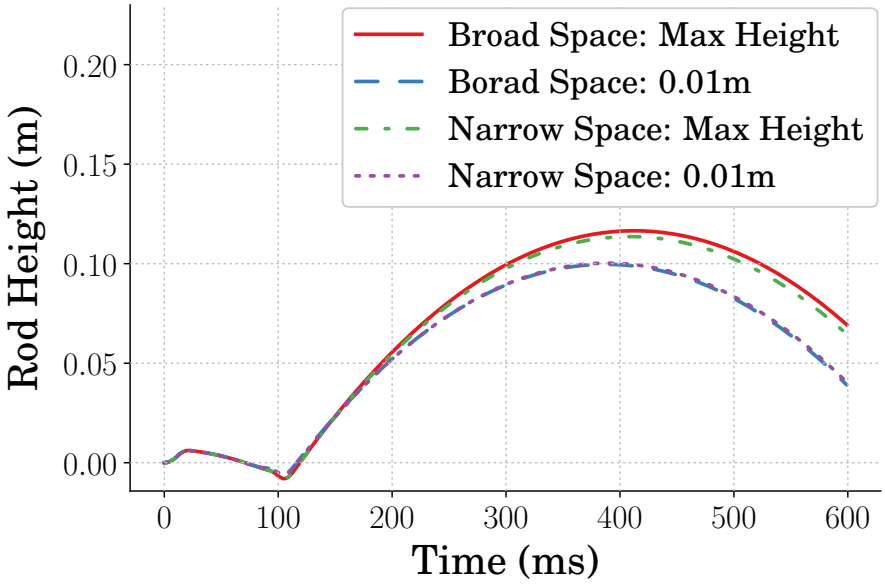


Figure 24. Height vs Time of Average Optimal Designs

with minimal error. The difference seen in maximum height reached between the two cases represents the difference in the damping ratio design space the agents had access to. The peak heights achieved can be compared again to Figures 14 and 15 to show that the agents learned designs nearing those achieving maximum performance.

...

This table is too wide. Not sure yet how to fix it.

Table 3. Learned Design Parameters

Training Case		Design Parameter	Mean	STD
Narrow Design Space	Learn Max Height	Spring Constant	3.62e03	3.82e01
		Damping Ratio	3.37e-04	2.11e-03
	Learn Specified Height	Spring Constant	7.74e03	1.24e03
		Damping Ratio	4.55e-03	6.49e-03
Broad Design Space	Learn Max Height	Spring Constant	3.55e03	4.86e01
		Damping Ratio	7.53e-03	8.86e-06
	Learn Specified Height	Spring Constant	7.07e03	2.16e02
		Damping Ratio	7.54e-03	3.27e-05

4.4 Conclusion

The monopode model was used in conjunction with a predetermined control input to determine if a reinforcement learning algorithm (TD3) could be used to find optimal performing design parameters regarding jumping performance. This work was done in part to determine if reinforcement learning could be used as the mechanical design learner for an intelligent concurrent design algorithm. It was shown that when providing an agent with a design space that was smaller in size, the agents performed well in finding design parameters which met the performance constraints. The designs found were high in design variance, however. It was additionally shown that when provided with larger design space, the agents excelled at finding design parameters which were lower in design variance but still met the design constraints.

V Concurrent Design of the Monopode System

Finding a control architecture for a mechanically defined system is often the the workflow for generating a controlled robotic system. However, the mechanical system is not always a simple one and generating a controller for it may require a more complex workflow. It is of interest as well to allow the mechanical parameters of the system, and therefore the system description, to be fluid allowing for a more optimal mesh between controller and system. Designing the system and control input in unison has been researched and is ofter refereed to as concurrent design []. The approach often takes a traditional route utilizing traditional methods to define the system and controller.

However the utilization of more complex deep learning methods has also been shown to be an effective strategy for finding optimal concurrent designs []. It has been used to find a concurrent designs for legged robotic systems leading to improved performance in regards to movement velocity []. Some research has done where the controllers were deployed in a simulation to real process validating that this technique is an effective strategy for developing a system/controller architecture [].

5.1 Algorithm Implementation

Show how the algorithm was implemented.

5.2 Jumping Cases

The cases which the agents were learning designs and controllers for.

5.3 Mechanical Designs

- Figure: the designs that the agent learned
- Figure: The learning curve during training of the agents

5.4 Controller Performance

- Figure: the performance of the controllers with nominal designs

- Figure: the performance of the controllers with learned designs

5.5 Conclusion

The conclusion.

VI Concurrent Design of a Two-Link Flexible-Legged Jumping System

Discussion on concurrent design for robotic systems. How does this relate to our system and work? Overview of the system.

- Figure: system
- Table: System parameters

6.1 Jumping Cases

The cases which the agents were learning designs and controllers for.

6.2 Mechanical Designs: Simulation

This is the data basically.

6.3 Controller Performance: Simulation

- Figure: the performance of the controllers with nominal designs
- Figure: the performance of the controllers with learned designs

6.4 Mechanical Designs: Sim-to-Real

This is the data basically.

6.5 Controller Performance: Sim-to-Real

- Figure: the performance of the controllers with nominal designs
- Figure: the performance of the controllers with learned designs

6.6 Conclusion

The conclusion.

VII Appendix: StableBaselines3

Maybe a needed description of the code.

VIII Appendix: Equations of Motion

EOMs.

Bibliography

[1] Park, H. W., Wensing, P. M., and Kim, S., "High-speed bounding with the MIT Cheetah 2: Control design and experiments," *International Journal of Robotics Research*, vol. 36, no. 2, pp. 167–192, 2017.

[2] Blackman, D. J., Nicholson, J. V., Pusey, J. L., Austin, M. P., Young, C., Brown, J. M., and Clark, J. E., "Leg design for running and jumping dynamics," *2017 IEEE International Conference on Robotics and Biomimetics, ROBIO 2017*, vol. 2018-Janua, pp. 2617–2623, 2018.

[3] Seok, S., Wang, A., Chuah, M. Y., Hyun, D. J., Lee, J., Otten, D. M., Lang, J. H., and Kim, S., "Design principles for energy-efficient legged locomotion and implementation on the MIT Cheetah robot," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 3, pp. 1117–1129, 2015.

[4] Sugiyama, Y. and Hirai, S., "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.

[5] Galloway, K. C., Clark, J. E., Yim, M., and Koditschek, D. E., "Experimental investigations into the role of passive variable compliant legs for dynamic robotic locomotion," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 1243–1249, 2011.

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

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

[8] Ahmadi, M. and Buehler, M., "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.

[9] Folkertsma, G. A., Kim, S., and Stramigioli, S., "Parallel stiffness in a bounding quadruped with flexible spine," *IEEE International Conference on Intelligent Robots and Systems*, pp. 2210–2215, 2012.

[10] Kani, M. H. H. and Ahmadabadi, M. N., "Comparing effects of rigid, flexible, and actuated series-elastic spines on bounding gait of quadruped robots," pp. 282–287, 2013.

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

[12] Vecerik, M., Hester, T., Scholz, J., Wang, F., Pietquin, O., Piot, B., Heess, N., Rothörl, T., Lampe, T., and Riedmiller, M., "Leveraging Demonstrations for Deep Reinforcement Learning on Robotics Problems with Sparse Rewards," pp. 1–10, 2017.

[13] Plappert, M., Andrychowicz, M., Ray, A., McGrew, B., Baker, B., Powell, G., Schneider, J., Tobin, J., Chociej, M., Welinder, P., Kumar, V., and Zaremba, W., "Multi-Goal Reinforcement Learning: Challenging Robotics Environments and Request for Research," pp. 1–16, 2018.

[14] Ha, S., Xu, P., Tan, Z., Levine, S., and Tan, J., "Learning to walk in the real world with minimal human effort," *arXiv*, no. CoRL, pp. 1–11, 2020.

[15] Zhao, W., Queralta, J. P., and Westerlund, T., "Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: A Survey," *2020 IEEE Symposium Series on Computational Intelligence, SSCI 2020*, pp. 737–744, 2020.

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

[17] Harper, M. Y., Nicholson, J. V., Collins, E. G., Pusey, J., and Clark, J. E., "Energy efficient navigation for running legged robots," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2019-May, pp. 6770–6776, 2019.

[18] Pace, J., Harper, M., Ordonez, C., Gupta, N., Sharma, A., and Collins, E. G., "Experimental verification of distance and energy optimal motion planning on a skid-steered platform," *Unmanned Systems Technology XIX*, vol. 10195, p. 1019506, 2017.

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

[20] Sorensen, K. L. and Singhose, W. E., "Command-induced vibration analysis using input shaping principles," *Autom.*, vol. 44, pp. 2392–2397, 2008.

[21] Singer, N. C. and Seering, W. P., "Preshaping Command Inputs to Reduce System Vibration," *Journal of Dynamic Systems, Measurement, and Control*, vol. 112, no. 1, pp. 76–82, 1990.

[22] Singhose, W., Seering, W., and Singer, N., "Residual Vibration Reduction Using Vector Diagrams to Generate Shaped Inputs," *Journal of Mechanical Design*, vol. 116, no. 2, pp. 654–659, 1994.

Albright, Andrew. Bachelor of Science, North Carolina State University, Spring 2020;
Masters of Science, University of Louisiana at Lafayette, Spring 2022
Major: Mechanical Engineering
Title of Thesis: Mechanical Design and Control of Flexible-Legged Jumping Robots
Thesis Director: Dr. Joshua E. Vaughan
Pages in Thesis: 125; Words in Abstract: 185

Abstract

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus nec tellus eget elit aliquet accumsan sit amet in lacus. In hac habitasse platea dictumst. Ut sit amet elit odio. Aenean lobortis mollis metus, sed consequat neque tristique in. Curabitur nec hendrerit metus. Praesent non scelerisque urna, vitae iaculis diam. Aliquam nisl est, imperdiet eu nulla sed, bibendum pulvinar arcu. In ultricies purus purus, vulputate congue justo volutpat ut. Donec nunc magna, rutrum nec turpis et, viverra efficitur lorem. In hac habitasse platea dictumst. Vestibulum maximus lobortis nisl, eget molestie sem sollicitudin nec. Mauris ut enim eu ipsum auctor rhoncus ac vel eros.

Vivamus tincidunt, tortor eu rutrum dapibus, orci turpis porta metus, ac iaculis quam eros sollicitudin nisl. Nam id massa elementum, commodo mi at, lobortis nisl. Fusce vestibulum eu lorem non aliquam. Morbi eleifend tortor id metus elementum, ac tincidunt lorem commodo. Pellentesque vestibulum, erat in tempus vehicula, ex urna auctor leo, ut lobortis eros mauris nec erat. Aliquam erat volutpat. Sed sed pretium risus.

Biographical Sketch

Forrest Montgomery was born in Lafayette, Louisiana for all intents and purposes. He began his academic career at the University of Louisiana with an internal struggle between majoring in Mechanical Engineering or Industrial Design. This thesis is evident of the choice he made. After earning his Bachelor's degree at the University of Louisiana at Lafayette in the Spring of 2015, he joined the CRAWLAB and conducted research in dynamics, controls, and robotics under the tutelage of Dr. Joshua Vaughan. This research culminated with earning a Master's degree in Mechanical Engineering again at the University of Louisiana at Lafayette in the Summer of 2017.