Towards Smooth Policies in Deep Reinforcement Learning for Musculoskeletal Simulations of Human Walking

Bachelor's Project Thesis

Carl Lange, s4007533, c.a.lange@student.rug.nl,
Supervisor: Prof. Dr. Raffaella Carloni, r.carloni@rug.nl

Abstract: Recent advances in the musculoskeletal modeling of human walking demonstrated the ability of Proximal Policy Optimization combined with imitation learning to train a simulated musculoskeletal model for human-like walking. However, the resulting gait featured erratically changing muscle activations, which imposes a hurdle for applying this technique to actuated prostheses. Therefore, the main goal of this paper was to build on the previous work and apply regularization techniques in order to learn a smoother policy with less erratic activations. The results demonstrate that L_2 layer regularization, changed neuron activation function, and reward shaping effectively decreased muscle erraticness and resulting torques. However, not all simulation runs achieved a stable gait pattern. Additionally, the learnt gaits that were stable showed large discrepancies in joint angles when compared to human data. In summary, the results suggest that the applied regularization techniques were successful at decreasing erraticness but there exists a trade-off between regularization and performance.

Towards Smooth Policies in Deep Reinforcement Learning for Musculoskeletal Simulations of Human Walking

Bachelor's Project Thesis

Carl Lange, s4007533, c.a.lange@student.rug.nl,
Supervisor: Prof. Dr. Raffaella Carloni, r.carloni@rug.nl

1 Introduction

Physics-based computer simulations emerged as a useful tool to study human movement and locomotion. One of the application domains is the engineering of actuated prostheses for human amputees. Passive prostheses are less efficient from a standpoint of metabolic energy demands (Fey et al., 2011) and often lead to asymmetric gait patterns causing "wear and tear" in the passive structures of the amputees (Martinez-Villalpando, 2012). Active, actuated prostheses can alleviate such problems. However, active prostheses raise the problem of actuator control constrained by the individual properties and gait patterns of amputees. Traditional control approaches have the downsides of requiring extensive parameter tuning for each individual as well as being unable to adapt to new gait patterns and situations (Lawson and Goldfarb, 2014).

Recently, Reinforcement Learning (RL), and Deep Reinforcement Learning (DRL) in particular, demonstrated to be a promising framework for developing the controllers of actuated prostheses. All but one submitted solutions for the NeuRIPS 2018 competition *Artificial Intelligence for Prosthetics* (Kidziński et al., 2020) deployed DRL algorithms to train a simulated musculoskeletal model with a lower-limb prosthesis to walk. A more comprehensive review of existing work on this topic can be found in Table 1.1, but it can be said that many of the learned policies, though successfully achieving stable, human-like gaits suffer from the problem of erratic actions. In the case of musculoskeletal models, this means erratically activating the muscle fibers of the model such that the activation differs vastly from time step to time step. Such erratic behavior may be caused by the complexity of the underlying deep neural networks (DNNs) that approximate the policy. Goodfellow et al. (2014) showed that the output of DNNs can differ vastly for only slightly different inputs.

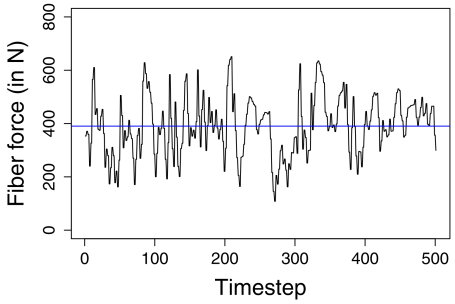


Figure 1.1: An example of a policy producing actions that lead to erratic changes in muscle fiber force. The graph is taken from De Vree and Carloni (2021).

Erraticness may be unproblematic in simulated prostheses but provides a limitation for applying such algorithms on physical hardware. Extreme and frequently oscillating activations increase wear and tear and thus decrease the life span and operability of actuators. Thus, to achieve the goal of controlling actuated prostheses with DRL algorithms, the learned policies need to be as smooth as possible without losing their ability to produce a stable gait.

This paper mainly builds upon work by De Vree and Carloni (2021) by using the same algorithm (PPO + imitation learning) and simulation environment OpenSim Delp et al. (2007). The two main objectives of this paper

are to

- replicate the success of De Vree and Carloni (2021) on a more realistic musculoskeletal model with four additional muscles
- investigate the effects of different regularization methods on the erraticness of muscle activations and joint torques

Achieving these objectives would pave the way for directly controlling physical prostheses with DRL controllers.

2 Theoretical background

This section provides a brief overview of the reinforcement learning framework and the advances in deep reinforcement learning.

Reinforcement learning The goal of reinforcement learning is to train an agent to act optimally in an environment. As opposed to the supervised learning paradigm, the RL agent learns by interaction with the environment. Specifically, the agent acts according to a policy π which maps any state of the environment $s \in \mathcal{S}$ to any of the possible actions $a \in \mathcal{A}$. The goal of the agent is to find the optimal policy π^* that maximizes the discounted cumulative reward. This task can be defined as a Markov Decision Process (MDP) $\langle \mathcal{A}, \mathcal{S}, \mathcal{R}, \gamma, \mathcal{T} \rangle$. The reward function $\mathcal{R}(s, a) \to \mathbb{R}$ returns a real number for a specific action a taken in a state s and is used to inform the agent of the "goodness" of its action in a state. The sum of future rewards is usually discounted over time by a discount factor γ. A discount factor favors rewards in the near future (i.e. only a few states away) over rewards received in the far future. Lastly, the transition function $\mathcal{T}(s'|s, a)$ is a deterministic or stochastic function that determines the next state s' of the environment given the current state s and action a. MDPs are said to be model-free if the transition function of the environment is not known to the agent. This is the case for many real-life applications such as controlling a musculoskeletal model (Sutton and Barto, 2018). The value function $V(s)$ defines the expected total reward received by the agent for being in state s.

Deep reinforcement learning Classic RL algorithms require explicit representations of the value function $V(s)$ or state-action value function $Q(s, a)$ in lookup tables. This makes such methods inapplicable for continuous state or action spaces, or even very large discrete spaces due to the lookup table limitations imposed by memory hardware. Deep Reinforcement Learning (DRL) solves this problem by using neural networks (NNs) to approximate components of the MDP such as the value function $V(s)$ and the policy $\pi(s)$.

Feedback loop The agent learns an optimal policy via a feedback loop with the environment. A schematic representation is shown in Figure 3.2. At time t the agent observes state s_t. A NN is used to parameterize the current policy π_t. With s_t as input to π_t, the agent chooses and executes action a_t. This changes the state to s_{t+1}, the agent receives the scalar reward r_t. The reward feeds into the loss function of the policy network and the agent updates its policy $\pi_t \to \pi_{t+1}$ by changing the weights of the NN according to the loss function.

3 Methods

In this section, the methodological details of the simulation, learning algorithm, reward functions, and experimental setup are explained.

3.1 Simulation

OpenSim, developed by Delp et al. (2007), is a physics simulation engine that is frequently used for biomechanical modeling of human movement and rehabilitation (Seth et al., 2018). In addition, OpenSim proved to be useful in related work on human locomotion simulation by De Vree and Carloni (2021); Weng et al. (2021); Nowakowski et al. (2021); Gao et al. (2020); Anand et al. (2019). The musculoskeletal model simulated in OpenSim has

Table 1.1: Overview of the relevant existing literature on regularization methods for Reinforcement Learning.

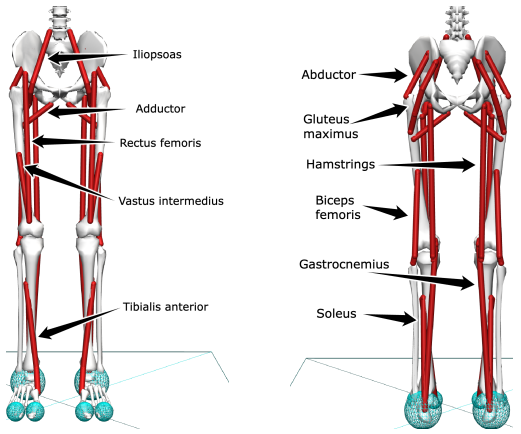
Reference	Category	Method	RL task	Outcome
Liu et al. (2019)	MLP architecture	Adding L_1 and L_2 regularization, Entropy regularization, Batch normalization, and Dropout to RL algorithms	9 continuous control tasks from MuJoCo and RoboSchool	Increased rewards, L_2 regularization being the most effective
Lee (2020)	MLP architecture	Testing the effects of different MLP activation functions on received reward in RL	Continuous control 2D car racing	ReLU-like activation functions received the highest rewards, with ReLU6 (Howard et al., 2017) being the best
Jang and Son (2019)	MLP architecture	Testing the effect of combinations of activation functions paired with kernel initializers on the performance of PPO	OpenAI Gym cartpole and pong	ReLU activation and random normal initializers lead to the highest reward in CartPole
Hossny et al. (2021)	MLP architecture	Making the activation function for each output neuron a learnable parameter of the MLP	OpenAI Gym pendulum, LunarLander, BipedalWalker	Parameterized activation functions lead to higher rewards and less fluctuating actions
van Wouden (2021)	Reward function	Adding a penalty term in the reward function that penalizes large changes of actions between time steps	Human walking with a simulated musculoskeletal model in OpenSim	No walking behavior was achieved by the agent
Asadulaev et al. (2020)	Policy loss function	regularizing the policy network by penalizing high Jacobian conditioning in the loss function	OpenAI Gym continuous control	improve policy stability
Shen et al. (2020)	Policy loss function	Adding a penalty term that encourages local Lipschitz continuity of the loss policy function	5 continuous control tasks from MuJoCo	Smoother policy and increased sample efficiency
Thodoroff et al. (2018)	Policy loss function	Smoothening historic estimated advantages with an exponential term	OpenAI Atari games	Higher reward received by regularized agents
Mysore et al. (2021)	Policy loss function	Adding a penalty term to the policy loss that penalizes large policy changes in the spatial and temporal domain	4 continuous control OpenAI tasks	Regularization improved policy smoothness
Raffin et al. (2020)	Exploration method	State-dependent exploration adapted to Deep Reinforcement Learning algorithms	4 continuous control PyBullet tasks	Regularized controller achieved similar performance with less erratic movements compared to a standard controller

Table 3.1: PPO hyperparameters. All values were adopted from De Vree and Carloni (2021).

Parameter	Value
Seed	999
Learning rate	0.0003 (annealed)
Steps per batch	1536
Batch size	512
Updates per batch	4
Discount factor γ	0.999
GAE λ	0.9
Entropy coefficient	0.01
Clip range	0.2

22 Hill-type muscles (Hill, 1938), 14 degrees of freedom (DoF) and was developed by Seth et al. (2018). A musculoskeletal model with only 18 muscles and 10 DoF was frequently used in previous research (Weng et al., 2021; De Vree and Carloni, 2021). In this study, a model extended by four muscles and 4 DoF was used because it is more realistic and the additional muscles should allow more granular control of the model.

In RL, the observation that the agent makes in a state is a crucial component. For this implementation, the observation was a vector of size 142. A more elaborate representation of the observation vector can be found in Table 3.2.



(a) Front view **(b) Back view**

Figure 3.1: Overview of the musculoskeletal model. It has 22 muscles, which are labeled in the figure. The model is symmetric, meaning it has eleven identical muscles on each side of the body.

3.2 Learning algorithm

Proximal Policy Optimization (PPO) was used as the learning algorithm. PPO belongs to the class of policy gradient methods, which update

Table 3.2: Overview of the observation vector. Values in parentheses indicate three values for (x, y, z) coordinates.

Variable	Amount	Values
Body parts	72	Position (3), velocity (3) for: femur, tibia, talus, calcaneus, toes, torso, head
Model: Hips	12	Angle, angular velocity: hip flexion, hip adduction, hip rotation
Model: Knees	4	Knee angle, angular velocity
Model: Ankles	4	Ankle angle, angular velocity
Model: Pelvis	18	Global position (3), global orientation (3), local linear velocity (3), local angular velocity (3), local position (3), local velocity, local orientation (3), local angular velocity (3)
Imitation data: Hips	12	Angle, angular velocity: hip flexion, hip adduction, hip rotation
Imitation data: Knees	4	Knee angle, angular velocity
Imitation data: Ankles	4	Ankle angle, angular velocity
Imitation data: Pelvis	12	local angular velocity (3), local position (3), local velocity, local orientation (3)
Sum	142	

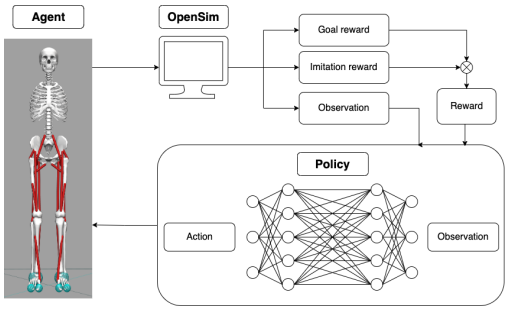


Figure 3.2: The RL training loop. The policy, parameterized by an MLP, maps an input observation vector to an action vector which is executed by the agent. The agent then receives a reward based on the changed state of the simulation and updates its policy.

the policy by performing grading ascent on the estimated gradient of the policy. PPO is the successor of Trust Region Policy Optimization (TRPO). TRPO, proposed by Schulman et al. (2015a), maximizes the expected ratio of the new policy π_θ and old policy $\pi_{\theta_{\text{old}}}$ while constraining the size of the update. The update step is calculated as the maximum Kullback-Leibler (KL) divergence of the policy's parameters θ. Specifically, TRPO maximizes $L^{CPI}(\theta)$:

$$\begin{aligned} \underset{\theta}{\text{maximize}} \quad & \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \hat{A}_t \right] \\ \text{subject to} \quad & \hat{\mathbb{E}}_t \left[\text{KL} \left[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \right. \right. \\ & \left. \left. \pi_\theta(\cdot \mid s_t) \right] \leq \delta \right] \end{aligned} \tag{3.1}$$

where $\hat{\mathbb{E}}_t$ is the empirical average at time t, π_θ is the new parameterized policy, $\pi_{\theta_{\text{old}}}$ the old parameterized policy, a_t the action taken at time t, s_t the state at time t, and $\hat{A}_t$ the estimated advantage at time t. The advantage is estimated using Generalized Advantage Estimation (GAE, Schulman et al. (2015b)). Expressing the changing policy with this ratio formed the main novelty of TRPO. Ratios larger than one indicate better action under π_θ compared to $\pi_{\theta_{\text{old}}}$, values smaller than one stand for worse actions in comparison. Although TRPO showed promising performance in the original paper, it turned out to be difficult to implement and complex to compute (Schulman et al., 2015a). Thus, Schulman et al. (2017) introduced PPO.

PPO's main improvement over TRPO is to clip the policy ratio. This prevents policy updates from being too large, i.e. π_θ being too far away from $\pi_{\theta_{\text{old}}}$ which could lead to unstable training behavior Schulman et al. (2017). PPO minimizes the objective:

$$L(\theta) = \hat{\mathbb{E}}_t \Big[\min \Big(r_t(\theta)\hat{A}_t, \\ \text{clip}\left(r_t(\theta),\ 1-\epsilon, 1+\epsilon\right)\hat{A}_t \tag{3.2}$$

where $r_t(\theta)$ is the policy ratio and ϵ is the clipping hyperparameter.

In this study, the objective of the agent is to control the skeleton's movement by activating its 22 muscles. An MLP represents the policy by outputting 22 floats in the range $[0, 1]$, given the observation vector as input as described in Table 3.2. Thus, the policy is represented by the weights θ of the MLP. The effectiveness of the current policy is evaluated using GAE (Schulman et al., 2015b) and a second MLP which is trained to learn the state-value function $V(s)$.

The implementation of PPO in Python is largely adopted from the Stable-Baselines 3 framework (Raffin et al., 2021). Table 3.1 provides an overview of all hyperparameters used in the PPO implementation. They were entirely adapted from the work of De Vree and Carloni (2021).

3.3 Imitation learning

Human data To increase the similarity between the gait learned by the controller and a human gait, imitation learning has been demonstrated to be successful by De Vree and Carloni (2021), who used walking data from 83 children. In this study, the gait kinematics from an able-bodied adult subject were used because the subject's mass and height closely resemble the measurements of the musculoskeletal model. The data was obtained from subject AB06 from a biomechanical study by Camargo et al. (2021). Subject AB06 was a 1.8 meters tall, 20-year-old male that was 74.8kg heavy. The data used for imitation learning features joint kinematics of the hips, knees, ankles, and pelvis and was recorded during ground-level walking at $1.12 \pm 0.21\mathbf{m/s}$ (average speed of all subjects for self-paced normal walking). For more methodological details of data collections, see Camargo et al. (2021).

Data processing The publicly available data set was processed for imitation learning. First, the musculoskeletal model was rotated by 90° clockwise around its y-axis to align the model-centered coordinate system with the collected data. Next, collected data was trimmed down to the interval $[12.8s,\ 16.8s]$ which corresponds to the time span in the trial when subject AB06 walks in a straight line. This trimmed data was used to calculate inverse kinematics (IK) with OpenSim's Inverse Kinematics tool. The root mean square error for the IK data was 0.261, which is sufficiently precise according to the OpenSim documentation (Delp et al., 2007). Finally, IK data was extracted and post-processed. In postprocessing, the first 39 time steps of the imitation data were removed to initialize the agent in the mid-swing phase of the gait cycle rather than starting standing. Such initialization techniques proved to be very useful during training. This way, the agent was forced to take the first step because the unstable starting position caused a forward movement that would result in a fall if the agent did not move. The resulting file contained 680 time steps with $\delta_t = 0.005$. For each time step, the following data points were included for the left and right limbs: hip flexion angle, hip adduction angle, hip rotation angle, knee angle, and ankle angle. Furthermore, pelvis coordinates (x, y, z) and pelvis orientation (tilt (around the z-axis), list (around the x-axis), rotation (around the y-axis)) were incorporated.

3.4 Reward formulation

The reward function plays an essential role in RL, as it encodes the objective for which the agent adjusts its policy. For the task of learning to walk, the reward function was composed of two parts: a goal term that rewards the agent for moving its pelvis forward in the sagittal plane and an imitation term that penalizes dis-

Table 3.3: This table provides an overview of all conducted training runs. Each run is constituted by a specific reward function and policy. The runs are categorized into four categories based on the type of regularization that is used in the run. In addition, the table shows how many parallel environments were used in each run and for how many total time steps each run was trained. The entries in the "Name" column serve as identifiers for the runs and are capitalized. Envs denotes the number of parallel environments that were used in the training process. This number was set to the number of CPUs of the hardware equipment.

Name	Category	Envs	Total steps	Policy	Reward
ReLU	Activation	4	3,913,728	ReLU	Base
LReLU	Activation	2	3,136,512	LReLU	Base
Tanh	Activation	1	3,087,360	Squash	Base
L_2 0.00005	Layer	16	4,128,768	$L_2 - 5e^{-5}$	Base
L_2 0.0005	Layer	4	3,102,720	$L_2 - 5e^{-4}$	Base
L_2 0.001	Layer	4	3,683,328	$L_2 - 1e^{-3}$	Base
Large (0.001)	Layer	4	3,428,352	Large	Base
Multiply	Reward	8	4,079,616	Squash	Multiply
Additive	Reward	8	4,300,800	Squash	Additive
Curriculum	Reward	8	6,494,208	Squash	Curriculum
Feet penalty	Tune	16	10,272,768	Squash	Feet penalty
Adduction penalty	Tune	8	10,844,160	Squash	Adduction
Base	Tune	8	7,262,208	Standard	Base

similarity between the agent's kinematics and the imitation data. In this study, the effects of six different reward functions were explored for their ability to achieve a stable gait and reduce erraticness in muscle activations. In the following equations, p refers to penalties, r to rewards, v to velocity, p to position, and "imi" to the corresponding data points in the imitation data. Table 3.3 shows which reward function was used in each simulation run.

Base reward The base reward function follows previous work from De Vree and Carloni (2021). The goal reward r_{goal_t} is calculated at every time step t using a velocity penalty p_{vel_t}:

$$p_{\text{vel}_t} = (p_{v_x} - \text{imi}\, p_{v_x})^2 + (p_{v_z} - \text{imi}\, p_{v_z})^2 \quad (3.3)$$

$$r_{\text{goal}_t} = exp(-8 * p_{\text{vel}_t}) \quad (3.4)$$

where v_x and v_z are the agent's pelvis velocity in the x and z directions.

The imitation reward r_{imi_t} is calculated using positional and velocity penalties for joint angles:

$$p_{\text{pos}_t} = \sum (p_{\text{agent}} - p_{\text{imitation}})^2 \quad (3.5)$$

$$p_{\text{vel}_t} = \sum (v_{\text{agent}} - v_{\text{imitation}})^2 \quad (3.6)$$

where the sum ranges over the knee angle, ankle angle, hip adduction, and hip flexion, each for both legs as well as pelvis rotation, list, and tilt. The imitation reward is a weighted sum of the exponentiated penalty terms:

$$r_{\text{imi}_t} = 0.75 * exp(-2 * p_{\text{pos}_t} + 0.25 * exp(-2 * p_{\text{vel}_t}) \quad (3.7)$$

Combining both reward terms leads to the total reward r_t:

$$r_t = 0.4 * r_{\text{goal}_t} + 0.6 * r_{\text{imi}_t} \quad (3.8)$$

All weights were adopted from De Vree and Carloni (2021) who found that these values lead

to the best results. This reward composition rewards the agent when it walks (i.e. the pelvis moves) in x direction and penalizes any deviations of the knee, ankle, pelvis, and hip joint angles from the imitation data.

Erraticness reward Previous work by van Wouden (2021) attempted to decrease muscle activation erraticness by including a penalty term in the reward function, shown in Equation 3.9. The additional penalty was an exponentiated sum for all muscles of the current muscle activation and activation at the previous time step.

$$r_{\text{err}} = exp(-c * \sum_{m=0}^{m=22} (\text{activation}_t - \text{activation}_{t-1})^2) \tag{3.9}$$

In van Wouden (2021), no value was stated for the constant factor c multiplied by the sum. In this study, r_{err} is calculated slightly different:

$$r_{\text{err}} = 1 - \frac{1}{22} * \sum_{m=0}^{m=22} (\text{activation}_t - \text{activation}_{t-1})^2) \tag{3.10}$$

As all muscle activations are in the range $[0, 1]$, the mean activation difference can maximally be 1. Thus, if all activations from the 22 muscles are maximally different from their previous values, the agent receives an erraticness reward of $1 - \frac{1}{22} * 22 = 0$. This erraticness penalty can be combined with the base reward function in two possible ways. The penalty term can be added as another summand to the weighted sum of total rewards, resulting in the reward function

$$r = w_{\text{goal}} * r_{\text{goal}} + w_{\text{imi}} * r_{\text{imi}} + w_{\text{err}} * r_{\text{err}} \tag{3.11}$$

This reward function was used in the Additive run (see Table 3.3 for an overview of all conducted runs). It comes with a potential downside. Not only do the weights of the components have to be tuned, the agent could learn to maximize this reward by just maximizing the r_{goal} and r_{imi} terms while discarding the r_{err} term. This potential problem can be solved by multiplying the r_{err} reward with the sum of r_{goal} and r_{imi}:

$$r = (w_{\text{goal}} * r_{\text{goal}} + w_{\text{imi}} * r_{\text{imi}}) * r_{\text{err}} \tag{3.12}$$

This prevents the reward from increasing if the r_{err} is low. The Multiply run used this reward function.

Curriculum reward The results from van Wouden (2021) show that the used reward formulation did not produce a walking gait. Thus, it might be the case that the regularization came at the cost of the expressiveness of the policy. One potential solution here is to first train the controller to walk, and once a full gait cycle is achieved, add regularization to the reward function. To train the controller to walk, the base reward function (Equation 3.8) was used. Then, a reward function similar to the additive erraticness reward in Equation 3.11 was applied. Furthermore, the weights of the components changed when the training episode becomes longer to encourage learning focus on decreasing erraticness.

$$r = w_{\text{goal}} * r_{\text{goal}} + w_{\text{imi}} * r_{\text{imi}} + w_{\text{err}} * r_{\text{err}} \tag{3.13}$$

The weight vector w containing w_{goal}, w_{imi}, and, w_{err} is

$$w = \begin{cases} [0.45, 0.45, 0.1]^T, & \text{if } t < 150 \\ [0.3, 0.45, 0.25]^T, & \text{otherwise} \end{cases} \tag{3.14}$$

where t is the time step within the current episode.

Gait-tune rewards Intermediate results included a walking agent. However, that agent had a physically impossible walking gait, as its bones penetrated each other during some parts of the gait cycle. Therefore, two other reward functions were designed to discourage such undesirable behavior. In both rewards, the r_{goal} reward was adjusted. In the Adduction penalty reward, a penalty for activations of the hip adduction muscles was included with the goal of

preventing excessive hip adduction that could lead to legs moving to undesirable positions. The penalty is formulated as the sum of activations of the left and right adductor muscles:

$$p_{\text{adduction}} = \sum \text{adductor activation} \quad (3.15)$$

The penalty is incorporated into the goal reward as

$$r_{\text{goal}_t} = exp(-8 * p_{\text{vel}}) - 0.5 * p_{\text{adduction}} \quad (3.16)$$

This reward was used in the Adduction penalty run.

Because the learnt gait showed very straight legs with little knee flexion, the weight of knee flexion deviation from the imitation data was doubled in the imitation reward. To discourage the agent from crossing its feet, the sum of goal reward and imitation reward was multiplied with a factor p_{foot} based on the amount of foot crossing:

$$p_{\text{foot}} = \max(0.5 - \text{y\_left} - \text{y\_right},\ 0.1)$$
$$\text{p}_{\text{foot}} = \begin{cases} p_{\text{foot}}, & \text{if y\_left} \geq \text{y\_right} \\ 1, & \text{otherwise} \end{cases} \quad (3.17)$$

where y_left and y_left are the positions of the left and right foot in the frontal plane. This leads to the total reward being

$$r = (w_{\text{goal}} * r_{\text{goal}} + w_{\text{imi}} * r_{\text{imi}}) * p_{\text{foot}} \quad (3.18)$$

This reward was used in the Feet penalty run.

3.5 Erraticness methods

Table 1.1 categorizes attempts to decrease erraticness or learn smoother policies in the literature. In this study, the effects of reward shaping and MLP regularization are investigated. For the former, the reward function of the agent is modified, as described extensively in Section 3.4. For the second category, the MLP is regularized using layer regularization and activation functions.

Layer regularization L_2 regularization adds an additional penalty term to the loss function of the MLP. This term, shown in Equation 3.19, penalizes large weights of individual neurons, thereby decreasing the chance of overfitting (Ng, 2004).

$$L = L_s - \lambda \sum_{i=0}^{N} \theta_i^2 \quad (3.19)$$

where L is the resulting loss function of the MLP, L_s the standard loss function, and θ the weight vector of size N of the MLP. Liu et al. (2019) find that L_2 regularization generally performs the best when compared to other regularization methods for MLPs. Liu et al. tuned λ, the coefficient for the additional L_2 loss term, to their RL problems using a range of $[0.00005,\ 0.001]$. In this study λ was evaluated for three values: 0.00005, 0.0005, and 0.001.

Activation functions In MLPs, the activation function of the neurons mathematically determines the neuron's output based on all its input from other neurons and a bias. Though activation functions are classically not considered to be regularization methods, they can have significant a impact on performance. Despite Dubey et al. (2021) reporting that some activation functions like the rectified linear unit (ReLU) perform well on a wide range of tasks, there is no one-fits-all activation function. Rather, the choice of activation function seems to be application-dependent. In the domain of RL, little research has been conducted on the effects of activation functions. In many RL frameworks (Raffin et al., 2021; Dhariwal et al., 2017) Tanh is the default activation function. In this study, the effects of using ReLU (Eq. 3.21) and leaky ReLU (LReLU, Eq. 3.22) activation functions are explored and compared to the Tanh function.

$$\text{Tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (3.20)$$

$$\text{ReLU}(x) = \max(0,\ x) = \begin{cases} x, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases} \quad (3.21)$$

$$\text{LReLU}(x) = \max(0,\ x) = \begin{cases} x, & \text{if } x \geq 0 \\ 0.01x, & \text{otherwise} \end{cases} \tag{3.22}$$

(Dubey et al., 2021).

3.6 Experimental setup

This subsection outlines how the regularization methods are tested and what data is collected. Three Linux computers were used for training, each with one GPU and 16, 8, and 4 CPUs respectively.

Training parameters Every training run (see Table 3.3) consists of a combination of reward function (see Section 3.4) and policy (see Table 3.4). The time for training the individual runs was heavily dependent on the number of total time steps, parallel environments, and computing hardware. These factors can also be seen in Table 3.3. Parallel environments were used during training for two reasons. Firstly, the trajectory collection part of PPO can be parallelized. When PPO is trained on a single environment, the agent acts in the environment for some amount of steps x, stores the observations, actions, and rewards, and then updates the policy based on the collected experience. With parallel environments, multiple agents go through the experience collection process at the same time, thereby collecting $x * y$ steps for updating the policy if y parallel environments are used. Thus more total time steps can be collected in the same wall time than with a single environment. In addition, every policy update is based on more total time steps, leading to potentially more stable updates. This benefit is also supported by the second feature of parallel environments. Each environment has a different random seed. Therefore, the collected trajectories differ between environments and should in theory generalize better.

Evaluation of erraticness To reliably evaluate the erraticness of a learned policy, there needs to be a method of quantification. This study used two quantitative measures for erraticness. The first one is adapted from Raffin et al. (2020) and termed continuity cost.

Table 3.4: Overview of the policies used in the experiment runs. All log probabilities were initialized to 0 and weights were initialized orthogonally. The number of neurons corresponds to the two hidden layers of each MLP. For the shared-layer policy, the first hidden layer is shared between the policy and value network and contains 512 neurons. The MLP then has separate heads. The policy head features a hidden layer with 128 neurons and the value head's hidden layer contains 312 neurons. When policies use a Tanh function to squash the output, the value of all neurons in the final layer is squashed through that function.

Name	Neurons	Activation	Squashing	$L_2\ \lambda$
Standard	312, 312	Tanh	None	None
Squash	312, 312	Tanh	Tanh	None
ReLU	312, 312	ReLU	Tanh	None
LReLU	312, 312	LReLU	Tanh	None
Large	512, 512	ReLU	Tanh	0.001
$L_2 - 5e^{-5}$	312, 312	Tanh	Tanh	0.00005
$L_2 - 5e^{-4}$	312, 312	Tanh	Tanh	0.0005
$L_2 - 1e^{-3}$	312, 312	Tanh	Tanh	0.001

$$\mathcal{C} = 100 \times \mathbb{E}_t \left[\left(\frac{a_{t-1} - a_t}{\Delta^a_{\max}} \right)^2 \right] \tag{3.23}$$

where a_{t-1} is the action taken at time step $t-1$, a_t is the action taken at time step t, and $\Delta^a_{\max}$ is the maximum action. Applied to the problem at hand, a_{t-1} is a vector of the 22 activations in the range $[0,\ 1]$ from the last time step, a_t is the vector of activations at the current time step, and $\Delta^a_{\max}$ is the maximum value for an action, so $\Delta^a_{\max} = 1$. The continuity cost for a trained agent is aggregated over time and for all muscles. A less erratic policy should lead to lower continuity costs.

The other measure investigates the torques that apply to the agent's knees and ankles in the sagittal plane when using a learned gait. The torques can be compared to those of a healthy human from experimental data. In this paradigm, a less erratic policy is represented

by a lower root mean square error of the mean torque M_x compared to M_x from the imitation data.

Evaluation of the gait The main metrics to evaluate which run achieved the best walking pattern are episode length (the agent should walk as far as possible) and Joints RMSE (the joint angles should be as similar to imitation data as possible). Furthermore, higher mean episode rewards indicate a good walking performance of the agent with respect to its specific reward function.

Simulation initialization The first 39 rows of the imitation data were removed during data processing. This leads to the agent starting in a right midstance phase of the gait, where the left heel is already lifted from the ground. This initialization of the agent, and the inertia that comes with it, has been empirically shown to improve the agent's learning ability.

Data collection During the training process of a run, reward and episode length are collected as the main training metrics. For the evaluation of a trained agent, the joint angles of hips, knees, ankles, and pelvis are stored. This data is used to compare the learned gait to the imitation data as well as to compute the torques that result at the knees and ankles. To evaluate the erraticness of the policy, the muscle activations and continuity cost for all muscles at each time step are calculated and stored during evaluation.

4 Results

This section presents the results of all training runs: the controller for normal walking, the controllers with reward regularization, the controller with layer regularization, and the controller with activation function regularization.

The runs are evaluated based on reward, episode length, continuity cost, resulting torques, joint angles, and achieved gait. The resulting torques, as well as joint angles, are compared to the imitation data using root mean square error (RMSE). The closer the RMSE is to 0, the more similar the agent's results are to the imitation data. Thus, a run is the best within a metric if it has the lowest RMSE out of all runs. The torques are normalized to the weight of the simulated model, 75.16kg. For the torques M_x in the agent's sagittal plane, the RMSE is calculated as

$$RMSE = \sqrt{\frac{(\text{agent} - \text{imi})^2}{t}}$$

with $t = \min(t_{\text{agent}},\ t_{\text{imi}})$. For torques, RMSE is aggregated over t for the knee and ankle joints of the right leg and the agent's x-dimension (sagittal plane). For joint angles, RMSE is aggregated over t and the pelvis, hip, knee, and ankle of the right leg. The continuity cost $\mathcal{C}$ is calculated for each muscle and each time step using Eq. 3.23 and aggregated over time and all muscles. Whether a full gait cycle was achieved or not is determined by analyzing the left foot of the agent. The agent starts the gait by rising the left foot. Thus, if two occasions of the left foot touching the floor are observed in the simulation, the agent completed a gait cycle.

Reward Figure 4.5 shows the mean episode rewards of the runs achieved during training, grouped by run category. The plots for the Reward (Figure 4.5b) and Layer (Figure 4.5d) categories both show that generally, the received rewards of stronger regularized runs are lower and increases stagnate earlier. In the Activation category (Figure 4.5a), the runs with Tanh and ReLU activation functions clearly achieved higher rewards which continue to increase, in contrast to the LReLU run. The Base run in the Tune category (Figure 4.5c) accomplished the highest reward out of the three runs. However, all three runs had different reward functions so a numerical comparison of rewards is less indicative of performance than the trend of the reward curve. Note that the Curriculum run starts at a reward of 100 because the training process started from the trained Tanh run, as explained in Section 3.4.

Table 4.1: Summary of all evaluation metrics. For each run category, the best value within a category is shown in *italic font*. The best value out of all runs is displayed in bold font. M_x, and Joint RMSE scores are reported in units of RMSE. Joint RMSE, M_x, and $\mathcal{C}$ are rounded to three, two, and one decimal respectively. Reward and episode length are rounded to integers.

Name (see Table 3.3)	Category	Reward	Episode length	$\mathcal{C}$	M_x	Joint RMSE	Gait cycle
ReLU	Activation	105	*242*	2.40	5.93	*0.419*	No
LReLU	Activation	62	149	2.03	*5.4*	0.43	No
Tanh	Activation	*107*	211	***1.52***	5.83	0.495	Yes
L_2 0.00005	Layer	*79*	*167*	2.54	6.8	0.306	No
L_2 0.0005	Layer	67	149	2.04	7.6	0.321	No
L_2 0.001	Layer	54	143	*2.03*	*6.0*	***0.279***	No
Large	Layer	54	137	2.34	6.6	0.503	No
Multiply	Reward	78	187	*1.58*	*5.4*	0.417	No
Additive	Reward	89	220	2.21	5.7	*0.401*	Yes
Curriculum	Reward	***199***	***486***	2.30	6.1	0.705	Yes
Feet penalty	Tune	*145*	255	*2.34*	***3.3***	*0.425*	Yes
Adduction penalty	Tune	120	186	2.42	6.7	0.451	Yes
Base	Tune	*145*	*256*	2.69	7.1	0.71	Yes

Table 4.2: Evaluation metrics for all runs, aggregated by run category.

Category	Reward	Length	$\mathcal{C}$	M_x	Joints	Gait
Activation	91.3	201.	1.98	5.72	0.448	0.333
Layer	63.5	149	2.24	6.74	0.352	0
Reward	122	298.	2.03	5.74	0.508	0.667
Tune	137.	266.	2.48	5.71	0.529	1

4.1 Normal walking

Table 4.2 shows evaluation metrics of all conducted runs, aggregated by category. In terms of episode length, runs in the reward category performed the best while Layer had the on average shortest episodes. In addition, none of the Layer runs achieved a stable gait. However, the Layer runs have the lowest Joints RMSE, indicating that the learned movement, though unstable, was most similar to human walking.

The pattern of Reward runs having a long episode length but high Joint RMSE persists when looking at the results of individual runs. From all runs, Curriculum achieved by far the longest episodes. Although this run learned a stable gait, the plotted gait pattern in Figure 4.3 (red line) and the high Joints RMSE provide evidence for the gait pattern being dissimilar to human walking. This finding is further supported by a visual analysis of the emerging gait. Figure 4.4 depicts an example: from 4.4a to 4.4b, the agent adducts its left leg excessively which causes the leg to move through the right leg. Correlation analysis for episode length and Joint RMSE reveals that there exists a moderate positive correlation between the two variables (*Pearson's* $r(11) = 0.697$). As both metrics are essential indicators for the goal of human-like walking, a trade-off must be made. The Feet penalty run serves as an example. It learned a stable gait, reaching an episode length of 255 while maintaining a Joints RMSE of 4.25 which is lower than the mean Joints RMSE of 4.51. It should also be noted that the goal of having a small Joint RMSE is important but becomes pointless if no gait cycle is achieved.

For further analysis, joint angles of the right ankle, knee, and hip of the trained agents are plotted in Figure 4.3 as a function of the gait cycle stage. Specifically, data is shown for all runs that performed best in at least one metric from Table 4.1. In addition, the imitation data ± one standard deviation (SD) is shown in grey for comparison. At the ankle joint, all runs but Tanh show somewhat similar kinematics (within one SD), although they lack some gran-

ularity, such as the small decrease in the angle at around 30% of the gait cycle. None of the runs shows the patterns of human hip flexion in their kinematic data. In fact, the hip flexion angle remains somewhat constant for all runs except Tanh, which shows some hip extension in the last third of its gait cycle. For knee kinematics, all runs except L_2 (0.001) display some similarity to the imitation data: Their knee angles decrease in the second quarter of the gait cycle before it increases again. However, all of the runs have consistently larger knee angles than the imitation data.

An interesting observation is that whether a gait cycle was achieved or not is positively correlated with Joints RMSE (*Pearson's* $r(11) = 0.585$). This can be interpreted as less-human-like gaits correlating with more chance to achieve a gait cycle.

4.2 Erraticness

The erraticness of each run's policy is evaluated from two points of view. The "action" point of view examines the continuity cost $\mathcal{C}$ of actions. The other view is more outcome-oriented, focusing on the torques that result from actions.

Actions The average continuity cost for all muscles over all time steps is shown in Table 4.1, and aggregated by run category in Table 4.2. This table shows that Activation runs had the lowest average continuity cost, whereas Tune runs had the highest average continuity cost. Out of all runs, Tanh achieved the lowest continuity cost, followed by the Multiply run. Importantly, the mean continuity cost scores of all regularization categories were lower than the mean score of Tune, the only category without regularization.

From the Layer runs, L_2 regularization with $\lambda = 0.001$ lead to the lowest continuity cost. Ordering the "layer" runs from most regularization ($\lambda = 0.001$) to least ($\lambda = 0.00005$), one can observe an inverse correlation between the level of regularization and continuity cost. Another pattern becomes apparent within the Layer runs: Although runs with more regularization seem to have lower continuity cost,

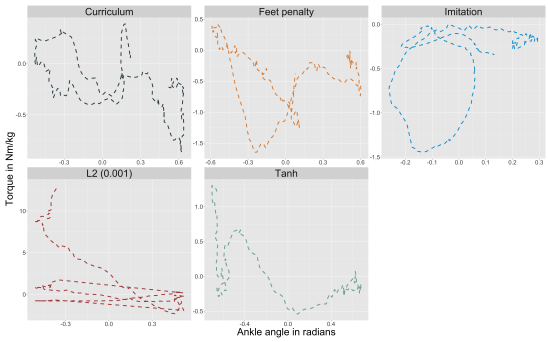


Figure 4.1: Torque normalized to model mass as a function of right ankle angle.

they also perform worse in terms of reward and episode length. In the Layer category, the highest reward and episode length were achieved by the run with the smallest λ. The same relation is found in the Reward category. The highest reward and longest episodes were both reached by the Curriculum run. However, the Reward run with the lowest continuity cost was Multiply, followed by Additive. In the Tune category, the Feet penalty reward resulted in a slightly lower continuity cost than the other two runs. All run categories individually as well as together show two patterns. Firstly, stronger regularization occurs together with lower continuity cost. Secondly, high continuity cost seems to be correlated with performance and gait stability. In fact, episode length and reward are both positively correlated with continuity cost: *Pearson's* $r_{\text{rew}}(11) = 0.329$; *Pearson's* $r_{\text{len}}(11) = 0.265$. Solely the Tanh run forms an outlier, as it achieved the lowest continuity cost out of all runs while having the highest reward within its run category.

Outcomes Next to assessing whether the actions produced by the policy are less erratic, the torques that result at the agent's joints from the executed actions can be evaluated. The M_x in Table 4.1 and 4.2 show the RMSE of the resulting torques at the knee and ankle in the sagittal plane of the agent, with respect to the imitation data.

The Tune runs show the on average lowest RMSE for torques, closely followed by the

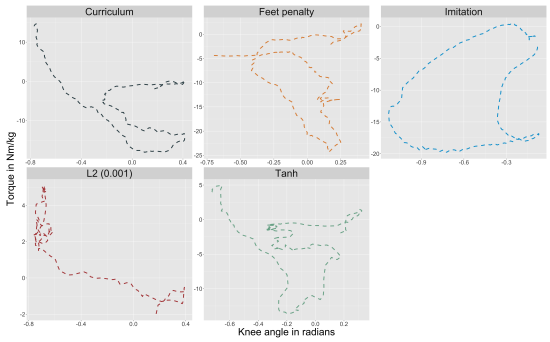


Figure 4.2: Torque normalized to model mass as a function of right knee angle.

Activation and Reward runs. The individual run with the lowest RMSE torque is the Feet penalty run from the Tune category.

From the Activation runs, LReLU recorded the lowest torque RMSE. The Layer category paints a similar picture for M_x as it did for continuity cost: more regularization correlates with lower torque RMSE. The run that resulted in the lowest RMSE had $\lambda = 0.001$. In the Reward category, this trend remains. The run with the highest reward and episode length, Curriculum, also registers the largest RMSE for torques within its category. Moreover, the run with arguably the most reward regularization, Multiply has the lowest RMSE. From the unregularized runs in the Tune category, the Feet penalty run achieved the lowest RMSE, while being identical in reward to the Base run and very close in terms of episode length.

Interestingly, the lowest torque RMSE of 3.32 was scored by the unregularized Feet penalty run. The value can be interpreted as the torques that apply to the knee and ankle of the Curriculum run being 3.32 $\frac{Nm}{kg}$ higher than in the imitation data. The agent's deviations from the imitation data are further supported by plots of torque as functions of knee and ankle angles in Figures 4.1 and 4.2.

It can be seen that none of the runs produce torque patterns similar to the imitation data. For the torques applying at the ankle, see Figure 4.1, one can notice that all runs produce torques at ankle angles that are much larger than in the imitation data. Another interesting observation is that all resulting torques in the imitation data are negative, meaning they accelerate the talus or tibia in the negative x direction of the agent. In contrast, all runs register some torques with positive values. This is likely another contributing factor to the unstable gait cycles of many runs. Similar observations can be made from the torque plot of the knee joint in Figure 4.2. In the imitation data, all torques happen at negative knee angles, i.e. when the knee is flexed. This is not the case for the simulation runs, which all show some torques applying at positive knee angles, i.e. extended knees. Torques at positive knee angles imply (over)extended, very straight legs. The gait patterns of the knee in Figure 4.3 align with this observation, as the simulation runs show consistently larger (positive) joint angles. The torque plots for the knee of the Feet penalty run seem most similar to the torques in the imitation data in terms of torque range and knee angle range. In addition, it is the only run for which the torques form a closed ellipsoid like the imitation data. The closed ellipsoid can be interpreted as the agent performing a full cycle of knee flexion and extension during the gait cycle. However, even for the Feet penalty run the difference in the shape of the ellipsoid and the imitation data torques indicate a discrepancy between the learned gait and imitation data.

5 Discussion

Two objectives were addressed in this study: 1) Replicating previous successful work by De Vree and Carloni (2021) on the 22 muscle model with 4 additional muscles and 2) assessing the effects of regularization techniques on policy erraticness. The results show that the former objective turned out to be more difficult than expected. The Base run, the first attempt to replicate De Vree and Carloni (2021)'s result, did not learn to walk longer than one gait cycle, and the gait was less similar to imitation data than results from De Vree and Carloni (2021)'s. The Curriculum run, which achieved the longest episodes, mastered many stable gait cycles but kinematic analyses re-

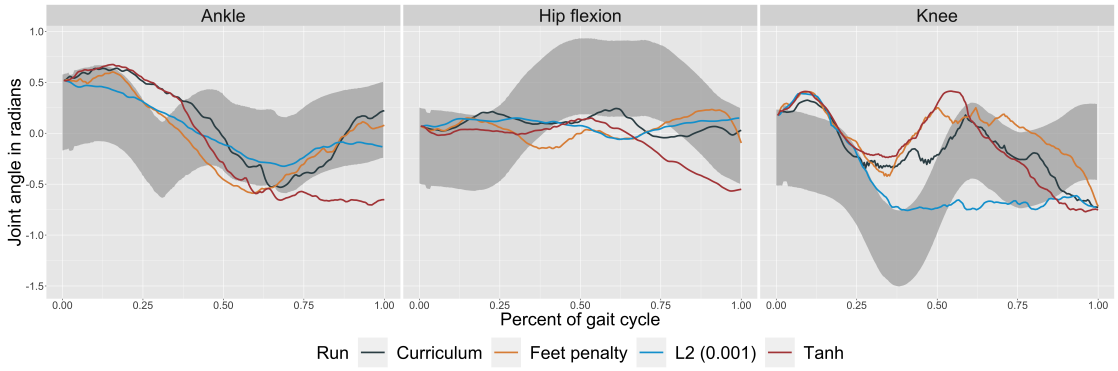
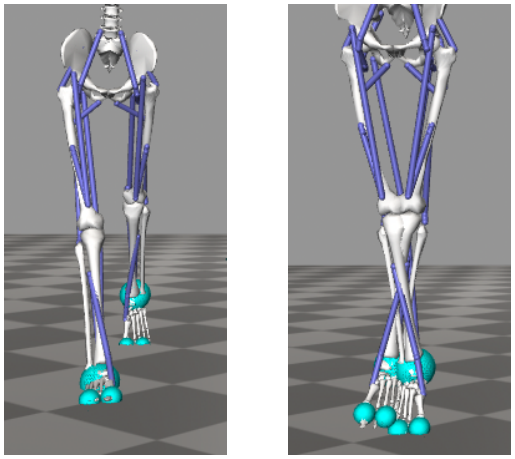


Figure 4.3: Gait pattern of different runs plotted as joint angle over the percent of the gait cycle. Imitation data ± 1 standard deviation is shown in grey.



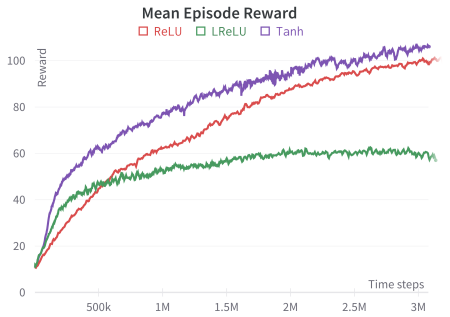
(a) Swing phase

(b) Terminal stance phase

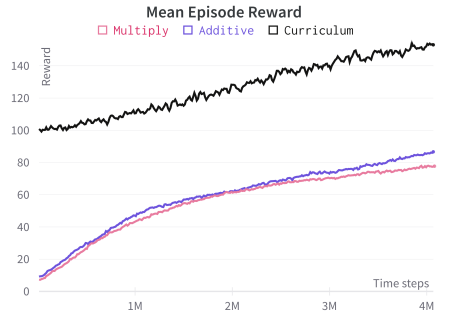
Figure 4.4: Physically impossible gait pattern: The agent swings its left leg through the standing right leg from the initial contact phase to the terminal stance phases of the gait.

vealed that the gait was lacking appropriate knee and hip flexion. Both indicators used to quantify policy erraticness, continuity cost $\mathcal{C}$, and resulting torques at the knee and ankle M_x, could be decreased by regularization. The Activation category achieved the lowest scores for both variables out of all categories. Thus, it can be concluded that all applied regularization methods are effective for decreasing continuity cost, and that reward regularization and activation functions can lead to more human-like torques.

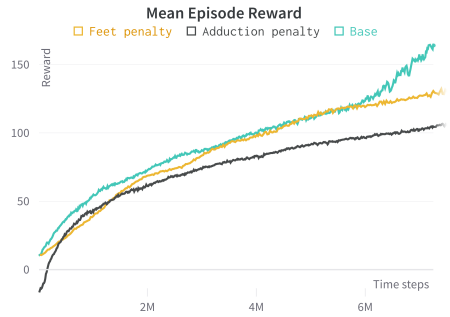
However, the lack of stable gaits in most runs forms an important caveat to this conclusion. In the bigger picture, the goal is to apply DRL architectures to physical prostheses. With this in mind, achieving smoother policies is a necessary but by no means sufficient condition for the application of DRL. Looking at the results from this table through the lens of application, the Curriculum and Feet penalty runs seem to be the best results. The Curriculum run learned a gait pattern that lead to the longest episodes with a stable gait, lower continuity cost, and lower M_x than the Base run. On the other side, the Feet penalty run also achieved more than one gait cycle and in addition featured the torques that were most similar to the imitation data. Because all simulation runs in this study were conducted on a healthy musculoskeletal model, it might be possible that an amputee model with a prosthesis reacts differently to the regularization.



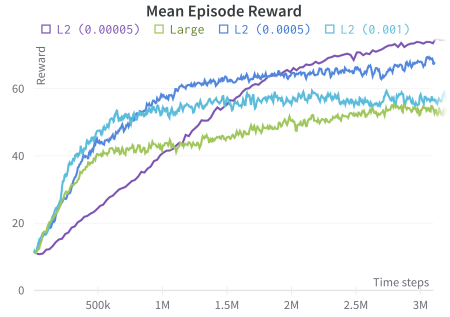
(a) Activation



(b) Reward



(c) Tune



(d) Layer

Figure 4.5: Rewards received during the training process, grouped by run category.

The totality of all runs, as pointed out in Section 4, hints at a trade-off between regularization and performance. Ideally, such a trade-off should not have to be made in real applications. Therefore, the metrics that were used here to quantify erraticness might be invalid. Continuity cost which is adapted from Raffin et al. (2020) was originally formulated for robotic control, not human musculoskeletal walking. To exemplify that directly incorporating continuity cost into the objective may be harmful, the reward function of the Multiply run (Equation 3.12) multiplies the goal and imitation reward by a continuity cost penalty and thereby emphasizes regularization a lot. As PPO maximizes its cumulative reward, the agent learns to produce actions with low continuity cost to satisfy the multiplicative constraint in the reward function. This is also evident in Table 4.1 which shows that the Multiply run had the second lowest continuity cost overall. However, Multiply did not result in a stable gait and had the shortest episodes within its category. Another possible explanation for the regularization-performance trade-off is that shifting emphasis on reducing continuity cost takes away from the importance of goal and imitation reward. This is the case for the Additive and Multiply runs. As shown in Equation 3.14, the weights for goal and imitation reward are decreased in favor of the continuity cost weight. For increasing the continuity cost term, the stability or similarity of the gait is extraneous.

An interesting finding in the results is that the Layer runs perform worst in terms of M_x. Instead of the dissimilar torques being a consequence of the agents not learning a stable gait, the torques may be the reason for the agents failing to walk.

5.1 Limitations

OpenSim One of the limitations is the physics simulation software OpenSim that was used in this study. Despite its widespread use for musculoskeletal modeling, OpenSim has been shown to be much slower than the simulation engine MuJoCo in terms of wall time needed for simulation steps (Ikkala

and Hämäläinen, 2020). Developed by Todorov et al. (2012), it has the disadvantage that it does not come with several musculoskeletal models as OpenSim does. However, Ikkala and Hämäläinen converted a musculoskeletal model from OpenSim to MuJoCo and demonstrated that MuJoCo can be up to 600 times faster than OpenSim for simulations of musculoskeletal models. Because simulation duration showed to be the bottleneck time-wise during the training process, such a speed-up would allow for either much longer training runs, or more total training runs. Both options would likely improve the final outcomes. A second limitation is not caused by OpenSim directly but rather by the way the geometry of the musculoskeletal model is constructed and interpreted by OpenSim. Specifically, the model's `bodies` (bones, etc.) do not have contact meshes and therefore cannot collide with each other in OpenSim. This allows behavior as illustrated in Figure 4.4, where the agent moves its legs through each other. This training outcome is clearly undesirable because it does not transfer to any real-life application. To prevent such behavior in subsequent runs, additional components were added to the reward function that penalize excessive adduction and crossing of the feet. Although subsequent runs showed fewer occasions of body parts moving through each other, the altered and increased complexity of the reward function could have contributed to the difficulty of the control problem. If the physical simulation engine would cause the agent to fall every time it tries to move a body part through another one, all stable gaits would necessarily lack such behavior. This would very likely simplify the training process.

Lee (2020) and Jang and Son (2019) both found that ReLU-like activation functions outperform Tanh in terms of received reward. Figure 4.5a does not follow this trend, the Tanh activation function shows the highest rewards. One possible reason for the disagreeing findings is that previous work tested activation functions on generally less complex RL tasks and that their results, therefore, do not generalize well to musculoskeletal control.

Other limitations of this work involve the implementation of layer regularization. Experiments in Liu et al. (2019) show that regularizing only the policy network lead to better outcomes than applying the regularization methods to both networks, especially for harder tasks. However, the implementation framework for PPO that was used in this study, *Stable Baselines 3* by Raffin et al. (2021), does not provide a straightforward method for regularizing only the policy network. Therefore, all layer regularization and activation functions were applied to both networks in this study. This could possibly explain why Layer runs did not perform better.

Furthermore, the results in Table 4.1 can only be interpreted with care because not all runs allow direct comparisons. Some runs within the same category differ in more than one dimension. For example, all Layer runs except L_2 0.00005 were trained with four parallel environments while 16 parallel environments were used for L_2 0.00005 due to limited availability of computational resources. Such differences can impact the training process and thus should be considered when drawing conclusions from the results. For reference, Table 3.3 provides an overview of all conducted runs. Corresponding policies are explained in Table 3.4, and Tables 4.1 and 4.2 show the results from the runs.

Generalization Although the grand purpose of this study considers amputees and prostheses, all runs were conducted on a healthy musculoskeletal model. Afterward, one single simulation was run with the amputee model to test whether the results generalize easily from a healthy to a prosthetic model. The prosthetic model has a simulated transfemoral prosthesis that replaces the vasti, tibialis anterior, soleus, and biceps femoris muscles of the model's left leg with actuators. The observation space was adjusted such that observations of body parts that were no longer present in the prosthetic model were replaced by their counterparts. This simulation did not result in any stable gait pattern. The result provides no grounds for extrapolating the effectiveness of regularization tech-

niques for healthy models to prosthetic models.

5.2 Future work

This study provides a promising stepstone for further research. As the results show, classic regularization techniques from Deep Learning, as well as reward shaping for regularization, and choice of activation function impact continuity cost of actions and resulting torques. However, further research should confirm or disprove whether continuity cost is a valid indicator of policy erraticness for musculoskeletal modeling. Furthermore, more work is needed to optimize the apparent regularization-performance trade-off. Lastly, expanding the advances from Ikkala and Hämäläinen (2020) to migrate musculoskeletal models from OpenSim to MuJoCo offers a promising boost in time efficiency.

6 Conclusions

This paper used PPO combined with imitation learning for musculoskeletal simulations of human walking with a 22 muscle model. Additionally, several regularization methods from L_2 regularization to reward shaping and different activation functions were tested for their ability to decrease the erraticness of learned policy. The results demonstrate that all three categories of methods were successful at decreasing the measures of erraticness, continuity cost, and resulting joint torque compared to baseline. However, it also became evident that there exists a trade-off between decreasing these measures and learning a stable, human-like gait.

Acknowledgements I hereby want to thank my supervisor Prof. Dr. Raffaella Carloni for the support and guidance I received during this study's completion. In addition, I want to thank Rutger Luinge, Elena Poeltuijn, and Robin Kock, all fellow B.Sc. students at the University of Groningen, for inspiring discussions and advice about the implementation of PPO. Lastly, I thank Rik Timmers, University of Groningen research assistant, for technical support with computation hardware.

References

Anand, A. S., Zhao, G., Roth, H., and Seyfarth, A. (2019). A deep reinforcement learning based approach towards generating human walking behavior with a neuromuscular model. In *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, pages 537–543. IEEE.

Asadulaev, A., Kuznetsov, I., Stein, G., and Filchenkov, A. (2020). Exploring and exploiting conditioning of reinforcement learning agents. *IEEE Access*, 8.

Camargo, J., Ramanathan, A., Flanagan, W., and Young, A. (2021). A comprehensive, open-source dataset of lower limb biomechanics in multiple conditions of stairs, ramps, and level-ground ambulation and transitions. *Journal of Biomechanics*, 119:110320.

De Vree, L. and Carloni, R. (2021). Deep reinforcement learning for physics-based musculoskeletal simulations of healthy subjects and transfemoral prostheses' users during normal walking. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 29:607–618.

Delp, S. L., Anderson, F. C., Arnold, A. S., Loan, P., Habib, A., John, C. T., Guendelman, E., and Thelen, D. G. (2007). Opensim: open-source software to create and analyze dynamic simulations of movement. *IEEE transactions on biomedical engineering*, 54(11):1940–1950.

Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., Wu, Y., and Zhokhov, P. (2017). OpenAI baselines. `https://github.com/openai/baselines`.

Dubey, S. R., Singh, S. K., and Chaudhuri, B. B. (2021). A comprehensive survey and performance analysis of activation functions in deep learning. *ArXiv*, abs/2109.14545.

Fey, N. P., Klute, G. K., and Neptune, R. R. (2011). The influence of energy storage and return foot stiffness on walking mechanics

and muscle activity in below-knee amputees. *Clinical Biomechanics*, 26(10):1025–1032.

Gao, X., Si, J., Wen, Y., Li, M., and Huang, H. H. (2020). Knowledge-guided reinforcement learning control for robotic lower limb prosthesis. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 754–760. IEEE.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, 27.

Hill, A. V. (1938). The heat of shortening and the dynamic constants of muscle. *Proceedings of the Royal Society of London. Series B-Biological Sciences*, 126(843):136–195.

Hossny, M., Iskander, J., Attia, M., Saleh, K., and Abobakr, A. (2021). Refined continuous control of ddpg actors via parametrised activation. *AI*, 2(4).

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Ikkala, A. and Hämäläinen, P. (2020). Converting biomechanical models from Opensim to muJoCo. In *International Conference on NeuroRehabilitation*, pages 277–281. Springer.

Jang, S. and Son, Y. (2019). Empirical evaluation of activation functions and kernel initializers on deep reinforcement learning. In *ICTC 2019 - 10th International Conference on ICT Convergence: ICT Convergence Leading the Autonomous Future*.

Kidziński, Ł., Ong, C., Mohanty, S. P., Hicks, J., Carroll, S., Zhou, B., Zeng, H., Wang, F., Lian, R., Tian, H., et al. (2020). Artificial intelligence for prosthetics: Challenge solutions. In *The NeurIPS'18 Competition*, pages 69–128. Springer.

Lawson, B. E. and Goldfarb, M. (2014). Impedance & admittance-based coordination control strategies for robotic lower limb prostheses. *Mechanical Engineering*, 136(09):S12–S17.

Lee, D. (2020). Comparison of reinforcement learning activation functions to improve the performance of the racing game learning agent. *Journal of Information Processing Systems*, 16(5).

Liu, Z., Li, X., Kang, B., and Darrell, T. (2019). Regularization matters in policy optimization. *arXiv:1910.09191 [cs, stat]*.

Martinez-Villalpando, E. C. (2012). *Design and evaluation of a biomimetic agonist-antagonist active knee prosthesis*. PhD thesis, Massachusetts Institute of Technology.

Mysore, S., Mabsout, B., Saenko, K., and Mancuso, R. (2021). How to Train Your Quadrotor: A Framework for Consistently Smooth and Responsive Flight Control via Reinforcement Learning. *ACM Transactions on Cyber-Physical Systems*, 5(4):1–24.

Ng, A. (2004). Feature selection, l1 vs. l2 regularization, and rotational invariance. *Proceedings of the twenty-first international conference on Machine learning*.

Nowakowski, K., Carvalho, P., Six, J.-B., Maillet, Y., Nguyen, A. T., Seghiri, I., M'pemba, L., Marcille, T., Ngo, S. T., and Dao, T.-T. (2021). Human locomotion with reinforcement learning using bioinspired reward reshaping strategies. *Medical & Biological Engineering & Computing*, 59(1):243–256.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N. (2021). Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8.

Raffin, A., Kober, J., and Stulp, F. (2020). Smooth exploration for robotic reinforcement learning. In *Smooth Exploration for Robotic Reinforcement Learning*.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015a). Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. (2015b). High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Seth, A., Hicks, J. L., Uchida, T. K., Habib, A., Dembia, C. L., Dunne, J. J., Ong, C. F., DeMers, M. S., Rajagopal, A., Millard, M., Hamner, S. R., Arnold, E. M., Yong, J. R., Lakshmikanth, S. K., Sherman, M. A., Ku, J. P., and Delp, S. L. (2018). Opensim: Simulating musculoskeletal dynamics and neuromuscular control to study human and animal movement. *PLoS Computational Biology*, 14.

Shen, Q., Li, Y., Jiang, H., Wang, Z., and Zhao, T. (2020). Deep reinforcement learning with robust and smooth policy. In *37th International Conference on Machine Learning, ICML 2020*, volume PartF168147-12.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Thodoroff, P., Pineau, J., Durand, A., and Precup, D. (2018). Temporal regularization in Markov Decision Process. In *Advances in Neural Information Processing Systems*, volume 2018-December.

Todorov, E., Erez, T., and Tassa, Y. (2012). MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033.

van Wouden, M. (2021). Deep reinforcement learning for physics-based musculoskeletal simulations of an osseointegrated tranfemoral amputee model during normal walking. Bachelor's thesis.

Weng, J., Hashemi, E., and Arami, A. (2021). Natural walking with musculoskeletal models using deep reinforcement learning. *IEEE Robotics and Automation Letters*, 6(2):4156–4162.