Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Robotic Systems Lab
Prof. Dr. Marco Hutter

Semester Thesis

Design Optimization for Series Elastic Actuators on Quadrupedal Robots

Autumn Term 2023

Supervised by:
Fabian Tischhauser
Filip Bjelonic

Author:
Sujet Phodapol

Declaration of Originality

I hereby declare that the written work I have submitted entitled

Design Optimization for Series Elastic Actuators on Quadrupedal Robots

is original work which I alone have authored and which is written in my own words.[1]

Author(s)

Sujet	Phodapol

Student supervisor(s)

Fabian	Tischhauser

Committee members(s)

Filip	Bjelonic

Supervising lecturer

Marco	Hutter

With the signature I declare that I have been informed regarding normal academic citation rules and that I have read and understood the information on 'Citation etiquette' (`https://www.ethz.ch/content/dam/ethz/main/education/rechtliches-abschluesse/leistungskontrollen/plagiarism-citationetiquette.pdf`). The citation conventions usual to the discipline in question here have been respected.

The above written work may be tested electronically for plagiarism.

Zürich, 30.01.2023	[illegible]
Place and date	Signature

[1]Co-authored work: The signatures of all authors are required. Each signature attests to the originality of the entire piece of written work in its final form.

Intellectual Property Agreement

The student acted under the supervision of Prof. Hutter and contributed to research of his group. Research results of students outside the scope of an employment contract with ETH Zurich belong to the students themselves. The results of the student within the present thesis shall be exploited by ETH Zurich, possibly together with results of other contributors in the same field. To facilitate and to enable a common exploitation of all combined research results, the student hereby assigns his rights to the research results to ETH Zurich. In exchange, the student shall be treated like an employee of ETH Zurich with respect to any income generated due to the research results.
This agreement regulates the rights to the created research results.

1. Intellectual Property Rights

1. The student assigns his/her rights to the research results, including inventions and works protected by copyright, but not including his moral rights ("Urheberpersönlichkeitsrechte"), to ETH Zurich. Herewith, he cedes, in particular, all rights for commercial exploitations of research results to ETH Zurich. He is doing this voluntarily and with full awareness, in order to facilitate the commercial exploitation of the created Research Results. The student's moral rights ("Urheberpersönlichkeitsrechte") shall not be affected by this assignment.
2. In exchange, the student will be compensated by ETH Zurich in the case of income through the commercial exploitation of research results. Compensation will be made as if the student was an employee of ETH Zurich and according to the guidelines "Richtlinien für die wirtschaftliche Verwertung von Forschungsergebnissen der ETH Zürich".
3. The student agrees to keep all research results confidential. This obligation to confidentiality shall persist until he or she is informed by ETH Zurich that the intellectual property rights to the research results have been protected through patent applications or other adequate measures or that no protection is sought, but not longer than 12 months after the collaborator has signed this agreement.
4. If a patent application is filed for an invention based on the research results, the student will duly provide all necessary signatures. He/she also agrees to be available whenever his aid is necessary in the course of the patent application process, e.g. to respond to questions of patent examiners or the like.

2. Settlement of Disagreements

Should disagreements arise out between the parties, the parties will make an effort to settle them between them in good faith. In case of failure of these agreements, Swiss Law shall be applied and the Courts of Zurich shall have exclusive jurisdiction.

Zürich, 13.09.2022
Place and date

สุเจตน์ ไพดาพล
Signature

Contents

Preface

First of all, I would like to express my gratitude to the Robotic System Lab (RSL) for giving me the opportunity to do research on this interesting topic. I especially would like to thank my supervisors, Fabian Tischhauser and Filip Bjelonic. They provide me with a lot of helpful feedback and suggestions during the project, which helps me to work get the most out of this project.

Zurich, January 2023
Sujet Phodapol

Abstract

Legged robots have been using series elastic actuators for a long time. The compliance component shows the potential for several advantages, such as energy-efficient locomotion and the potential for faster motion. However, one of the primary difficulties is how to control and choose the appropriate design parameter for the given task. Model-based approaches are widely used to address this issue. Reinforcement learning, however, has not yet received much attention. In this project, we propose a learning-based design optimization that takes account of the design parameter optimization along with the control policy at the same time. We develop two simulatable models which integrated series elastic actuators into the existing model at the Knee Flexion/Extension joint of a small quadrupedal robot. Then, we set up a learning environment with suitable reward functions to get a functional control policy. Our results show that using this design optimization framework, it is possible to obtain the optimal spring constant and controller for a given task. With this optimal design, the maximum velocity of the robot with the spring is increased by 34.0%, while mechanical power and actuator loss are reduced by 26.6% and 54.1%, respectively.

Symbols

Acronyms and Abbreviations

HAA Hip Abduction/Adduction.

HFE Hip Flexion/Extension.

KFE Knee Flexion/Extension.

PD Proportional–Derivative.

PPO Proximal Policy Optimization.

RL Reinforcement Learning.

RSL Robotic System Lab.

SEA Series Elastic Actuators.

Chapter 1

Introduction

1.1 Motivation

Series Elastic Actuators (SEA) have been used in legged robot systems for years and frequently exhibit a number of benefits, including energy-efficient locomotion [1, 2, 3, 4, 5]. This elastic actuator is the actuator that, as illustrated in Fig. 1.1, connects the elastic element and load in a serial connection with the conventional actuator (i.e., the motor and gearbox). With this configuration, states at the gearbox's output can be decoupled from the states of the load. This elastic component is also anticipated to facilitate impact absorption and energy storage, potentially resulting in energy-efficient locomotion.

However, by introducing this elastic component into the robot, it will be challenging to create the controller due to bandwidth restrictions, especially with the conventional model-based control approach, leading to a trade-off between complexity and controllability [6]. On the other hand, the development of learning-based controllers, such as Reinforcement Learning (RL), shows that these controllers have the ability to handle complicated systems, particularly legged robots [7, 8]. As a result, we expect that these controllers will enable us to develop a functional controller that is optimal for the compliant robot.

In this thesis, we propose a design optimization pipeline using a Reinforcement Learning controller in the Isaac Gym simulator [9]. With this framework, we build the controller while optimizing design parameters to produce the best design parameter and control policy for the given task. In this project, a small quadrupedal robot named Minimal will be used as the testing robot.

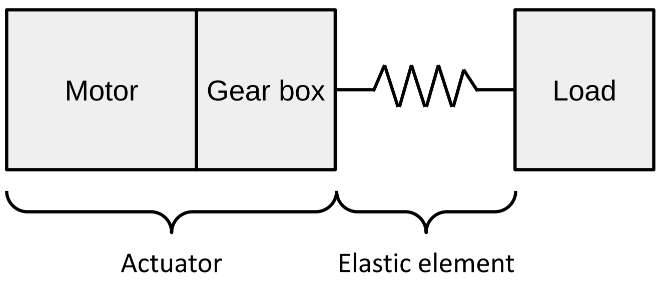


Figure 1.1: The basic configuration of SEA consists of the actuator, elastic element and load.

1.2 Problem formulation

The goal of this thesis is to obtain the best spring constant for a small quadrupedal robot, namely Minimal (a mostly 3D-printed robot as shown in Fig. 1.2). This robot has 12 degrees of freedom in total with 3 joints on each leg: the Hip Abduction/Adduction (HAA), Hip Flexion/Extension (HFE), and Knee Flexion/Extension (KFE). Minimal's total mass is 4.3 kg. The specification provides the actuator's maximum torque and velocity limits, thus we need to include these constraints in the model. We integrate the SEA into only the KFE joints which we hypothesize will have the biggest impact on the performance of the running of the robot. As a result, this joint will not have a position limit on the robot, unlike other joints (such as HAA and HFE) that do. Constraints of the robot can be summarized in Tab. 1.1.

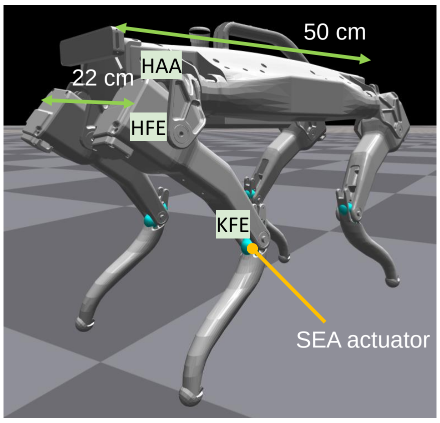


Figure 1.2: Minimal with the SEA integration on KFE joints.

Constraints	**Value**
Maximum actuator velocity	30 rad/s
Maximum actuator torque	4 Nm
HAA position limit	-30 - 30 deg
HFE position limit	8 - 80 deg

Table 1.1: These robot's constraints will be used as the limitations for creating models and learning environments in the simulation.

Since we hypothesize that SEA could demonstrate the benefit of potentially faster motion according to the decoupling between the joint velocity and the actuator velocity, the goal of this design is to design a robot that can run as quickly as possible on flat terrain within the boundaries of the given constraints while utilizing the least amount of power.

1.3 Related work

Design optimization is a time-consuming and complicated process in robot design because it is an iterative process and, in many cases, it may be difficult to get trivial solutions. Additionally, modifying the design typically comes at a high expense. Typically, the hardware design and controller design processes are separated during the development process. To clarify, we optimize the design parameter based on expert knowledge and constraints before later designing the controller based on this designed parameter. The works from Bolívar-Nieto et al. [10, 11] demonstrate how to design a series elastic actuator using a convex optimization framework that can best choose the right spring size given the actuator's limitations. However, with this framework, we may obtain the ideal design parameter for the actuator, but not for the robot. The best actuator design could not be the best design for the entire robot locomotion system.

On the other hand, researchers propose a number of methods to merge the design of the hardware and control policy together thanks to the advantages of learning and simulation frameworks. With this co-optimization, we will obtain the optimal parameter together with the optimal controller. One framework is to integrate the design parameter into the learning policy, namely hardware as policy [12]. In this approach, the hardware parameter will be learned by the policy as a neural network parameter. Thus, during the learning process, the hardware parameter can be directly optimized. However, the control policy from this co-optimization framework may not be able to extend to the new robot design (i.e. changing design parameters). For instance, in order to obtain a working policy for new design parameters, we need to train the robot from scratch once more if we create a new robot. Another work suggests an approach to deal with this re-compute issue by sampling the robot and parameterising the design features into the controller [13]. This method enables the policy to control all the variations in the design space.

The work from Belmonte-Baeza et al. [14] demonstrates a successful end-to-end solution for a model-free meta-reinforcement learning design optimization framework to obtain the ideal leg length for the quadrupedal robot. This framework performs better than the model-based method since it is not limited by any predefined gait patterns. The research from Bjelonic et al. [15] also displays the co-optimised design framework, which is capable of designing an appropriate parallel elastic knee joint along with the locomotion controller for a quadrupedal robot. By implementing the designed actuator and evaluating it in real-world experiments, they also highlight the value of this framework.

In this thesis, we model and analyze the performance of the robot with SEA and then compare it to the robot without SEA on a small quadrupedal robot. This project focuses primarily on developing the model of the robot equipped with this actuator in the simulation. Then the learning environment is set up such that the control policy is able to learn to control the robot in the design parameter space. This framework will optimize the spring constant along with a functional controller for a particular range of parameters. With this policy, we can evaluate and determine the optimal design parameter for the given task.

Chapter 2

Method

2.1 Overview

The workflow of the optimization process can be summarized in Figure 2.1. To incorporate SEA into the current model, a stable simulation model needs to first be built. The full model and the simplified model are the two models that will be presented in the next section. Second, a reward function that is appropriate for the given task is shaped based on the model in order to make the robot learn to walk. The reward function and hyperparameters will be tuned. In order to determine the best parameter (i.e. spring constant), the working control policy will be evaluated with the particular task. The outcome is then compared to the benchmark (robot without SEA).

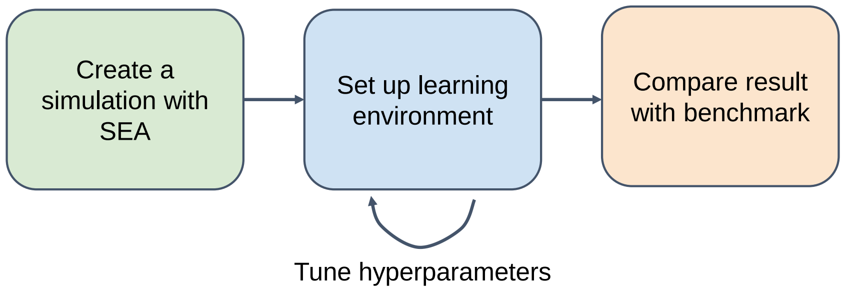

Create a simulation with SEA
Set up learning environment
Compare result with benchmark
Tune hyperparameters

Figure 2.1: Overview of the workflow.

2.2 Create a simulation with SEA

In this section, we examine the SEA mathematical model and methods for incorporating the SEA into the current simulation model. Two models, the full model and the simplified model, will be examined. The full model is a model with an additional link to approximate SEA dynamics, while the simplified model is a model with an assumption on perfect position control for the motor.

2.2.1 Full model

The goal of the full model is to create an approximation of the real SEA constrained by the available simulator. First, we add an additional third link (rotor) in between these two links as illustrated in Figure 2.2. Second, by assuming a linear torsional spring attached between the rotor and shank, we can calculate the spring torque (τ_s) from the angle difference between the rotor angle (θ_{rotor}) and shank angle (θ_{shank}) as shown in Eq. 2.3. The joint between the rotor and shank is then driven by this computed spring torque, while the joint between the thigh and rotor is driven by the motor torque (τ_m). These torques are then sent to each joint and the dynamics of the system are solved inside the rigid body simulation. Proportional–Derivative (PD) controller is used to control the motor. The torque interaction between the body can be derived as the equation of motions as follows:

$$I_{rotor}\ddot{\theta}_{rotor} = \tau_m - \tau_s - b\dot{\theta}_{rotor} \tag{2.1}$$

$$\Delta\theta = \theta_{shank} - \theta_{rotor} \tag{2.2}$$

$$\tau_s = k_{spring}\Delta\theta \tag{2.3}$$

Where I_{rotor} is the moment of inertia of the rotor, we can estimate the value from the actual motor using assumptions about the rotor's geometry and gearbox ratio. $\dot{\theta}_{rotor}, \ddot{\theta}_{rotor}$ are the angular velocity and angular acceleration of the rotor. k_{spring} is the spring constant between the rotor and shank. b is the viscous friction between the rotor and thigh, which we assume to be zero.

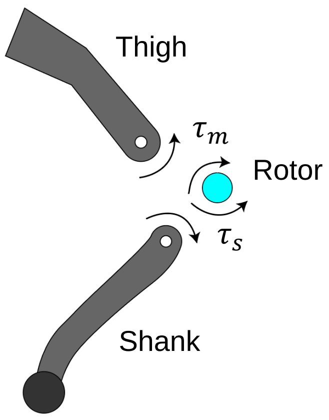


Figure 2.2: Free body diagram of the full model showing SEA integration in the KFE joint.

2.2.2 Simplified model

Although we assume the full model to be more accurate at representing the real SEA, there are several parameters that we need to assume in this model (i.e. PD controller gains, rotor inertia (I_{rotor}) and viscous friction (b)). Because SEA should be able to decouple the motor and joint (i.e. the velocity of the motor and the joint after the spring can be different), we simplify the model by making further simplifications. To reduce the number of design parameters, we make the assumption that the motor at KFE has the perfect position control. As a result, the action from the neural network in the simplified model is the position of the motor (θ_{motor}) instead of the target position for the motor as in the full model. With this assumption, we will have a very stiff motor. Additionally, by eliminating the rotor's dynamic, this assumption will make the motor torque and spring torque to be equal. (i.e. $\tau_{motor} = \tau_{spring}$) by deriving the following equations:

$$I_{rotor}\ddot{\theta}_{rotor} = \tau_m - \tau_s - b\dot{\theta}_{rotor} \tag{2.4}$$
$$I_{rotor}\ddot{\theta}_{rotor} = b\dot{\theta}_{rotor} = 0 \tag{2.5}$$
$$\tau_m = \tau_s \tag{2.6}$$
$$\tau_s = k_{spring}(\theta_{motor} - \theta_{joint}) \tag{2.7}$$

Where θ_{joint} is the angle of the shank with respect to the thigh. We need different methods to incorporate the actuator limitations into the simplified model because we assume perfect position tracking here. First, we constrain the maximum torque by forcing the actuator position to not deviate from the joint position beyond the maximum target. This maximum target can be computed directly from the maximum torque and the spring constant. For instance, nothing happens if the actuator position remains inside the maximum target. The maximum target position will be adjusted, on the other hand, if the actuation position is outside of the maximum target as shown in Fig. 2.3.

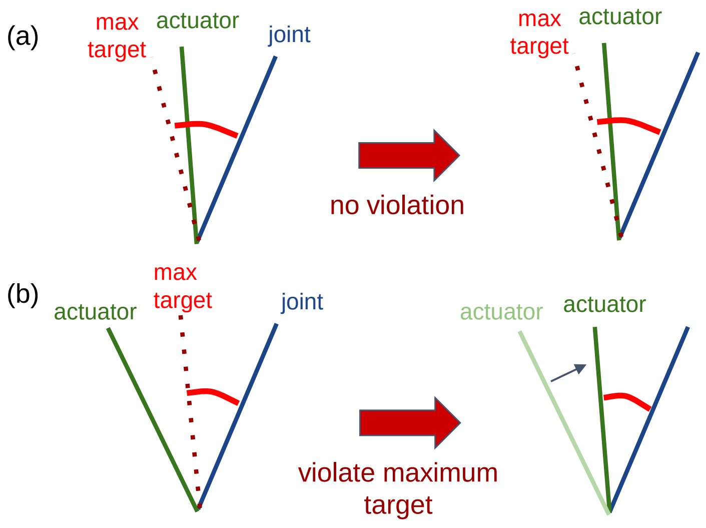


Figure 2.3: The figure shows the adjusting mechanism to include torque limitation into the simplified model.

For the velocity limit, we constraint the actuator by using the negative reward function to penalize the velocity that exceeds the maximum velocity limit (this will be explained in detail in section 2.3). The actuator velocity is computed from the previously updated actuator position and the current actuator position.

In this project, we will use the simplified model to represent the robot with SEA. The main difference between the full model and the simplified model will be explained more in detail with the step response in chapter 3.1.

2.3 Learning environment

In this section, we set up the learning environment to train the robot to learn how to walk in accordance with our objective task. We built up a simulator that can create the robot with various spring constants in order to obtain a single control policy that can be used for the full range of design parameters space. Then, we will train this policy using the Proximal Policy Optimization (PPO) [16] algorithm.

First, we use the Isaac Gym simulator [9] as a multi-body physics simulator for the training environment. Since our policy goal is to control the locomotion of robots within a range of design parameters (i.e., a spring constant), we modify this learning environment such that it spawns thousands of simplified model robots with different spring constants and initial states. In each robot, the spring constant is the same for all KFE joints. Every time a robot respawns, this spring constant is changed.

Second, in order to achieve our objective of designing the optimal design parameter for a robot, we need to incorporate the objective into the reward functions. Therefore, the first necessary reward function is one that encourages the robot to run as quickly as possible in the x-axis direction. We can be defined this reward to be linearly proportional to the velocity of the robot ($v_{x-robot}$) as follows:

$$r := v_{x-robot} \tag{2.8}$$

According to experiments, if we choose this reward to be an absolute value or the square of the velocity, it may cause a robot to move its body back and forth instead of running forward.

The following reward functions are designed in order to minimize energy loss, thus they are negative rewards. We define two energy losses: mechanical power and actuator loss. The proportion of power exerted by actuators is used to determine mechanical power as follows:

$$r := -\sum_{i=1}^{4} \tau_i \times \omega_{actuator} - \sum_{j=1}^{8} \tau_j \times \omega_{joint}; i \in \text{joint with SEA}, j \in \text{joint without SEA} \tag{2.9}$$

Where, for a joint with SEA, $\omega_{actuator}$ is computed from the difference between the current actuator position and the previous actuator position (i.e. action and previous action from the control policy). For other joints, ω_{joint} can be measured from the simulation. τ is the torque exerted in each joint.

Actuator loss is the loss from the current in the actuator, which can be estimated from the square of each actuator's torque as follows:

$$r := -\sum_{i=1}^{12} \tau_i^2; i \in \text{joint} \tag{2.10}$$

Another important reward is to add constraints to the maximum velocity of actuators. This reward is designed to be negative if the velocity of the actuator exceeds the maximum velocity limit and be zero if the constraint is not violated.

$$r := \begin{cases} -||\omega_{actuator} - \omega_{actuator(max)}||^2, & |\omega_{actuator}| \geq |\omega_{actuator(max)}|. \\ 0, & \text{otherwise.} \end{cases}$$

With this reward function, we can impose velocity constraints into the simplified model. (More details on other reward functions, observation and the values of their parameters can be seen in Appendix A.)

The following step is to train control policy using RL. The PPO algorithm is used with mentioned reward functions. Outputs of the control policy are target joint position for HAA and HFE joints, while they are actuator position for KFE joint. Our initial goal is to find a single control policy, which can control a robot with design parameters ranging from a very soft to very stiff spring (e.g. a range from $k = 2 - 100$ Nm/rad). We make the assumption that a very stiff spring can act as an approximation for a stiff robot (i.e., a robot without SEA (benchmark)). However, we are unable to obtain the functional control policy for this design space. The reasons might be the limitation of the simulation and also the depth of the neural network. As a result, we truncate our design space to the range from $k = 2 - 10$ Nm/rad. We then train one control policy for this design space and define the robot with SEA in this truncated range as the soft robot. For the benchmark, we use the same reward functions to train another control policy for the robot without SEA (i.e., a stiff robot). In other words, two neural network control policies are trained, one for the soft robot and another for the stiff robot.

Chapter 3

Results and Discussion

In this chapter, we verify the model's accuracy and assess how well the soft robot, a robot with SEA, performs the objective task over a range of spring constants. We will start by comparing the differences in the robot's locomotion in the design parameter range and obtain the optimal design parameter in the given range. The robot will next be assessed using the benchmark for the fastest running on flat terrain. Three main objectives will be observed: maximum velocity in the x-direction, mechanical power and actuator loss.

3.1 Single-Leg Model

The experiment is set up by fixing the base to float in the air and then sending a step command to the KFE joint. The analytical solution allows us to directly calculate the oscillation motion's period (T) as follows:

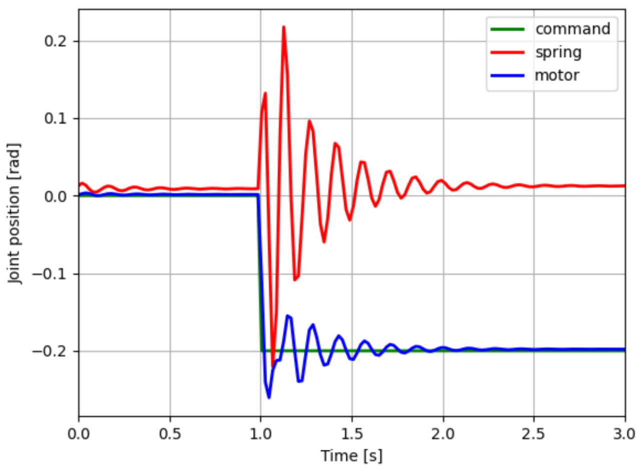
$$T = \sqrt{(k/I_{total})} \tag{3.1}$$

$$I_{total} = I_{cm(shank)} + m_{shank} l_{shank}^2 + I_{cm(foot)} + m_{foot} l_{foot}^2 \tag{3.2}$$

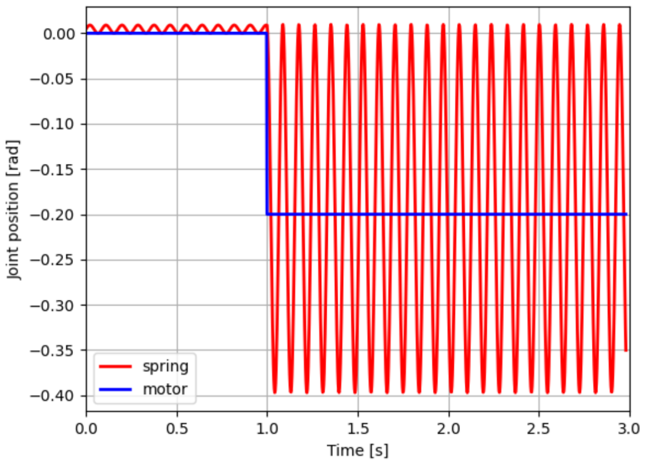
Where k is the spring constant. I_{total} is the total moment of inertia of the leg (i.e., shank and foot) around KFE joint. $I_{cm(\cdot)}$ denotes the moment of inertia around the center of mass, $m_{(\cdot)}$ represents the mass and $l_{(\cdot)}$ is the length from the centre of mass to the KFE joint

From the actual robot's parameters, we can compute $I_{total} = 4.01 \times 10^{-4} kgm^2$ for Minimal. From the analytical solution, we expect to observe the step response in the harmonic motion around the equilibrium with a stiff position controller. If the result shows spring behaviour with an oscillation period that is identical to the analytical result, we can confirm that the simulation is stable.

With this validation setup, we can simulate the behaviour of the full and simplified model. As shown in Fig. 3.1, we can observe the different responses between the two models. Since the full model includes rotor dynamics in the dynamics of the system, the actuator tries to control the position to track the step command, resulting in damping out the spring oscillation as shown in Fig. 3.1a. On the other hand, with the assumption of the simplified model, we are not concerned with the rotor dynamics, which will simplify the problem. For this reason, the simplified model is employed in this thesis.



(a) Full model



(b) Simplified model

Figure 3.1: The difference between the step response on a single leg of the full and simplified model.

3.2 Optimal design

In this section, the trained control policy from the design optimization framework is implemented on robots with different design parameters and assessed it using an objective task. In order to obtain the best design for this particular task, robots throughout the design space are sampled. Following that, the optimal robot with SEA and the benchmark with respect to the objective function will be compared.

Robots are created using various spring constants, and the objective task is to make them run in the x-direction until it reaches a steady state. The highest velocity, amount of mechanical power used, and actuator loss are then measured. As shown in Fig. 3.2 all the soft robots can run faster than the stiff robot. The result also shows that the optimal spring constant is 4.25 Nm/rad with a maximum velocity of 6.07 m/s (increasing by 34% from the benchmark (4.53 m/s)). Additionally, soft robots also demonstrate a lower consumption compared with stiff robots. From Fig 3.3, a robot with the optimal design parameter (i.e the fastest robot) can decrease the consumption in mechanical power and actuator loss by 26.6 % and 54.1%, respectively.

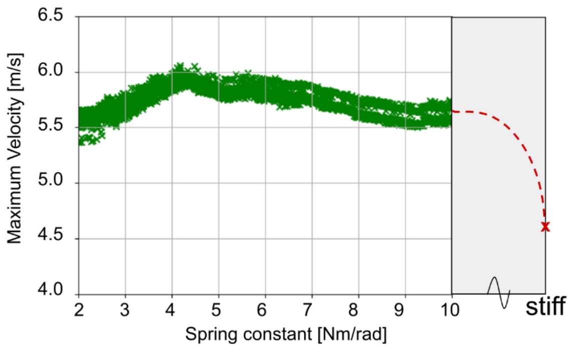
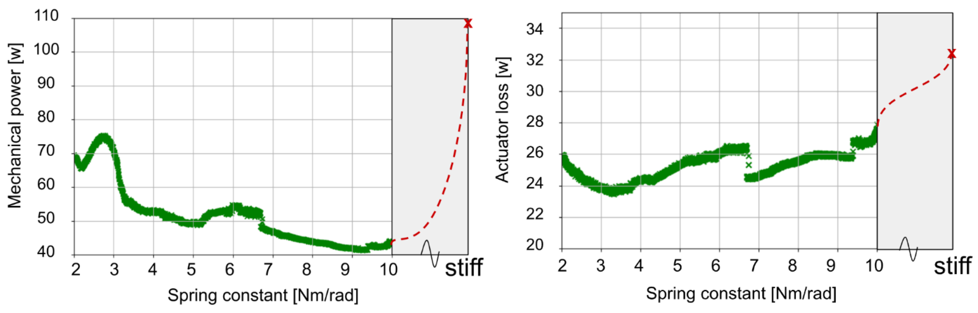


Figure 3.2: The maximum velocity of various spring constants



(a) Mechanical power (b) Actuator loss

Figure 3.3: The power consumption of various spring constants

Since our control policy works only in our design range (a range from $k = 2 - 10$ Nm/rad), the optimal parameter we obtain is only optimal within this range. It is also possible that the optimal value may not be the one we obtain but rather one that lies in the area beyond our design range, the grey area.

3.3 Locomotion analysis

In this section, we observe the behaviour of the robot in the design space. There are three spring constants in this investigation: 2, 10 (boundary cases) and 4.25 (optimal design) Nm/rad. As illustrated in Fig. 3.4, We can observe the gait difference between the robot with SEA and without SEA (benchmark). The former demonstrates pronking gait while the latter exhibits a trotting gait. The main difference in motion between the boundary cases can also be observed in the swing phase, where the softer case ($k = 2$ Nm/rad) pulls the front legs' shanks farther to the rear (exceeding the centre point), whereas the stiffer case ($k = 10$ Nm/rad) only moves the shanks in a smaller range in the front. As shown in Fig. 3.5, the optimal design runs with a motion that is more similar to the softer one.

As illustrated in Fig. 3.6a and Fig. 3.7a, we can observe that the softer one ($k = 2$ Nm/rad) fully exploits the maximum allowed velocity, while it does not fully utilize the maximum allowed torque. For the stiffer one ($k = 10$ Nm/rad), the opposite behaviour is observed as shown in Fig. 3.6c and Fig. 3.7c. For the optimal design, the actuator is optimized to reach both the maximum velocity and the maximum torque limitations as displayed in Fig. 3.6b and Fig. 3.7b, resulting in the fastest locomotion.

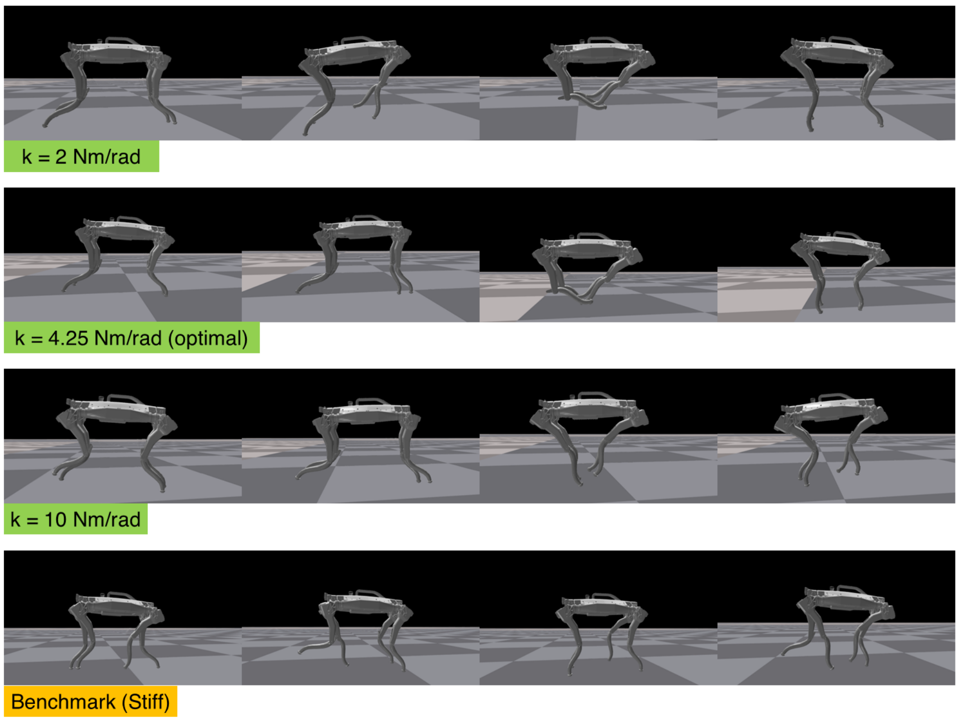
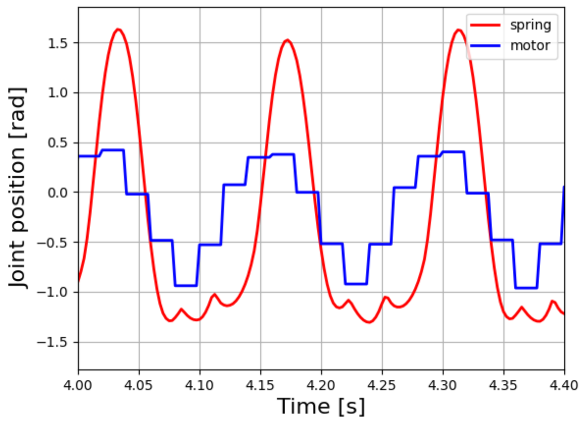
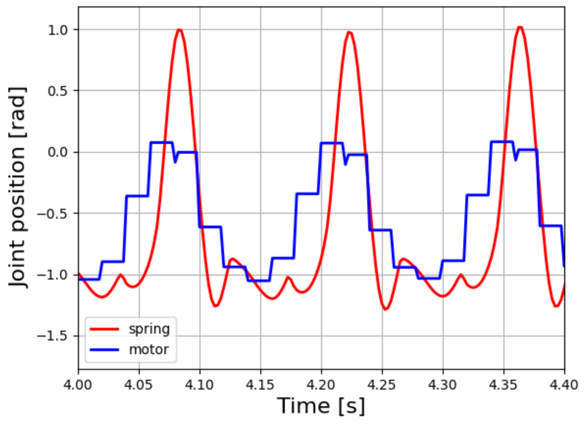


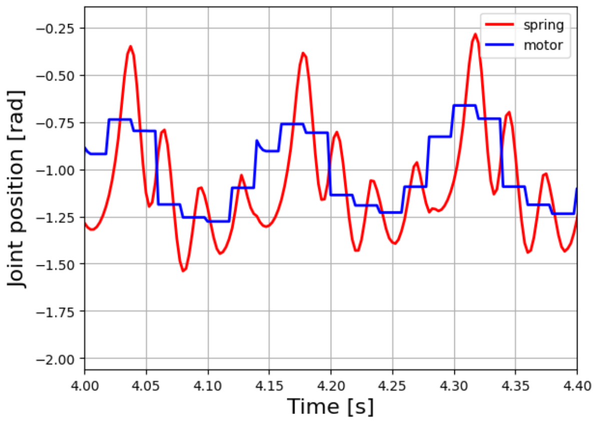
Figure 3.4: The gait of each robot with different spring constants in one cycle.



(a) $k = 2$ Nm/rad

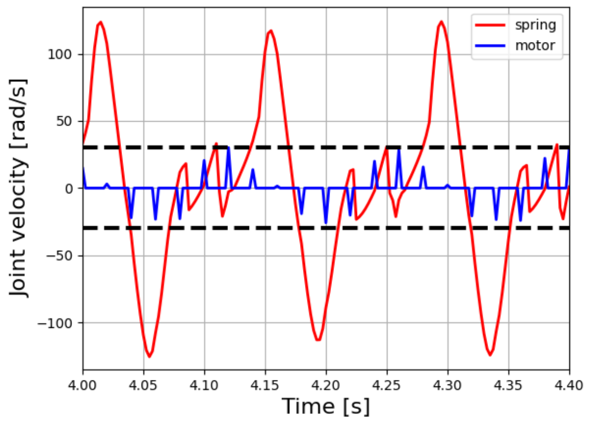


(b) $k = 4.25$ Nm/rad

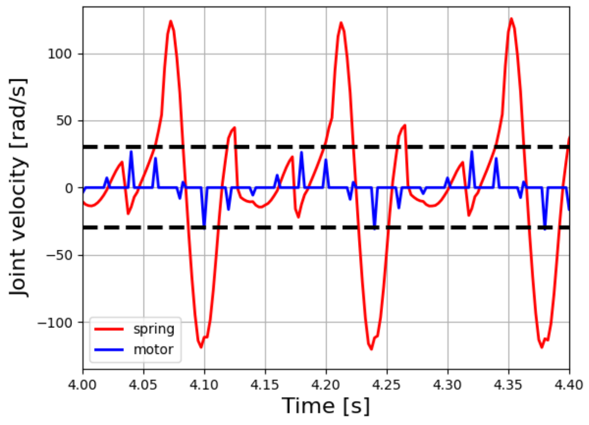


(c) $k = 10$ Nm/rad

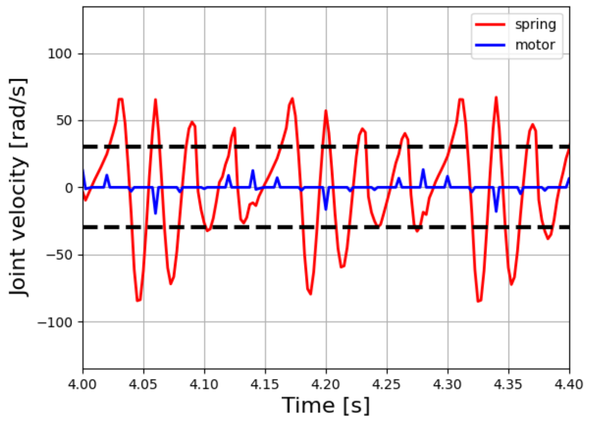
Figure 3.5: The KFE joint position for different spring constant (forward direction is a negative value and the backward direction is positive).



(a) $k = 2$ Nm/rad

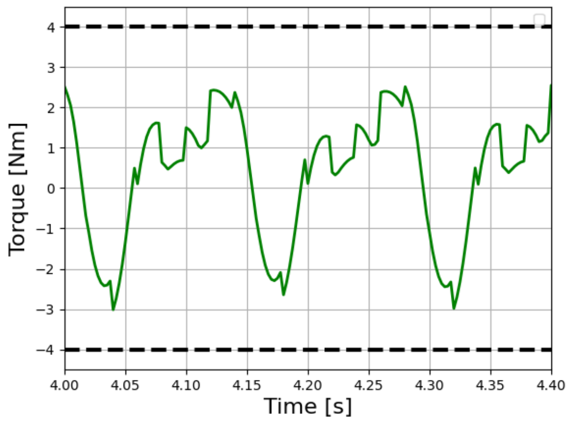


(b) $k = 4.25$ Nm/rad

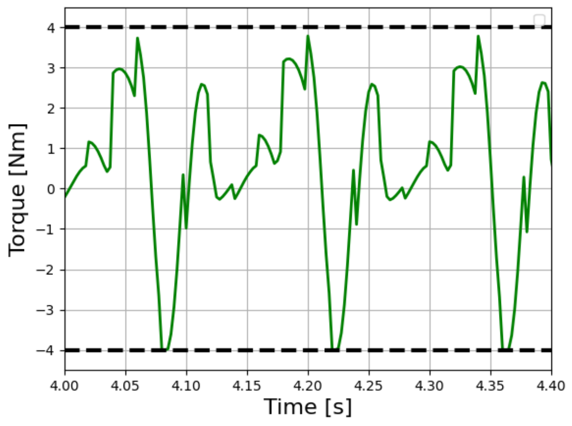


(c) $k = 10$ Nm/rad

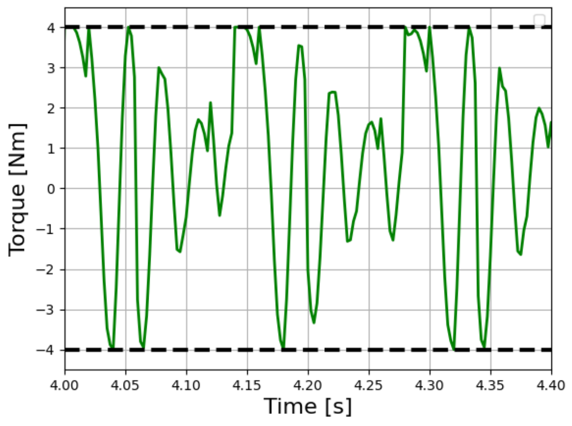
Figure 3.6: The KFE joint velocity for different spring constant (forward direction is a negative value and the backward direction is positive).



(a) $k = 2$ Nm/rad



(b) $k = 4.25$ Nm/rad



(c) $k = 10$ Nm/rad

Figure 3.7: The KFE joint torque (similar to spring torque) for different spring constant.

Chapter 4

Conclusion

In this project, We develop two simulatable models which introduce SEA into a small quadrupedal robot. We also provide a learning environment setup and suitable reward function to train the robot on a given task (i.e. run as fast as possible on flat terrain with minimum energy consumption). The result can show the advantages of SEA for both faster locomotion and also a decrease in power consumption.

The main challenge in this project comes from the model development. Both full and simplified models have been examined. The full model represents the full dynamics of the SEA by including the additional body in the joint. However, from the preliminary investigation, there are a number of assumptions and also parameter estimations that we need to concern about in this model, including rotor and gearbox inertia, friction parameters and controller gain. On the other hand, by assuming perfect motor position control, the simplified model reduces the number of design parameters. With this model, we can obtain a working control policy from RL that functions with a variety of design parameters, enabling us to apply this control policy to evaluate the given task and provide the optimal design. However, there are some limitations in this simplified model in that we update the position of the actuator in the control policy time which causes the instantaneous change in position and a spike in torque. This issue can be further solved by interpolating the action during the simulation time step or using velocity control instead of position control.

The findings show that SEA can be used to improve the performance of a small quadrupedal robot's locomotion. The optimal design outperforms the benchmark in terms of both faster motion (increase 34%) and also less power usage (decrease mechanical power by 26.6% and actuator loss by 26.6%). Although the finding highlights the advantages of including SEA in a robot, it is based on a number of assumptions (i.e., perfect position tracking actuator for a simplified model). The results are also subjected to a small set of design parameters; therefore, further studies are needed to expand the parameter set and evaluate the outcomes. The hyperparameter adjustment is another factor that may have a significant impact on the learning outcome. Because we want to obtain a single control policy that can control both compliance and the stiff robot, we also need to construct a model that can also represent the stiff robot (i.e. the benchmark). With a single policy, we expect to observe the more obvious transition between gait and also reduce the bias of training two networks.

In further investigations, we intend to integrate SEA into other joints, including HAA and HFE joints in order to assess the performance and weigh the benefits and drawbacks of the more complex robot. Also, we would like to extend the experiment to include walking on uneven ground and command tracking, which will result in more dynamic and less periodic movements, such as turning and changing directions. Furthermore, we also would like to examine further the full model by estimating parameters and comparing outcomes to the simplified model.

Bibliography

[1] N. Kashiri, A. Abate, S. J. Abram, A. Albu-Schaffer, P. J. Clary, M. Daley, S. Faraji, R. Furnemont, M. Garabini, H. Geyer, A. M. Grabowski, J. Hurst, J. Malzahn, G. Mathijssen, D. Remy, W. Roozing, M. Shahbazi, S. N. Simha, J.-B. Song, N. Smit-Anseeuw, S. Stramigioli, B. Vanderborght, Y. Yesilevskiy, and N. Tsagarakis, "An overview on principles for energy efficient robot locomotion," *Frontiers in Robotics and AI*, vol. 5, 2018.

[2] J. Pratt and B. Krupp, "Series elastic actuators for legged robots," *Proceedings of SPIE - The International Society for Optical Engineering*, 09 2004.

[3] A. G. L. Junior, R. M. de Andrade, and A. B. Filho, "Series elastic actuator: Design, analysis and comparison," in *Recent Advances in Robotic Systems*, G. Wang, Ed. Rijeka: IntechOpen, 2016, ch. 10.

[4] G. A. Pratt and M. M. Williamson, "Series elastic actuators," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1995, pp. 3137–3181.

[5] M. Hutter, C. Gehring, M. Bloesch, M. Hoepflinger, C. Remy, and R. Siegwart, "Starleth: a compliant quadrupedal robot for fast, efficient, and versatile locomotion," 09 2012, pp. 483–490.

[6] M. Hutter, C. Gehring, M. Bloesch, M. Hoepflinger, P. Fankhauser, and R. Siegwart, "Excitation and stabilization of passive dynamics in locomotion using hierarchical operational space control," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 2977–2982.

[7] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," 2021.

[8] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, no. 47, oct 2020.

[9] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State, "Isaac gym: High performance gpu-based physics simulation for robot learning," 2021.

[10] E. A. Bolívar-Nieto, T. Summers, R. D. Gregg, and S. Rezazadeh, "A convex optimization framework for robust-feasible series elastic actuators," *Mechatronics*, vol. 79, p. 102635, 2021.

[11] E. A. Bolívar-Nieto, G. C. Thomas, E. Rouse, and R. D. Gregg, "Convex optimization for spring design in series elastic actuators: From theory to practice," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 9327–9332.

[12] T. Chen, Z. He, and M. Ciocarlie, "Hardware as policy: Mechanical and computational co-optimization using deep reinforcement learning," 2020.

[13] J. Won and J. Lee, "Learning body shape variation in physics-based characters," *ACM Trans. Graph.*, vol. 38, no. 6, nov 2019.

[14] A. Belmonte-Baeza, J. Lee, G. Valsecchi, and M. Hutter, "Meta reinforcement learning for optimal design of legged robots," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 12 134–12 141, oct 2022.

[15] F. Bjelonic, J. Lee, P. Arm, D. Sako, D. Tateo, J. Peters, and M. Hutter, "Learning-based design and control for quadrupedal robots with parallel-elastic actuators," *IEEE Robotics and Automation Letters*, pp. 1–8, 2023.

[16] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017.

Appendix A

Definition

A.1 Observation function definition

- **Base linear velocity:** This term represents the linear velocity of the robot.
- **Base angular velocity:** This term represents the angular velocity of the robot.
- **Projected gravity:** This term represents the value of the projected gravity in the simulation.
- **Joint position:** This term represents the position of each joint of the robot.
- **Joint velocity:** This term represents the velocity of each joint of the robot.
- **Actions:** This term represents the action from a control policy, which is the actuator position in the simplified model.
- **Spring constant:** This term represents the value of the spring constant which is a design parameter in this project.
- **Theta target:** This term represents theta difference between actuator position and joint position.

A.2 Reward function definition

- **Tracking angular velocity reward:** This term minimizes the error between the command angular velocity and the robot angular velocity.

$$r \coloneqq exp(-||\omega_{command} - \omega_{robot}||^2/\sigma)$$

- **Tracking linear velocity in y-axis reward:** This term minimizes the error between the command linear velocity and the robot linear velocity in the y-axis direction.

$$r \coloneqq exp(-||v_{y\,command} - v_{y\,robot}||^2/\sigma)$$

- **Maximum linear velocity in x-axis reward:** This reward encourages the robot to move as fast as possible in the x-axis direction (i.e. the front of the robot)

$$r \coloneqq v_{x-robot}$$

- **Action rate reward:** This term encourages the robot to smooth the action by minimizing the difference between the previous and the current command.

$$r := -0.02 \cdot ||action_{previous} - action_{current}||^2$$

- **Foot Slippage reward:** This term penalizes foot velocity while in contact with the ground.

$$r := -0.02 \cdot \sum_{i=1}^{4} ||v_{i\,foot} \cdot contact||; i \in \text{foot}$$

- **Joint position limit selected reward:** This reward penalizes the selected joint

$$r := -0.5 \cdot \sum_{i=1}^{8} (\theta_i - \theta_{i\,soft\,limit}); i \in \text{joint without SEA}$$

- **Joint torque reward:** This term decreases the torque of each joint, which can also imply the electrical energy consumption (i.e. actuator loss).

$$r := -\sum_{i=1}^{12} \tau_i^2; i \in \text{joint}$$

- **Actuator velocity reward:** This term penalizes the velocity of the actuator if the velocity of the motor is greater than the maximum velocity limit and does nothing otherwise.

$$r := \begin{cases} -0.05 \cdot ||\omega_{actuator} - \omega_{actuator(max)}||^2, & |\omega_{actuator}| \geq |\omega_{actuator(max)}|. \\ 0, & \text{otherwise.} \end{cases}$$

- **Mechanical Power reward:** This penalty minimizes the power consumption of the robot.

$$r := -\sum_{i=1}^{4} \tau_i \times \omega_{actuator} - \sum_{j=1}^{8} \tau_j \times \omega_{joint}; i \in \text{joint with SEA}, j \in \text{joint without SEA}$$

A.3 Hyperparameters

Hyperparameter	**Value**
Value loss coefficient	1.0
Clipping	0.2
Entropy coefficient	0.005
Epoch	5
Minibatch	10
Learning rate	0.001
Gamma	0.99
Lambda	0.95
KL Target	0.01

Table A.1: Hyperparameter used in PPO algorithm.

Appendix B

Code overview

B.1 Code overview

- **legged_gym/legged_gym/envs/locomotion/minimal/minimal_sea_config.py:**
  Defines learning environment and design parameters.
- **legged_gym/legged_gym/common/assets/robots/articulation.py:**
  Defines actuator and computes spring equation.
- **legged_gym/legged_gym/common/assets/robots/legged_robots/legged_robots_cfg.py:**
  Defines initialization of the learning environment.
- **legged_gym/legged_gym/envs/observations.py**
  Defines observation function.
- **legged_gym/legged_gym/envs/locomotion/observations.py**
  Defines observation function.
- **legged_gym/legged_gym/envs/rewards.py**
  Defines rewards function.
- **legged_gym/legged_gym/envs/locomotion/rewards.py**
  Defines rewards function.
- **legged_gym/resources/robots/minimal/urdf/minimal_sea_simplify.urdf**
  Defines robot parameters.
- **legged_gym/tests/test.py**
  Scripts to test the learning environment.
- **legged_gym/tests/test.py**
  Scripts to test the learning environment.
- **legged_gym/tests/test_step_response.py**
  Scripts to run step response experiment.
- **legged_gym/scripts/train.py**
  Scripts to train the control policy.
- **legged_gym/scripts/play.py**
  Scripts to run the trained control policy.
- **legged_gym/scripts/play_new_plot.py**
  Scripts to run the trained control policy with the modified plot.

- **legged_gym/scripts/play_analyzer.py**
  Scripts to run the trained control policy for the objective experiment.