Learning Power Conservative Jumping Strategies for Flexible-Legged Systems ~~or Learning Power Conservative Flexible-Legged Jumping Strategies~~

Andrew Albright * **Joshua Vaughan** **

* *University of Louisiana at Lafayette, Lafayette, LA 70503 USA (andrew.albright1@louisiana.edu)*
** *University of Louisiana at Lafayette, Lafayette, LA 70503 USA (joshua.vaughan@louisiana.edu)*

Abstract: Legged locomotive systems have many advantages over their wheeled counterparts, such as their ability to navigate rough terrain. They have the ability to deploy many ~~navigating~~ techniques to overcome obstacles, one of which is jumping. Still, there are disadvantages to overcome when using legged systems, such as their lack of energy efficiency. To combat this lack of efficiency, flexible links can be used to conserve energy that would otherwise be wasted during locomotion. Furthermore, control methods can be utilized which ~~assist in~~ both a jumping systems ability to jump high and their ability to conserve power. Using reinforcement learning (RL) to create these controllers for flexible-legged jumping systems can lead to designs that outperform, in terms of jumping height, traditional optimization methods for jumping. Furthermore, using a power-conservative reward function to train the agent results in a control strategy which balances jump height and energy usage, leading to more efficient jumping control.

Keywords: Power Efficient, Reinforcement Learning, Flexible Robots, Legged Locomotion, Jumping

1. INTRODUCTION

Legged locomotive robots have many advantages over their wheeled counterparts, for example their ability to more easily navigate harshly uneven terrain (Saranli et al. (2001), Park et al. (2017) Blackman et al. (2018) Seok et al. (2015)). However, there are disadvantages as well, one of which is power consumption. There are several factors that contribute to this disadvantage. For example, in many cases several motors are required to actuate several linkages to move only a few feet, whereas a wheeled system needs only a single motor to turn a wheel to accomplish the same task. Another factor, one that is particularly prevalent when navigating uneven terrain, is the challenge of defining how a walking robot places its feet in such a way that reduces ~~the level of~~ wasted energy ~~by harshly taking steps.~~

In an effort to alleviate the power consumption issues seen when using legged systems ~~to accomplish locomotive tasks such as walking, running and jumping~~, research has been conducted which replaces rigid aspects of said systems with flexible ones. It has been shown that this not only leads to higher performance but also higher efficiency (Sugiyama and Hirai (2004), Galloway et al. (2011), Hurst (2008), Seok et al. (2015)).

While the introduction of flexible components ~~within robotics systems~~ solves some challenges related performance measures like power consumption, it raises other challenges. Modeling the ~~the~~ systems becomes more difficult because the models become highly nonlinear. Different approaches have been taken to mitigate these issues. A popular and successful example being the use of series elastic actuators instead of flexible links (Iida et al. (2005), Ahmadi and Buehler (1997)). Other solutions seek to create control methods which are suited to non-rigid systems. One of those control methods is ~~the use of~~ reinforcement learning (RL), to create controllers based on learned control strategies that are developed through interactions with an environment. These RL controllers ~~both during and after training~~ are called control agents or just agents.

In this work, RL is utilized as a training platform for developing control strategies for a simplified jumping robot. The RL agent is trained to maximize jump height while conserving power. It is shown that when tasked as such, ~~during training,~~ the agent learns unique control strategies to maximize the jumping ~~potential~~ of the system, as well as balancing power usage.

The next section will look at some related work in the fields of RL, flexible systems and power conservative control strategies. ~~Following that,~~ a description of the environment used for ~~trining~~ and evaluating the RL agents will be provided ~~along~~ with a short description of the RL algorithm used for this work. Then a breakdown of the reward functions used to train the agents and the experiments performed will be shown. ~~After which,~~ evaluations ~~will be~~ discussed and conclusive remarks will be made.

2. PREVIOUS WORK

2.1 Reinforcement Learning for Legged Locomotion

Research has shown that using RL for defining control strategies for legged systems is a viable path for simple and even complex legged locomotive systems. Yang et al. (2019) show that using a model-based method can require an order of magnitude less environment interactions comparing to the best performing model-free algorithms to train controlling agents. However, model-based methods expose other challenges. Learning a model often requires many interactions, and the inevitable limited environment that is learned leads to limited exploration during controller training. As such, model-free methods are often of interest as they can be deployed directly on hardware to learn based on interactions with the environment, or in simulation. Peng et al. (2016) showed the viability of such an approach training many different agents to locomote navigating uneven terrain. Reda et al. (2020) showed similar successes using a host of different environments. It addition they were also able to display the apparent difficulties of defining an environment for creating robust controllers.

2.2 Reinforcement Learning for Flexible Systems

As discussed, flexible systems do have some advantages over their rigid counterparts, particularly ones which are used for locomotion tasks. Thuruthel et al. (2019) shows the use of model-based methods for controlling soft-robots where as a part of training, learning the model is required. In contrast to that work, Dwiel et al. (2019) shows that using model-free methods and algorithms such as DDPG Cui et al. (2019) prove effective for both locomotive and non-locomotive tasks where flexible systems are considered. Additionally, comparing RL control strategies to more traditional control strategies such as PD control, He et al. (2020) shows that the comparison results allot that RL is practically applicable for controlling flexible systems. Furthermore, RL control can be used for partial control of flexible systems, for example using it to damp out vibration while a separate controller is used to determine general pose Cui et al. (2019).

2.3 Flexible Robots Improved Performance

Using flexible components within robotic systems has shown great potential for conserving power. Ahmadi and Buehler (1997) showed the results of a common technique which is using flexible joints (series elastic actuators) to increase energy efficiency. Flexible joints are not the only way to increase efficiency, but a similar technique which seeks to emulate the tendons seen in nature can produce similar efficient results Folkertsma et al. (2012). In the work completed by Seok et al. (2015) and Seok et al. (2013), design principles for efficient quadruped locomotion are discussed, and evidence supports the use of flexible components increases efficiency.

2.4 Control for Power Efficiency

In stead of relying solely on the design of a system to increase efficiency, Harper et al. (2019) shows that the use of model predictive control methods can lead to more efficient locomotion strategies. In contrast to studying traditional methods of control to accomplish energy efficient strategies, little work has been done to evaluate the potential of using RL control strategies to directly effect power efficiency.

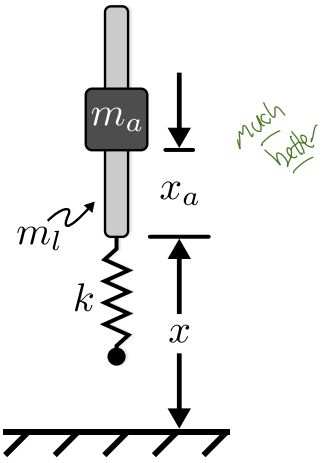

Fig. 1. Pogo-stick Model: A Simplified Flexible-Legged Locomotive Jumping System

3. POGO-STICK ENVIRONMENT

3.1 The pogo-stick environment

Representing a flexible-legged locomotive system is the jumping model shown in Fig. 1. This model can be used as a base representation for many different jumping animals, which in turn represent many different kinds of jumping gaits Blickhan and Full (1993). As such, this model is used as the environment that the control agents are trained and evaluated on to determine if reinforcement learning can define efficient strategies. During training, the agent is tasked with learning a control strategy with the input defined as the acceleration of the mass, m_a, along the rod, m_l.

Variables m_a and m_l represent the masses of the actuator and leg respectively. Because this system is selected as a representation of flexible systems, a nonlinear spring with constant k is used as a flexible component. A damper (not shown in Fig. 1) is present within the model of the system as well and is represented by the variable c. The vertical position relative to the ground is represented by x, and the position which the actuator moves along the rod is represented by x_a. The values of many of these parameters are detailed in Table 1. Note that the system is constrained to move vertically so that the agent does not learn to keep the system balanced.

The equations of motion for the system are:

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

where x and $\dot{x}$ are position and velocity of the rod respectively, $\ddot{x}_a$ is the acceleration of the actuator (as well as the agents action), m_t is the mass of the complete

Table 1. Training and Evaluation Parameters

Model Parameter	Value
Mass of Leg, m_l	0.175 kg
Mass of Actuator, m_a	1.003 kg
Natural Frequency, ω_n	11.13 Hz
Spring Constant, k	200000 N/m
Actuator Stroke, $(x_a)_{\max}$	25 mm
Actuator Velocity, $(\dot{x}_a)_{\max}$	2.0 m/s
Actuator Acceleration, $(\ddot{x}_a)_{\max}$	10 m/s

system, and α is used to piecewise the EOM defining a different EOM depending if the pogo-stick is on or off the ground. α is defined as:

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

Note that the acceleration term related to spring constant, in a nonlinear term. This is is to represent the use of a nonlinear spring.

4. REINFORCEMENT LEARNING

For the experiments in this work, a traditional reinforcement learning environment and setup is utilized. Learning the environment is defined by a Markov Decision Process (MDP) where there exists a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, a transition probability function $p : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$, which is used to define the probability of entering into state s_{t+1} from state s given action a is taken, a reward function $R : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, which defines the reward $R(s_{t+1}, s_t, a_t)$ received by entering into state s_{t+1} from state s_t by taking action a, and a future discounting factor $\gamma \in [0, 1]$ which weights the importance of estimated future rewards. Iteratively stepping through the MPD will generate a policy $\pi_\phi : \mathcal{S} \to \mathcal{A}$ where ϕ are defined so that all $s_0 \in \mathcal{S}$ have a defined maximum $J(\phi) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_{t+1}, s_t, a_t)\right]$, being the expected reward.

The algorithm used to define the weights ϕ is Stable Baselines written version of of Twin Delayed Deep Deterministic Policy Gradient (TD3) (?). This is an actor-critic, off-policy RL algorithm wherein the actor is represented by the policy π_ϕ taking the actions a, and the critic is represented by the estimated expected return of taking action a in state s and following the policy π from there after. The critic is a neural network itself with parameters θ that are updated according to temporal difference error found between a set of twin target networks. These target networks are updated to follow the critic network every defined n updates of the critic network.

5. REWARD FUNCTIONS

5.1 Reward the Agent for Jumping High

To accomplish the task of jumping high, a reward function is constructed which represents the height of the pogo-stick above the ground at any given time step. The signal returned to the agent is normalized with a predefined maximum height so that if the pogo-stick is at or above that maximum, the agent is receiving maximum reward. This reward function is defined as:

$$r = \frac{x_t - x_{min}}{x_{max} - x_{min}} \tag{3}$$

where x_t is the height at the current time step, x_{min} is the minimum height of the system and x_{max} is the max height of the system. Note that x_{max} is set based on expectations gathered from experimental data, and is set to 0.9 meters.

5.2 Reward the Agent for Jumping Efficiently

The second reward function used is one that seeks to balance jumping height with power consumption. To do this, the reward signal returned is defined as the ratio of the current height at a given time step to the power used from time zero to the current time step. The power used at a given time step is defined by:

$$p_t = m_a \, \ddot{x}_{a_t} \, \dot{x}_{a_t} \tag{4}$$

where m_a is the mass of the actuator, $\ddot{x}_{a_t}$ is the acceleration of the actuator at time t, and $\dot{x}_{a_t}$ is the actuators velocity at time t. The efficiency at a given time step is then:

$$e_t = \frac{x_t}{\sum_{t=0}^{t} p_t} \tag{5}$$

This efficiency value is then normalized where the maximum and minimum limits are determined based on experimental realizations of efficiency expectations. The maximum efficiency, e_{max}, is set to 0.002, and the minimum efficiency, e_{min}, is set to 0. The reward function for the case of maximizing agent efficiency can then be written as:

$$r = \frac{e_t - e_{min}}{e_{max} - e_{min}} \tag{6}$$

To receive any reward the agent is required to first get off the ground. Maximizing reward requires the agent to maximize jump height while minimizing power used.

6. EXPERIMENTS

6.1 Define Agents Task

To evaluate if reinforcement learning, and more specifically if the TD3 algorithm (Fujimoto et al. (2018)), can define more efficient control strategies, two types of jumping experiments are performed. For the first, the agent is tasked with learning both high jumping and efficient jumping strategies for single jumps with the pogo-stick environment. This kind of jump can be visualized within Fig. 6.

For the second jumping strategy, the agent is tasked with learning both high jumping strategies and efficient strategies for a jump that is called a stutter jump which can be visualized in Fig. 3. Here, the pogo-stick is allowed to leave the ground once more than in the case of a single jump, allowing it to compress the spring farther and jump higher in the final jump. In this jumping strategy, the actions the agent takes early in a jumping episode have a

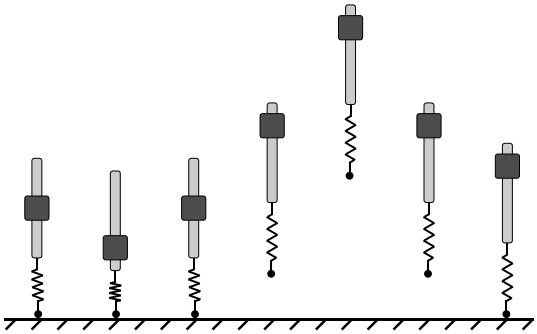

Fig. 2. Pogo-stick Single Jump

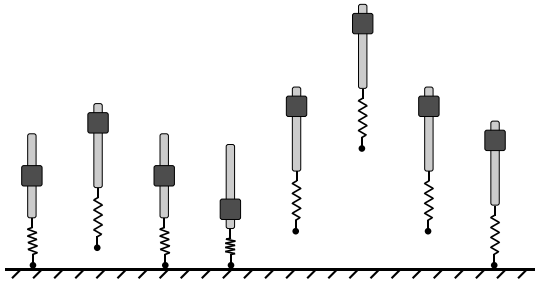

Fig. 3. Pogo-stick Stutter Jump

greater effect on the final jump. This is because the more energy it can store in the spring, the higher it will be able to jump.

6.2 Define Agent Training

To find control solutions for each task, ten different agents are trained on each task individually. The training schedule can be seen in Table 2. Note that the seeds represent the integer seeds passed to the training algorithm which are used to initialize the the neural networks. The same networks are therefore initialized for both high jumping tasks and power conserving tasks.

During training, the agents are alloted 500k steps in their environment, where at each step the agent is presented with the the partial state of the pogo-stick which it uses to take an action in the environment. The state and action can be written as:

$$\mathcal{S} = [x_{a_t}, \dot{x}_{a_t}, x_t, \dot{x}_t] \tag{7}$$

$$\mathcal{A} = [\ddot{x}_{a_t}] \tag{8}$$

where, x_t, $\dot{x}_t$, x_{a_t} and $\dot{x}_{a_t}$ are the positions and velocities of the rod and actuator respectively, and $\ddot{x}_{a_t}$ is the acceleration of the actuator.

The environments are initialized so that the pogo-stick's starting state is according to Table 3. The rods starting position is determined by the amount the spring is compressed according to the total weight of the system. The actuators starting position is at the midpoint of its stroke so that it is free to move in either direction.

Episode terminations are defined based on the two different tasks the agent is to accomplish. The first episodic termination stipulation is after the agent completes a

Table 2. Agent Training Schedule

Task	Jump Type	# Agents	Seeds
Jump High	Stutter Jump, One Jump	10 per Jump Type	9, 16, 104, 107, 250,
Conserve Power	Stutter Jump, One Jump	10 per Jump Type	676, 767, 868, 878, 918, 947

Table 3. Pogo-stick Initial State

State Variable	Value
Position of rod, x	$-\left[\frac{m_t\, g}{k}\right]^{1/3}$ mm
Velocity of rod, $\dot{x}$	0.0 m/s
Acceleration of rod, $\ddot{x}$	0.0 m/s^2
Position of actuator, x_a	$\frac{1}{2}(x_a)$max mm
Velocity of actuator, $\dot{x}_a$	0 m/s
Acceleration of actuator, $\ddot{x}_a$	0 m/s^2

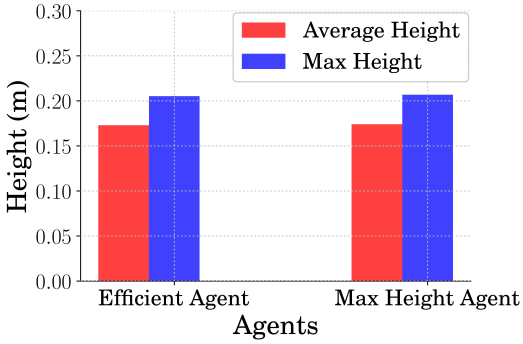

Fig. 4. Single Jumping: Average & Max Heights for both High Jumping and Power Conserving Agents

single jump. This is defined as the rod's position being greater than zero, then returning to zero. Here, the episode is terminated and the environment's state is reinitialized according to Table 3. The second type of episode termination is based on the stutter jump motion. This is allowing the rod's position to be greater than zero twice and then returning to zero. Note that if the two defined terminations do not occur, the episode is terminated after 500 time steps, or 5 seconds.

7. AGENT EVALUATION

During the evaluation of the 40 different agents that were trained, it is apparent that some of them did not learn good control strategies. Some did not learn how to actuate the system in a way which is required to launch the pogo-stick into the air. However, many did learn strategies, and of the ones that did, it is apparent that the efficient agents learned more efficient control methods.

8. CONCLUSION

* Using RL leads higher jumping robot control designs * Using RL also leads to more energy efficient controller designs * Mention future work giving RL access to the design parameters to further optimize jump height and power consumption

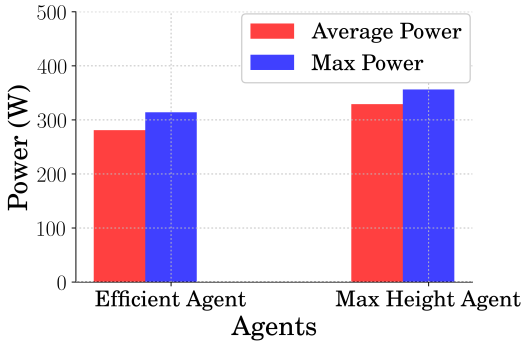

Fig. 5. Single Jumping: Average & Max Power Usage for both High Jumping and Power Conserving Agents

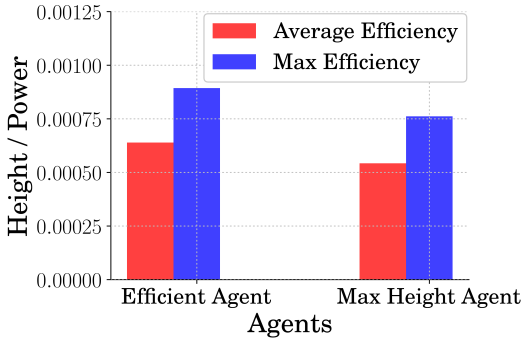

Fig. 6. Single Jumping: Average & Max Efficiency for both High Jumping and Power Conserving Agents

ACKNOWLEDGEMENTS

Thanks to the Louisiana Crawfish Board for their support in the CRAWLAB. It is through their support that research like this is possible.

REFERENCES

Ahmadi, M., Buehler, M., 1997. Stable control of a simulated one-legged running robot with hip and leg compliance. IEEE Transactions on Robotics and Automation 13 (1), 96–104.

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

Blickhan, R., Full, R. J., 1993. Similarity in multilegged locomotion: Bouncing like a monopode. Journal of Comparative Physiology A 173 (5), 509–517.

Cui, L., Chen, W., Wang, H., Wang, J., 2019. Control of Flexible Manipulator Based on Reinforcement Learning. Proceedings 2018 Chinese Automation Congress, CAC 2018, 2744–2749.

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

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

Fujimoto, S., Van Hoof, H., Meger, D., 2018. Addressing Function Approximation Error in Actor-Critic Methods. 35th International Conference on Machine Learning, ICML 2018 4, 2587–2601.

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

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

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

Hurst, J., 2008. The Role and Implementation of Compliance in Legged Locomotion. The International Journal of Robotics Research 25 (4), 110.

Iida, F., Gómez, G., Pfeifer, R., 2005. Exploiting body dynamics for controlling a running quadruped robot. 2005 International Conference on Advanced Robotics, ICAR '05, Proceedings 2005, 229–235.

Park, H. W., Wensing, P. M., Kim, S., 2017. High-speed bounding with the MIT Cheetah 2: Control design and experiments. International Journal of Robotics Research 36 (2), 167–192.

Peng, X. B., Berseth, G., Panne, M. V. D., 2016. Terrain-Adaptive Locomotion Skills Using Deep Reinforcement Learning 35 (4), 1–12.

Reda, D., Tao, T., van de Panne, M., 2020. Learning to Locomote: Understanding How Environment Design Matters for Deep Reinforcement Learning.

Saranli, U., Buehler, M., Koditschek, D. E., 2001. RHex: A Simple and Highly Mobile Robot. International Journal of Robotics Research 20 (7), 616–631.

Seok, S., Wang, A., Chuah, M. Y., Hyun, D. J., Lee, J., Otten, D. M., Lang, J. H., Kim, S., 2015. Design principles for energy-efficient legged locomotion and implementation on the MIT Cheetah robot. IEEE/ASME Transactions on Mechatronics 20 (3), 1117–1129.

Seok, S., Wang, A., Chuah, M. Y., Otten, D., Lang, J., Kim, S., 2013. Design principles for highly efficient quadrupeds and implementation on the MIT Cheetah robot. In: Proceedings - IEEE International Conference on Robotics and Automation. IEEE, pp. 3307–3312.

Sugiyama, Y., Hirai, S., 2004. Crawling and jumping of deformable soft robot. 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 4 (c), 3276–3281.

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

Yang, Y., Caluwaerts, K., Iscen, A., Zhang, T., Tan, J., Sindhwani, V., 2019. Data efficient reinforcement learning for legged robots. arXiv (CoRL), 1–10.