Supplementary Material for Learning with Muscles: Benefits for Data-Efficiency and Robustness in Anthropomorphic Tasks

A Muscle model

In this section, the implementations of the muscle models for demoa and MuJoCo are described. The demoa model approximates biological muscles with more physiological detail and accuracy, whereas the simpler MuJoCo model allows the simulation of rudimentary muscular properties at minimal computational cost, rendering it usable for machine learning.

A.1 Muscle model in Mujoco

Even though the MuJoCo simulator includes the capability of simulating muscles, it requires the explicit definition of tendon insertion points and wrapping surfaces for each model. We, therefore, use our own muscle implementation for the MuJoCo experiments, that does not contain tendons. As a direct consequence, the muscle-fiber length is *uniquely* determined by the joint angle.

In the following, we describe activation dynamics, definitions of muscle-fiber length and velocity, the computation of the resulting torque and the parametrization.

Muscle-tendon-unit Each controllable joint of the MuJoCo model is actuated by two monoarticular muscles and we do **not** compute tendon length. We assume that:

$$l_{\mathrm{MTU}} = l_{\mathrm{CE}}, \tag{1}$$

where l_{MTU} is the length of the entire muscle-tendon-unit and l_{CE} is the length of the muscle fiber, or contractile element. We define muscle-fiber length and velocity by a linear equation [1, 2, 3]:

$$l_{\mathrm{CE},i} = m_i\,\phi_j + l_{\mathrm{ref},i} \tag{2}$$

$$\dot{l}_{\mathrm{CE},i} = m_i\,\dot{\phi}_j, \tag{3}$$

where ϕ_j is the joint angle, m_i and $l_{\mathrm{ref},i}$ are computed from user-defined parameters, and $i \in \{1, 2\}$, as we assume two antagonistic muscles per joint. The parameter m_i acts as a constant moment arm in our model, see Eq. 6.

Activation dynamics The evolution of muscle activity obeys the following first-order ordinary differential equation:

$$\dot{a}(t) = \frac{1}{\Delta t_a}(u(t) - a(t)), \tag{4}$$

where $u(t)$ is a control signal.

Muscle force Given the previous quantities, the muscle force is computed by:

$$F_i = \Big[\mathrm{FL}(l_{\mathrm{CE},i})\,\mathrm{FV}(v_{\mathrm{scale}}\,\dot{l}_{\mathrm{CE},i})a_i + \mathrm{FP}(l_{\mathrm{CE},i})\Big]\; F_{\max}, \tag{5}$$

where v_{scale} is a scaling parameter to adjust in which region of the force-velocity (FV)-curve typical fiber velocities operate. We can then compute the resulting joint torque:

$$\tau = -(m_1\,F_1 + m_2\,F_2). \tag{6}$$

The functions FL, FV and FP are given by MuJoCo internal functions that phenomenologically model experimental data and are applied to normalized muscle lengths and velocities, see MuJoCo documentation [4] and Fig. 1.

Parametrization As the there is a one-to-one mapping of joint angle to muscle lengths in our model, we can determine the required parameters m_i and $l_{\mathrm{ref},i}$, if a mapping of $l_{\min}$ to $\phi_{\min}$ and $l_{\max}$ to $\phi_{\max}$ is specified (assuming $l_{\max}$ to be the maximal muscle-fiber length l_{CE}). Inserting them into

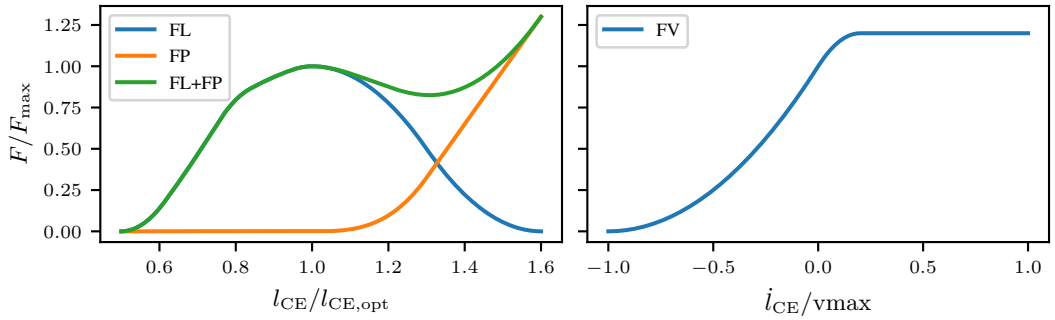

Figure 1: Force-length (FL) and force-velocity (FV) relationships and passive force (FP) used in MuJoCo [4]. We use the same phenomenological functions in our own MuJoCo muscle model. While FL and FV get scaled by the current muscle activity a, FP does not (see Eq. 5).

Table 1: Parameters for the MuJoCo muscle morphology.

(a) Muscle parameters

Parameter	**Value**
$l_{\max}$	1.05
$l_{\min}$	0.95
$\phi_{\max}$	$\pi/2$ [rad]
$\phi_{\max}$	-$\pi/2$ [rad]
Δt_a	0.01 [s]
v_{scale}	0.5

(b) Maximum isometric force

Task	**Value**
ArmMuJoCo	295 [N]
Biped	5000 [N]

Eq. 1 and solving the resulting system of equations gives:

$$m_1 = \frac{l^{\max} - l^{\min}}{\phi^{\max} - \phi^{\min} + \epsilon} \tag{7}$$

$$l_1^{\mathrm{ref}} = l_{\min} - m_1\, \phi_{\min} \tag{8}$$

$$m_2 = \frac{l^{\max} - l^{\min}}{\phi^{\min} - \phi^{\max} + \epsilon} \tag{9}$$

$$l_2^{\mathrm{ref}} = l_{\min} - m_2\, \phi_{\max} \tag{10}$$

$$\tag{11}$$

All in all, $\phi^{\min}$, $\phi^{\max}$, $l^{\min}$, $l^{\max}$ and $F_{\max}$ are required to be specified. The constant $\epsilon = 0.01$ ensures numerical stability. We use the same parametrization for each MuJoCo task, see Table 1.

The maximum and minimum joint angles were chosen to allow for a large range of motion. They do not constitute hard limits, but the passive elastic force FP will increase strongly when reaching them. The maximum and minimum fiber lengths are identical to the MuJoCo default values. As we want to study the benefits of muscular properties in learning, we chose the time and velocity scales $\Delta t_a = 0.01$ and $v_{\mathrm{scale}} = 0.5$ to be large enough to produce noticeable effects, such as low-pass filtering and self-stabilization properties, across all performed tasks. To determine maximum muscle forces, we trained muscle-actuator policies for a chosen maximum force value, after which we adjusted maximum torque-actuator forces to be identical or slightly larger to the maximally observed muscle forces in the final task policies. We repeated this procedure with different force values until good performance could be observed for both morphologies, see Suppl. E.1 for an evaluation across different force values. All other MuJoCo internal parameters related to muscle modeling are kept to their default values.

In practice, we implement the muscle model in Cython [5], interfacing with OpenAI gym [6], which achieves similar execution speed to native MuJoCo.

A.2 Muscle model in demoa

The muscle model implemented in demoa [7] includes additionally visco-elastic, passive tendon characteristics and muscle routing as joint angle-dependent lever arms to account for many physiological details. In the following, we describe the activation and contraction dynamics of the muscle model, as well as the tendon characteristics and the nonlinear lever arms.

Activation dynamics The muscles are activated with the learned and optimized control signal u, which is nonlinearly transformed into an activation signal. The activity a is following a first-order differential equation of normalized calcium ion concentration γ as introduced by Hatze [8] and simplified by Rockenfeller et al. [9, 10]:

$$\dot{\gamma}(t) = M_{\mathrm{H}}(u(t) - \gamma(t)) \tag{12}$$

and a nonlinear mapping onto the muscles activity

$$a(t) = \frac{a_0 + \varpi}{1 + \varpi}, \tag{13}$$

with $\varpi(\gamma(t), l_{\mathrm{CE}}(t)) = (\gamma(t) \cdot \rho(l_{\mathrm{CE}}))^{\nu}$ and $\rho(l_{\mathrm{CE}}) = \varpi_{\mathrm{opt}} \cdot \frac{l_{\mathrm{CE}}}{l_{\mathrm{opt}}} = \gamma_{\mathrm{c}} \cdot \rho_0 \cdot \frac{l_{\mathrm{CE}}}{l_{\mathrm{opt}}}$. The parameter values are chosen muscle non-specifically and are given in the description of the models (see [11, 12]).

Muscle-tendon-unit The predicted forces are modeled using Hill-type muscle models [13] including four spring-damper components (see Fig. 2): The contractile element (CE) models the active force production of biological muscle fibers, including the nonlinear *force-length* and nonlinear *force-velocity* relation. The parallel elastic element (PEE) models the passive connective tissue in the muscle belly and is arranged in parallel to the CE. The visco-elastic properties of the tendons are modeled using a serial elastic element (SEE) and a serial damping element (SDE). All in all, the governing model dependencies for all muscles $i = 1, ..., n$ are:

$$\dot{l}_{\mathrm{CE},i} = f_{\mathrm{CE}}(l_{\mathrm{CE},i}, l_{\mathrm{MTU},i}, \dot{l}_{\mathrm{MTU},i}, a_i) \tag{14}$$

$$\dot{a}_i = f_a(a_i, u_i, l_{\mathrm{CE},i}) \tag{15}$$

$$f_{\mathrm{MTU},i} = f_{\mathrm{MTU},i}(l_{\mathrm{MTU},i}, \dot{l}_{\mathrm{MTU},i}, l_{\mathrm{CE},i}, a_i)\,, \tag{16}$$

where the first differential equation (Eq. 14) denotes the contraction dynamics which models the velocity $\dot{l}_{\mathrm{CE}}$ of the contractile element. This contraction velocity is dependent on the current CE length l_{CE}, the length and contraction velocity of the muscle-tendon unit l_{MTU} and $\dot{l}_{\mathrm{MTU}}$ respectively, and the activity a. The latter is modeled by the activation dynamics (see Eq. 12,13,15). Finally, a force $f_{\mathrm{MTU},i}$ for each muscle is produced which is translated into joint torques.

Nonlinear lever arms To translate the force into joint torques, the muscle path around the joints is routed via deflection ellipses in demoa [14]. If the length of the half-axises of all ellipses are set to zero, this approach can be simplified to the more commonly used fixed via-point approach for muscle routing. Based on the resulting moment arms of the muscles, the force f_{MTU} is translated to generalized torques acting on the degrees of freedom of the system.

State of the system Using a musculoskeletal model with a Hill-type muscle model, as described in this section, increases the number of state variables because two additional differential equations need to be solved for each included muscle. The entire state vector x can therefore be formulated as:

$$x \in \mathbb{R}^{2n_{\mathrm{musc}}+2n_\theta} = \{\gamma_i, l_{\mathrm{CE},i}, \theta_j, \dot{\theta}_j\} \tag{17}$$

where θ and $\dot{\theta}$ represent the generalized joint angle coordinates and their respective velocities, and n_{musc} and n_θ denote the number of muscles and the number of joints, respectively.

B Experimental details and hyperparameters

In this section, we describe algorithm implementation details while also reporting additional settings that were used to obtain previously shown results, as well as used hyperparameters. The section is divided such that experiment details are shown with the control algorithm that was used to generate the results.

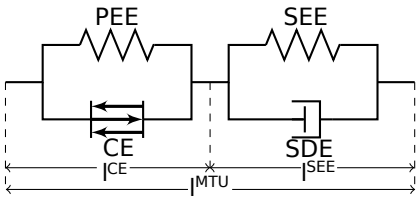

Figure 2: The muscle model in demoa is modeled as lumped Hill-type muscle model (figure adapted from Haeufle et al. [13]).

B.1 Optimal Control (OC)

In the optimal control case, we used the covariance matrix adaptation evolution strategy (CMA-ES) [15] to find the control policy $u(k)$. As mentioned in the main paper, we chose the same population size and number of generations for both actuator morphologies to allow for a fair comparison, even though the number of decision variables n_u is always larger in the muscle-actuated case. In all cases, if not otherwise mentioned, we use a fixed population size of 36 and a fixed number of generations of 100 while varying the control resolution c. If the control resolution is refined, this correlates to an increase in the number of decision variables n_u, however, we specifically did not change the population size or generation number because we wanted to compare the data-efficiency and learning with limited resources for the chosen actuators. The temporal control resolution c was typically varied for $c = \{0.05, 0.15, 0.3\}$ s. The upper bound of these control resolutions ($c = 0.3$ s) corresponds to a triphasic control pattern for a typical movement duration of 0.9 s as it was selected in the smooth point-reaching and squatting task. This selection of c was inspired by biological experiments, where it was shown that triphasic patterns occur in muscle surface electromyograms in typical point-reaching movements (e.g. see [16, 17]). The main hyperparameter of the CMA-ES algorithm σ was set to the default value of 0.2 if not otherwise stated.

B.2 Model Predictive Control MPC

We employed a warm start procedure using the CMA-ES optimizer and afterwards started the MPC routine with a local optimizer BOBYQA [18] (part of the standard python optimization package NLOPT). As temporal control resolution in this closed-loop setting, we chose a very fine resolution of $c = 0.01$ s, similar to the RL setup. This allows counteracting perturbations. The prediction horizon was varied between $t_{\text{pred}} = \{0.2, 0.3, 0.4, 0.5\}$ s as shown in the result section of the main paper.

B.3 Reinforcement Learning (RL)

We use the RL algorithm MPO [19], implemented in TonicRL [20]. Hyperparameters were optimized with a simplified in-house CEM optimizer. All RL experiments are averaged over 8 random seeds except for the hyperparameter optimization, which would have been computationally intractable. Each experimental run was computed with 1 NVIDIA V100 GPU and 20 CPUs of varying speed and type. We use a fixed control resolution of $c = 0.01$ s for all RL experiments.

B.3.1 Experimental details

We give further experimental details in this section.

Data-efficiency For the point-reaching experiments, we used the hyperparameters that were found in the meta-optimization (see Fig. 6) for both morphologies. For the hopping task, we used default MPO parameters. The updated learning curves with optimized parameters, as well as additional results on hyperparameters and maximum force settings can be found in Suppl. E.

Hyperparameter optimization We optimized the performance of both actuator morphologies in the precise point-reaching task in MuJoCo (see Fig. 5). For each iteration, N_{sets} sets of random parameters are drawn from fixed normal and log-normal distributions. For each of these sets, the task performance is evaluated after T_{train} environment interactions, where T_{train} is chosen such that a noticeable increase in performance can be observed with both actuator morphologies. After each iteration, M_{elite} elite parameter sets are chosen and the mean and standard deviation of each parameter-generating distribution is updated by fitting a (log-)normal distribution to the M_{elite} elite sets with maximum-likelihood estimation. See Table 2 for exact specifications. The meta-optimization for hopping can be found in Fig. 6.

Table 2: Settings for the hyperparameter search.

(a) Precise point-reaching

Parameter	Value
N_{sets}	50
T_{train}	2×10^6
M_{elite}	10

(b) Hopping

Parameter	Value
N_{sets}	20
T_{train}	5×10^6
M_{elite}	10

(c) Initial distributions

Parameter	Distribution	Bounds
lr_a	truncated log-normal	$[2.5 \times 10^{-4}, 3 \times 10^{-2}]$
lr_c	truncated log-normal	$[0.5 \times 10^{-2}, 10^{-1}]$
lr_d	truncated log-normal	$[0.5 \times 10^{-2}, 1]$
clip_a	truncated log-normal	$[10^{-7}, 10^{-4}]$
clip_c	truncated log-normal	$[10^{-7}, 10^{-4}]$

In the experiments, we only used truncated log-normal distributions to generate parameters. The samples were clipped to the bounds given in Table 2 and initial mean and standard deviation were chosen to lie inside the bounded interval. More precisely, we defined $\log(\mu) = (a + b)/2$ and $\log(\sigma) = (b - a)/4$, where a and b are the chosen bounds. The chosen parameters were the actor learning rate lr_a, the critic learning rate lr_c, the learning rate of the dual optimizer lr_d, the gradient clipping threshold for the actor clip_a and the critic clip_a.

Robustness point-reaching We trained policies with both morphologies in precise point-reaching for 1.5×10^7 iterations. The best performing policies were then evaluated for the perturbation experiments. For dynamic load, the mass of the hand is increased by 1.5 kg to simulate an object. For chaotic load, a ball with radius 0.12 m and a density of 1000 kg/m^3 is attached to a cable of length 0.6 m, that is connected to the hand. We sample 10 random goals from the training distribution and visualize three trajectories such that there are no overlapping paths. All 10 goals are shown in Fig. 4.

Robustness hopping We trained policies for both morphologies for hopping with the hyperparameters obtained in Fig. 6 and for 1.5×10^7 iterations. We then record 100 evaluation episodes where random forces drawn from $F_i \sim \mathcal{N}(\cdot|0, \sigma_F)$ are applied with a probability of $p = 0.05$ to the hip, knee and ankle joints and to the pelvis position and rotation. Center of mass trajectories are shown for an interval of 15 s in the main manuscript, black vertical bars mark episode resets due to extreme angles of the biped, which would cause it to fall to the ground. The performance for each perturbation level is divided by the unperturbed performance for each morphology to yield a relative performance comparison.

B.3.2 Hyperparameters

The hyperparameters for all RL tasks were set to the best performing runs in the shown hyperparameter optimization. They best trained policies were then used for the perturbation experiments.

C Models

We give detailed descriptions of the used models in this section. See Table 4 for more information about the MuJoCo models.

C.1 Arm

The Arm model consists of two segments connected with hinge joints moving against gravity. The ArmDemoa [11] is freely available using the multi-body software demoa [7]. In the muscle-actuated case, six muscles were included, modeled as Hill-Type muscles (A.2). Here, two monoarticular muscles, each for the shoulder and elbow joint, and two biarticular muscles acting on both joints are included. The segments are modeled as rigid bodies, and the dynamics are solved using the Euler-Lagrange equation. In the torque-actuated case, each joint is driven by one torque actuator. For more details on the demoa model, we refer to the Technical Report [11]. The variant ArmMuJoCo was derived from an implementation of Arm26 included in MuJoCo [4], it was modified to yield a

Table 3: RL parameters for MPO in TonicRL for the different tasks. Non-reported values are left to their default setting in TonicRL [20]. Common MPO settings are equal for all experiments.

(a) Point-reaching MPO (muscle)

Parameter	Value
$\mathrm{lr_a}$	3×10^{-4}
$\mathrm{lr_c}$	10^{-3}
$\mathrm{lr_d}$	2×10^{-2}
$\mathrm{clip_a}$	4×10^{-5}
$\mathrm{clip_c}$	3×10^{-5}
batch size	100
return-normalizer	No

(b) Point-reaching MPO (torque)

Parameter	Value
$\mathrm{lr_a}$	10^{-3}
$\mathrm{lr_c}$	5×10^{-3}
$\mathrm{lr_d}$	8.2×10^{-3}
$\mathrm{clip_a}$	7×10^{-6}
$\mathrm{clip_c}$	10^{-6}
batch size	100
return-normalizer	No

(c) Hopping MPO (muscle and torque)

Parameter	Value
$\mathrm{lr_a}$	3×10^{-4}
$\mathrm{lr_c}$	3×10^{-4}
$\mathrm{lr_d}$	10^{-2}
$\mathrm{clip_a}$	None
$\mathrm{clip_c}$	None
batch size	256
return-normalizer	Yes

(d) Hopping perturbation (muscle)

Parameter	Value
$\mathrm{lr_a}$	9×10^{-4}
$\mathrm{lr_c}$	3×10^{-3}
$\mathrm{lr_d}$	10^{-2}
$\mathrm{clip_a}$	10^{-5}
$\mathrm{clip_c}$	3×10^{-7}
batch size	256
return-normalizer	Yes

(e) Hopping perturbation (torque)

Parameter	Value
$\mathrm{lr_a}$	10^{-3}
$\mathrm{lr_c}$	7×10^{-4}
$\mathrm{lr_d}$	2×10^{-2}
$\mathrm{clip_a}$	2×10^{-5}
$\mathrm{clip_c}$	10^{-6}
batch size	256
return-normalizer	Yes

(f) Common MPO settings

Parameter	Value
buffer size	10^6
steps before batches	5×10^4
steps between batches	50
number of batches	50
n-step return	3
n parallel	20
n sequential	10

Table 4: State information for all MuJoCo environments. The elements actuator lengths and velocities are directly derived from MuJoCo internal attributes `actuator_length` and `actuator_velocity` and keep the two morphologies as consistent as possible.

model	observations
ArmMuJoCo (muscle)	joint positions, joint velocities, muscle positions, muscle velocities, muscle forces, muscle activities, goal position, hand position
ArmMuJoCo (torque)	joint positions, joint velocities, actuator positions, actuator velocities, actuator forces, goal position, hand position
Biped (muscle)	joint positions, joint velocities, muscle lengths, muscle velocities, muscle forces, muscle activities, head position, pelvis position, torso angle, scaled COM-velocity
Biped (torque)	joint positions, joint velocities, actuator lengths, actuator velocities, actuator forces, head position, pelvis position, torso angle, scaled COM-velocity

torque-variant similar to [21]. We additionally created a muscle-variant consisting of 2 muscles per joint. The maximum torques for the torque actuators were matched to the highest achieved torques by the trained muscle policies for both versions independently.

C.2 Biped

We converted the geometrical model of an OpenSim bipedal human without arms [22] for use in MuJoCo. The model, consisting of 7 controllable joints (lower back, hip, knee, ankle) moves in a 2D-plane. Each joint is actuated by two antagonistic muscles or one idealized torque actuator. During execution, we only allow control signals for one leg, the actions for the other leg are kept identical to the first one. This incentivizes symmetric hopping motions, even though both legs can still move differently due to differing initial configurations or external forces. The maximum torques for the torque actuators were matched to the highest achieved torques by the trained muscle policies.

C.3 FullBody

For the squatting and high-jumping task, we used the FullBody (allmin) model [12] which is freely available using the multi-body software demoa [7]. It consists of two legs and an upper body with a skeletal geometry similar to humans and moves in 3D. The ankle, knee and hip joints, as well as a lumbar and a cervical spine joint are controllable (8 controllable joints). The model also consists of two arms with their respective joints, however, these joints were not controlled in this study. In total, 14 joints are modeled with 20 degrees of freedom. Each controllable joint was either actuated by two muscles (A.2) set up in an agonistic-antagonistic setup (muscle-actuated case) or by one idealized torque actuator (torque-actuated case). The maximum allowed torques were matched to the highest torques that occurred in the optimization in the muscle-actuated case to allow for a fair comparison. Only monoarticular muscles (spanning one joint) were used. Furthermore, we reduced the number of control inputs n_u for this study by using symmetrical control signals for the left and right legs. Additional to the torques generated by the actuators, also joint limitations are modeled as linear one-sided spring-damper elements. We refer to the Technical Report [12] for more details.

D Tasks

We chose movement objectives which represent both, robotic challenges and naturally observed movements of humans.

Smooth point-reaching (OC/MPC) This task encourages smooth point-reaching. Therefore, the objective minimizes the L2-error between the desired angle endpoint and the desired joint angle velocity, as well as penalizing the angle jerk to ensure a smooth motion. The objective for smooth point-reaching is given by:

$$\varepsilon = \frac{\omega_i}{S_\mathrm{i}}(\theta_\mathrm{i} - \theta_\mathrm{i}^\mathrm{des})^2 + \frac{\omega_i}{S_\mathrm{i}}(\dot{\theta}_\mathrm{i} - \dot{\theta}_\mathrm{i}^\mathrm{des})^2 + \dddot{\theta}^2, \tag{18}$$

where θ_i denotes the joint angle, $\dot{\theta}_\mathrm{i}$ the joint angle velocity and the last term $\dddot{\theta}$ penalizes the angle jerk to ensure a smooth motion. ω_i and S_i are weighting and scaling parameters, respectively. Their values (shoulder and elbow) are given in Table 5. The scaling parameters were chosen based on

Table 5: Parameters for cost functions of OC/MPC tasks.

(a) Scaling parameters

parameter	value
$S_{\theta,\text{sh}}$	2.45 [rad]
$S_{\theta,\text{elb}}$	2.45 [rad]
$S_{\theta,\text{hip}}$	1.92 [rad]
$S_{\theta,\text{knee}}$	2.11 [rad]
$S_{\theta,\text{ank}}$	1.05 [rad]
$S_{\theta,\text{ls}}$	0.52 [rad]
$S_{\theta,\text{cs}}$	1.05 [rad]
$S_{\dot{\theta},\text{sh}}$	18.7 [rad/s]
$S_{\dot{\theta},\text{elb}}$	27.9 [rad/s]
$S_{\dot{\theta},\text{hip}}$	14.1 [rad/s]
$S_{\dot{\theta},\text{knee}}$	28.4 [rad/s]
$S_{\dot{\theta},\text{ank}}$	12.6 [rad/s]
$S_{\dot{\theta},\text{ls}}$	5.2 [rad/s]
$S_{\dot{\theta},\text{cs}}$	10.4 [rad/s]

(b) Weighting parameters

parameter	value
$\omega_{\theta,\text{sh}}$	2
$\omega_{\theta,\text{elb}}$	2
$\omega_{\theta,\text{hip}}$	2
$\omega_{\theta,\text{knee}}$	2
$\omega_{\theta,\text{ank}}$	2
$\omega_{\theta,\text{ls}}$	2
$\omega_{\theta,\text{cs}}$	2
$\omega_{\dot{\theta},\text{sh}}$	1
$\omega_{\dot{\theta},\text{elb}}$	1
$\omega_{\dot{\theta},\text{hip}}$	1
$\omega_{\dot{\theta},\text{knee}}$	1
$\omega_{\dot{\theta},\text{ank}}$	1
$\omega_{\dot{\theta},\text{ls}}$	1
$\omega_{\dot{\theta},\text{cs}}$	1

measured upper limits for human joint angular velocity [23] and human joint angle limits (Table 2 in [12]). The desired angle $\theta_{\text{i}}^{\text{des}}$ is set to $90°$ for both the shoulder (sh) and the elbow (elb) joint, as this requires a large motion. The movement duration in this task was set to 0.9 s.

Precise point-reaching (RL) We employ a similar reward function to [24]:

$$r = -\lambda_1(d - \log(d + \epsilon^2)) - \frac{\lambda_2}{N}\sum a_i^2 - 2, \tag{19}$$

where d is the Euclidean distance between end effector and target position, $\epsilon = 10^{-4}$ prevents numerical instabilities, $\lambda_1 = 0.1$ and $\lambda_2 = 10^{-4}$. A smaller distance d increases the overall reward, but in contrast to the usual Euclidean distance, the log-term increases rewards for very small distances even further, incentivizing precision. The episode does not terminate until a time limit of 1000 steps elapses.

Fast point-reaching (RL) This task is identical to the previous one, but, in addition to the time limit, the episode also terminates if the distance between end effector and target position is below 5 cm, which incentivizes reaching speed over precision.

Hitting a ball with a high velocity (OC/MPC) A ball with a mass of 250 g is dropped in front of the arm model and the controller learns to hit the ball with a high velocity by optimizing the following objective:

$$\varepsilon = -\max \dot{z}_{\text{ball}}, \tag{20}$$

where $\dot{z}_{\text{ball}}$ denotes the ball-velocity in z-direction (direction of gravity).

Squatting (OC/MPC) The objective for squatting is given by:

$$\varepsilon = \frac{\omega_i}{S_{\text{i}}}(\theta_{\text{i}} - \theta_{\text{i}}^{\text{des}})^2 + \frac{\omega_i}{S_{\text{i}}}(\dot{\theta}_{\text{i}} - \dot{\theta}_{\text{i}}^{\text{des}})^2, \tag{21}$$

where θ_{i} denotes the joint angle, $\dot{\theta}_{\text{i}}$ the joint angle velocity. ω_i and S_{i} are weighting and scaling parameters, respectively. Their values are given in Table 5. The scaling parameters were chosen based on measured upper limits for human joint angular velocity [23] and human joint angle limits (Table 2 in [12]). The movement duration in this task was set to 0.9 s. This squatting objective is taken from [25], where the desired hip $\theta_{\text{hp}}^{\text{des}}$, knee $\theta_{\text{kn}}^{\text{des}}$ and ankle $\theta_{\text{an}}^{\text{des}}$ joint angle are defined to be:

$$\begin{aligned}
\theta_{\text{an}}^{\text{des}} &= -20°, \\
\theta_{\text{kn}}^{\text{des}} &= \sin^{-1}(-\frac{L_s}{L_t} \cdot \sin(\theta_{\text{an}}^{\text{des}})) - \theta_{\text{an}}^{\text{des}} - \theta_{\text{an},0}, \\
\theta_{\text{hp}}^{\text{des}} &= -\theta_{\text{kn}}^{\text{des}} - \theta_{\text{an}}^{\text{des}}.
\end{aligned}$$

High-Jumping (OC/MPC) The objective for the high-jumping is taken from [26] and maximizes the position and velocity of the centre of mass of the human body model at the time of lift-off t_l. Additionally, we slightly expanded this objective to account for the three-dimensionality of our jumping model by penalizing deviations of the centre of mass in the x and y-direction:

$$\varepsilon = z_{\text{com}}(t_l) + \frac{\dot{z}_{\text{com}}^2(t_l)}{2g} - |(x_{\text{com}}(t_l) - 0)| - |(y_{\text{com}}(t_l) - 0)|. \tag{22}$$

Note, that z_{com} denotes the centre of mass position (CoM) in z-direction (direction of gravity). The model is initialized to start from a squatting position in this task.

Hopping (RL) We developed a reward function that is able to induce hopping in different leg-driven systems and can be applied independently of the actuator morphology. We did not obtain good results with height-based rewards or the gym hopper [6] reward function. The reward for hopping is given by:

$$r = \exp(\max\{0, \hat{v}_z^{\text{COM}}\}) - 1, \tag{23}$$

where v_z^{COM} is the z-velocity of the center of mass. The transformation $\hat{v} = \min\{10, 100\, v\}$ adjusts the sensitivity of the reward function while also preventing numerical overflows of the exponential function. Crucially, large positive velocities are weighted much more strongly than small or negative velocities, driving the system to maximum height periodic hopping. The second term prevents positive rewards for velocities close to zero, as $\exp(0) = 1$. We additionally use regularizing cost terms:

$$r_{\text{reg}} = r_{\text{alive}} - \lambda_1\, r_{\text{action}} - \lambda_2\, r_{\text{joint}}, \tag{24}$$

where $\lambda_1 = 10^{-4}$, $\lambda_2 = 10^{-3}$, r_{alive} is 1 if the episode does not terminate and 0 otherwise, $r_{\text{action}} = \sum a_i^2/N$ punishes large actions and r_{joint} punishes joint angles close to the limits of the system. Specifically:

$$r_{\text{joint}} = \begin{cases} -1, & \text{if } |q_{\max,i} - q_i| < 0.1 \\ -1, & \text{if } |q_{\min,i} - q_i| < 0.1 \\ 0, & \text{otherwise.} \end{cases} \tag{25}$$

Finally, we terminate the episode after the lapse of a time limit of 1000 iterations, or if different parts of the model are very close to the ground, as this indicates a fall. The termination conditions are:

$$\begin{aligned} h_{\text{skull}} &< 0.3\ [\text{m}] \\ h_{\text{pelvis}} &< 0.2\ [\text{m}] \\ h_{\text{tibia}_\text{l}} &< 0.3\ [\text{m}] \\ h_{\text{tibia}_\text{r}} &< 0.3\ [\text{m}] \\ \theta_{\text{torso}} &> 1.22\ [\text{rad}] \\ \theta_{\text{torso}} &< -0.88\ [\text{rad}], \end{aligned}$$

where h is the height of the respective body part and θ_{torso} marks the torso angle deviation from the upright position.

E Additional experiments (RL)

E.1 Maximum force variation

Although the maximum force of the actuators is not freely adjustable in real systems, it is trivial to do so in simulation and has a strong influence on performance. We, therefore, used the biped parameters resulting from our hyperparameter optimization and recorded learning curves for the hopping task for both actuator morphologies for different maximum actuator forces. For each setting, we set the maximum isometric force for all muscle actuators to a certain value, trained the systems to convergence, and then recorded the torque values occurring at each controllable joint during execution of the hopping behavior. We then trained torque-actuator policies while setting $\tau_{\max}$ to the previously observed maximum values for *each* individual joint. The results are shown in Fig. 3. Even though singular torque-driven runs are able to outperform all muscle-driven runs at the end of training, this not only takes a considerable number of learning iterations, but also comes at the cost of strong learning instabilities. Looking at the learned behaviors, the torque-driven policies tend to jump very high, but violate the allowed torso-angles at the peak due to their unstable explorative policies. No periodic hopping could be observed. The muscle-driven policies, on the other hand, achieve periodic hopping, even though the apex hopping height is smaller.

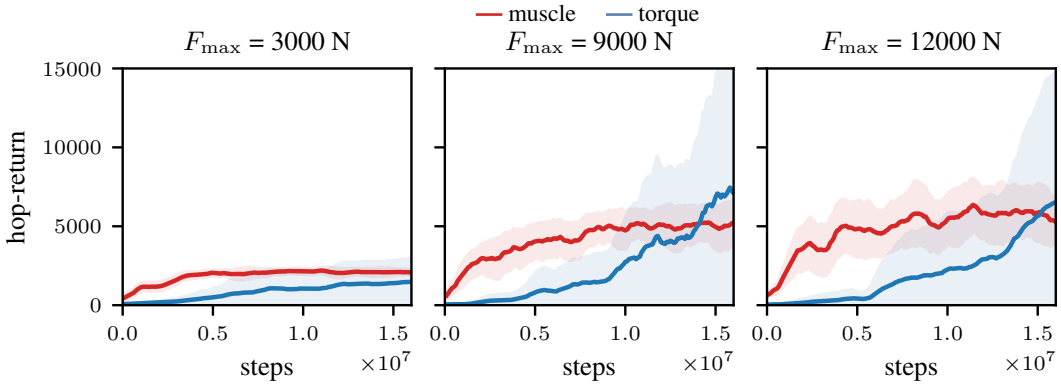

Figure 3: **Hopping performance for different actuator strengths.** Hyperparameters are optimized for hopping with a maximum muscle strengh of $F_{\max} = 5000$ N as used in the previous hopping experiment. The maximum isometric muscle force is set to different values and the policies are trained for the task. Afterwards, the maximum used torques for the learned behaviors are recorded for each joint and set to identical values for the torque actuator. Muscle-actuators lead to more consistent performance and yield periodic hopping. Torque-actuators yield unstable policies that manage to jump very high once, but terminate the episode due to falls.

E.2 Additional goals for point-reaching with perturbations

We show ten random arm goals for precise point-reaching with perturbations that were not present during training in Fig. 4.

E.3 Additional hyperparameter variations

We show the relative performance of all runs of the hyperparameter searches in polar coordinates for precise point-reaching and hopping for both actuator morphologies (Fig. 5 and Fig. 6). The angles mark the specified hyperparameter (see Suppl. B.3.1 for definitions), while the radius marks the chosen value in $\log_{10}$-coordinates. The top row marks performance with muscle-actuators, the middle row with torque-actuators and the bottom row shows histograms of returns for both morphologies at different iterations. For point-reaching, muscle morphology leads to a return distribution that is centered around the top-performing parameter sets, with almost no badly performing sets left at iteration 7. In contrast, for torque-morphology a large number of runs is still distributed at low return values. For hopping, a much harder task, muscle-morphology quickly leads to a large number of runs at the top-performance level, while some badly performing parameter sets remain even at iteration 5. For torque-morphology, a large peak can be observed for returns close to 0, as most sampled parameter sets do not achieve any kind of hopping. Only at iteration 7, a few singular well-performing runs appear, that strongly outperform even the best muscle-driven run. This was to be expected, as any muscle-actuator behavior can in principle be replicated by torque actuators, given that the policy is able to learn it. Muscle actuators, on the other hand, are restricted to trajectory-dependent output.

E.4 Additional actuator models

Similar to Peng et al. [27], we present more actuator models that are widely used in robotics. We consider the ideal torque actuator to be neutral in its properties—only executing exactly what it was told. In contrast, a PD-controller [28] embeds additional knowledge about position control elements and error propagation dynamics. For the RL experiments, we use an identical PD formulation to Peng et al. [27]:

$$u(t) = k_p \left(\hat{q}(t) - q(t)\right) + k_d \left(\hat{\dot{q}} - \dot{q}\right), \tag{26}$$

with the joint angles q, the joint velocities $\dot{q}$, the desired position $\hat{q}$ and the desired velocity $\hat{\dot{q}}$. We also set $\hat{\dot{q}} = 0$, similar to [27]. We tuned the PD-controller by hand to achieve good step-wise trajectory tracking, see Fig. 7. We also ensured that it remains stable for faster position changes.

As a second additional actuator, we implemented a low-pass filtered torque actuator. The control signal is filtered according to the simplified muscle activation dynamics in the MuJoCo muscle model

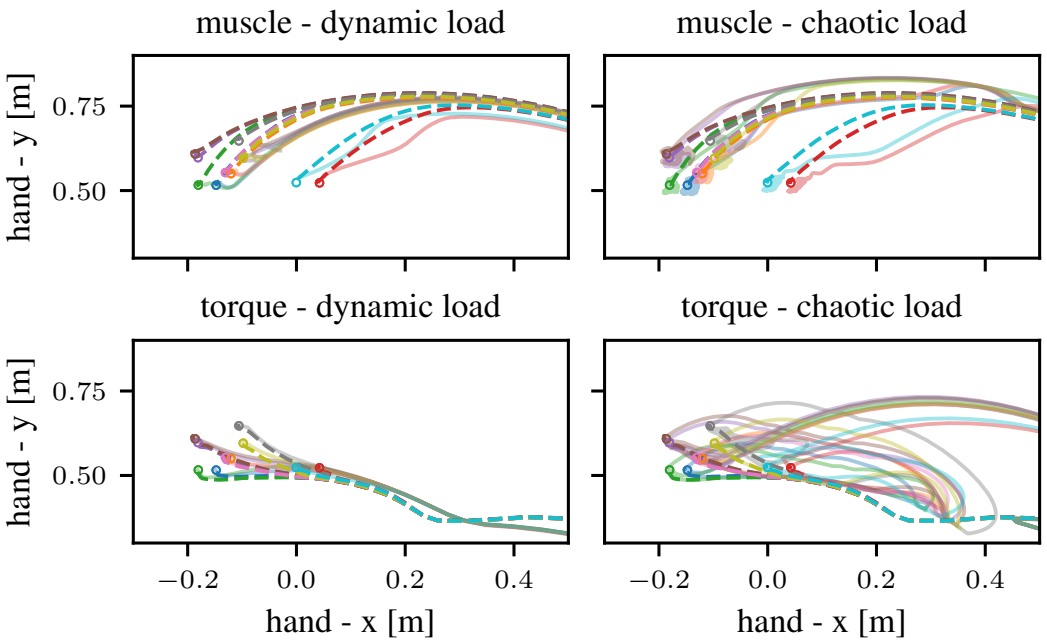

Figure 4: **Trajectories for dynamic (1.5 kg weight) and chaotic (attached ball) load.** Left: The torque actuator handles the dynamic load case slightly better than the muscle actuator for all goals, especially compared to the goals in grey, brown and purple. Right: The muscle-actuator performs very well for all chaotic load goals, except for a small deviation from the end-point. The torque actuator exhibits strong instabilities. The respective goal positions are marked as circles, the unperturbed baseline for each goal is shown with a dashed line, the perturbed trajectories with slightly transparent solid lines.

Eq. 4, which effectively act as a low-pass filter:

$$\dot{a}(t) = \frac{1}{\Delta t}(u(t) - a(t)), \tag{27}$$

which gets approximated in practice as:

$$a_{t+1} = a_t + \frac{\Delta t_{\text{sim}}}{\Delta t}(u(t) - a(t)). \tag{28}$$

The variable $a(t)$ denotes the effective action that is applied to the underlying torque actuator, $u(t)$ is the control signal, Δt is the time scale of the low-pass filter and Δt_{sim} is the time step of the *physics simulation*, which is not to be confused with the *control time step*: $\Delta t_{\text{control}} = 2\,\Delta t_{\text{sim}}$ for the MuJoCo simulations. All actuator properties such as the muscle dynamics, the low-pass filter and the PD controller are updated with the physics simulation time step, while the RL policy computes new actions only with the control frequency.

We repeated the precise point-reaching task with ArmMuJoCo with muscle actuation, torque actuation, PD actuation and two low-pass filter variants. The fast variant uses the time scale $\Delta t = 0.01$, which is the same as used in the muscle model and reacts very fast to new control signals. The slow variant uses $\Delta t = 1$ and produces a much stronger filtering effect. The results in Fig. 8 show that the muscle actuator outperforms all other variants.

Individual runs are shown in the right column in order to obtain an accurate picture of the variance across seeds for all actuators. Even when not considering the badly performing outliers, torque actuation seems to present larger variance than muscle actuation. The PD-controller performs worse than pure torque control for this task, which validates results by [29]: They found PD-controllers to perform worse than torque control when learning behaviors from scratch as opposed to tracking reference motions [27].

We noticed that, while the muscle only uses a maximum of ≈ 30 Nm during normal reaching, its properties allow it to intermittently use larger torques when perturbations are applied. We therefore

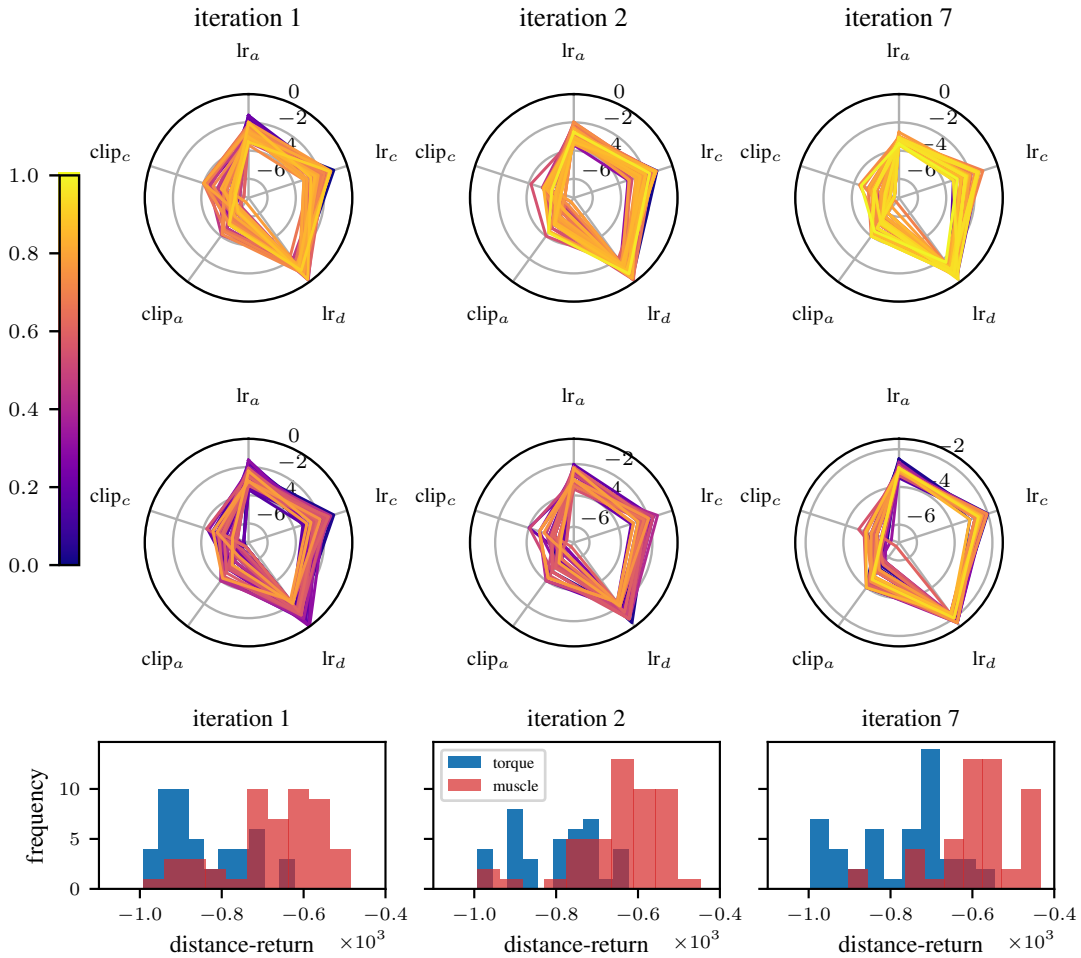

Figure 5: **Hyperparameter variation for precise point-reaching.** Hyperparameters are optimized following an iterative sampling scheme and individual runs train for 2×10^6 iterations. Fifty sets of parameters are sampled randomly from pre-determined distributions, the final performance is evaluated and used to adapt the sampling distributions for the next iteration. We record 7 iterations which equals 350 runs in total. We optimize 5 parameters related to MPO. The angle of the radarplot marks the parameter, the radius marks the value (in $\log_{10}$-coordinates). Top: Radarplot of parameters for the muscle in precise point-reaching at iteration 1, 2 and 7. The color marks the achieved performance of the parameter sample relative to the best achieved performance over **all** sampled parameters. Middle: precise point-reaching torque. Bottom: Histogram of returns for all sets of parameters at iterations 1, 2 and 7.

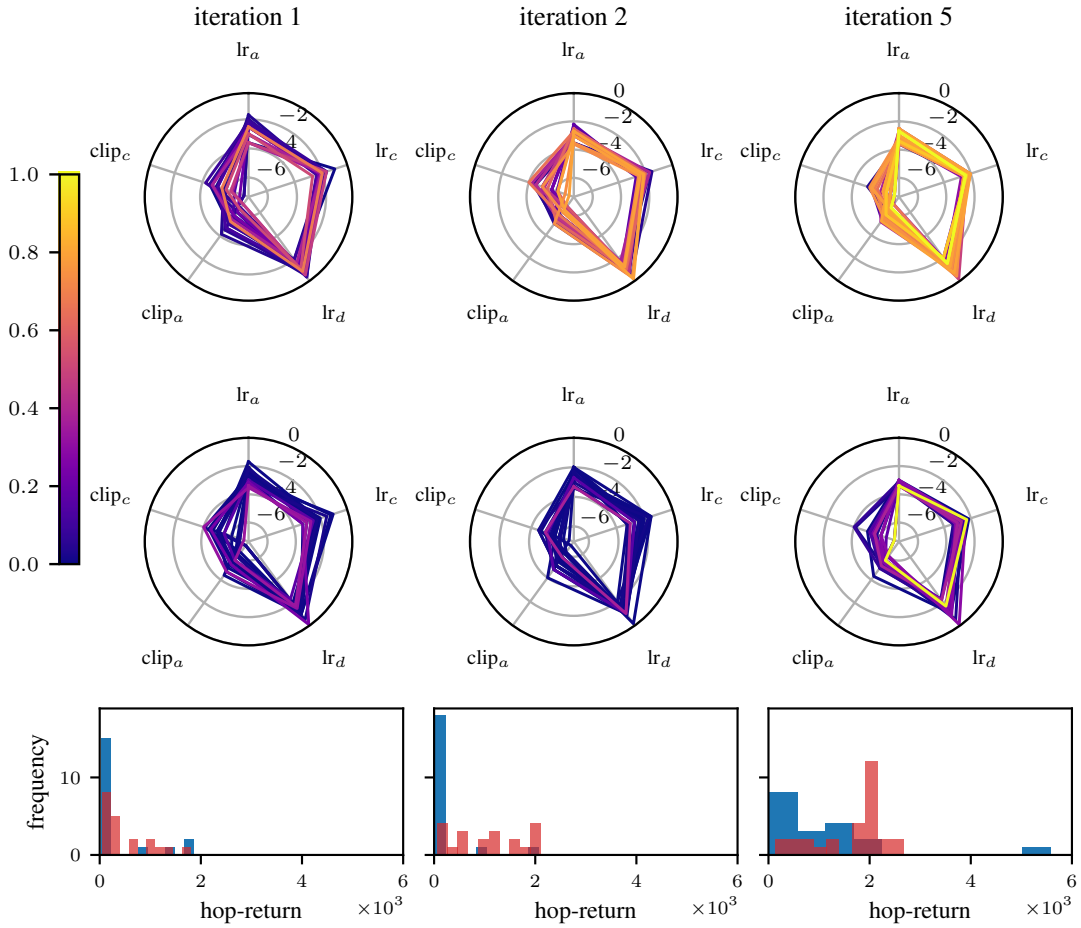

Figure 6: **Hyperparameter variation for hopping.** Hyperparameters are optimized following an iterative sampling scheme and individual runs train for 5×10^6 iterations. Twenty sets of parameters are sampled randomly from pre-determined distributions, the final performance is evaluated and used to adapt the sampling distributions for the next iteration. We record 5 iterations which equals 100 runs in total. We optimize 5 parameters related to MPO. The angle of the radarplot marks the parameter, the radius marks the value (in $\log_{10}$-coordinates). Top: Radarplot of parameters for the muscle in the hopping task at iteration 1, 2 and 5. The color marks the achieved performance of the parameter sample relative to the best achieved performance over **all** sampled parameters. Middle: hopping torque. Bottom: Histogram of returns for all sets of parameters at iterations 1, 2 and 5.

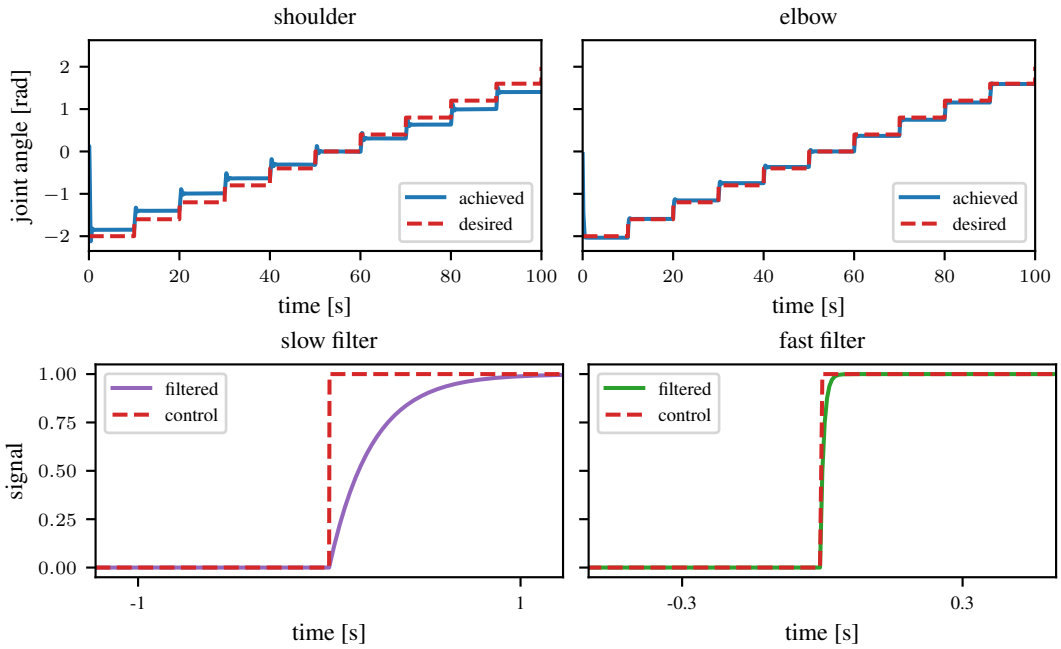

Figure 7: Top row: We tuned a PD-controller for ArmMuJoCo that is then used as an intermediate control layer for an RL agent. We tuned the parameters by hand to achieve good joint angle control over the workspace, shown in the figure for both joint angles. The slight mismatch in the shoulder joint (left) is due to gravitational forces, which are not counteracted in the controller design. An RL agent easily learns to compensate for this shift. Bottom row: We show two low-pass filtered torque actuators for an exemplary step-signal. The fast-filter uses the same parameters as the activation dynamics in the MuJoCo muscle model.

conduct a second series of experiments where we adjust the maximum allowed torque for all torque actuators to the intermittent upper limit of the muscle, which is $\tau_{\text{max}} = 60$ Nm. New learning curves were recorded for ArmMuJoCo point-reaching and are shown in Fig. 9. Generally, the performance for the non-muscular actuators decreases with larger torque limits. Only the PD-controller seems to exhibit smaller variance than in the small torque limit case.

E.5 Additional robustness experiments

In this section, we present evaluation of the robustness of the learned policies with a wide variety of masses and additional actuators. The results are reported for two different maximum torque limits for the non-muscle-based actuators, following the reasoning of Sec. E.4.

The variations are investigated for policies trained for point-reaching with ArmMuJoCo. All weights are added as a chaotic load that is attached with a rope. The results can be seen in Fig. 10 and Fig. 11 for $\tau_{\text{max}} = 30$ Nm and $\tau_{\text{max}} = 60$ Nm respectively. We use masses varying from 1 to 4 kg in the high force case, while they are halved in the other case. Even though the muscle actuator is the most stable across all variations, the pure torque actuator variant performs quite well when large forces are allowed. However, large torque limits also diminish the learning performance, as seen previously in Fig. 9. The results suggest a trade-off between learning speed and robustness for the torque controller, while the muscle actuator is able to leverage low forces during learning and automatically reacts to perturbations with stronger forces. The PD-controller only outperforms raw torque control for the large torque limit $\tau_{\text{max}} = 60$ Nm and a comparatively small perturbation mass of 1 kg, see Fig. 11 (second row, middle).

E.6 MuJoCo simulation time step ablation

To assess the influence of simulation accuracy on the obtained results, we record additional muscle and torque actuator learning curves with a much smaller simulation time step of $\Delta t_{\text{sim}} = 0.001$ instead of $\Delta t_{\text{sim}} = 0.005$. We additionally increase the frameskip of the simulation to achieve an equal control time step of $\Delta t_{\text{control}} = 0.01$ in both cases. The results are shown in Fig. 12. With

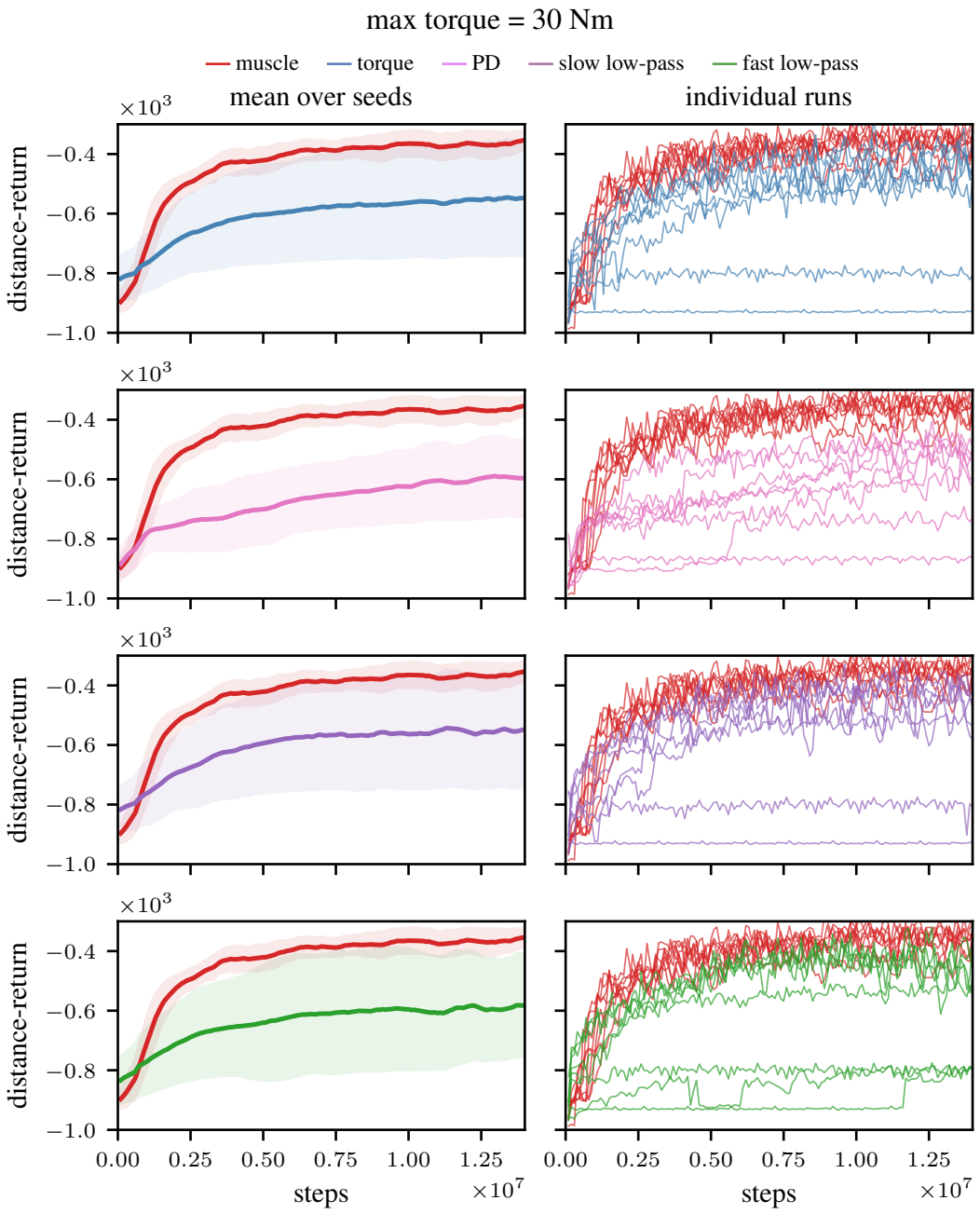

Figure 8: **The muscle actuator outperforms all other considered actuator designs.** We compare the learning curves for muscle, torque, PD and two low-pass filter actuators in the precise point-reaching task for ArmMuJoCo. Averages across random seeds and standard deviation are shown in the left column, individual runs in the right column. The torque actuator and the low-pass filter variants perform quite well, but their variance across seeds is larger than for the muscle, even when outliers are not considered. The PD-controller seems to exhibit less variance than a pure torque-driven approach, but the overall performance is worse. We recorded 8 random seeds for each actuator.

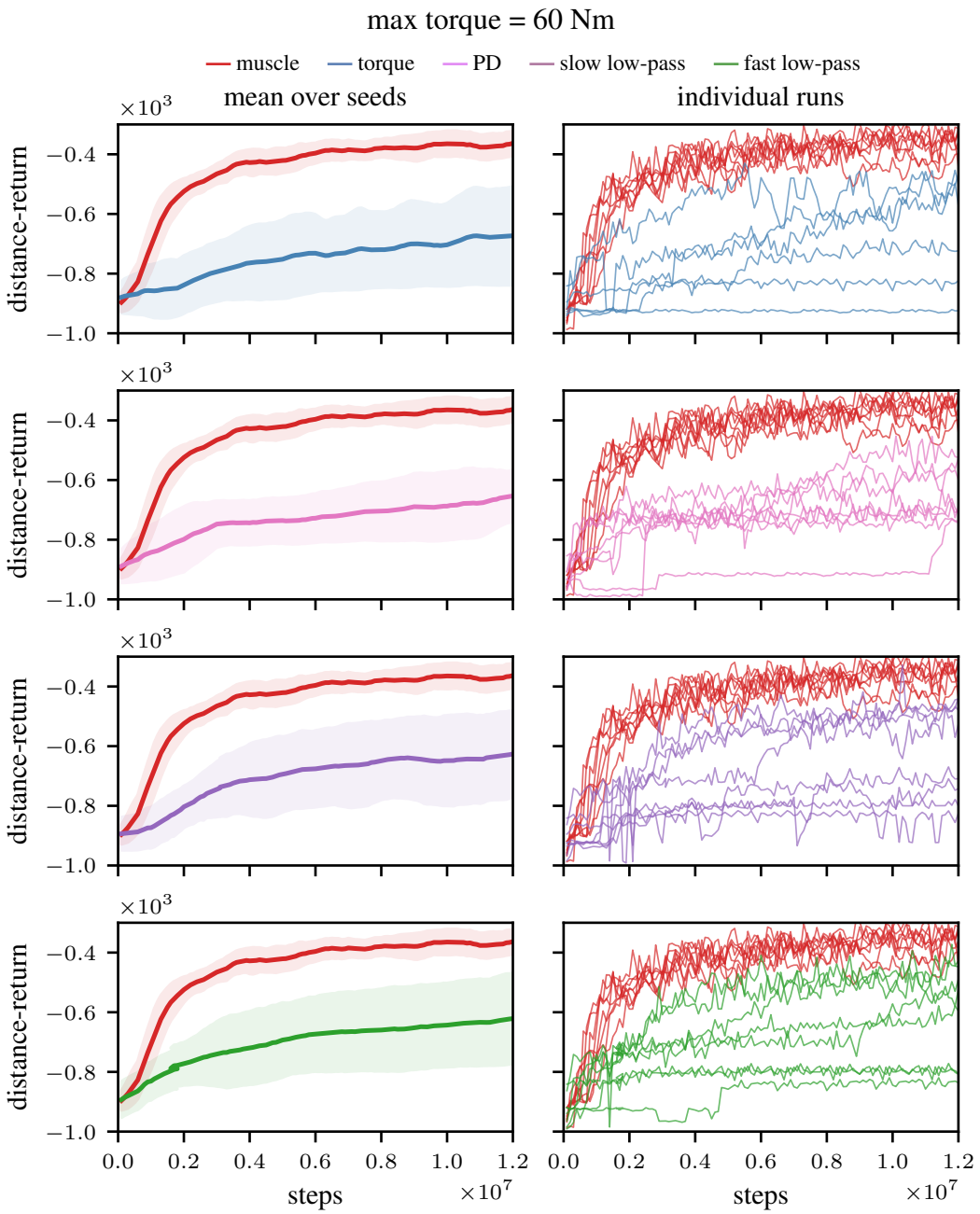

Figure 9: **Torque actuators perform worse when the maximum allowed force is increased.** We repeat the experiment in Fig. 9, but allow the torque actuator to use a maximum torque of $\tau_{\max} = 60$ Nm. This value is the maximum torque that the muscle actuator can output in perturbation experiments, even though it is not reached during point-reaching under normal conditions. While singular runs still perform well for the torque actuator variants and the PD controller achieves even less variance across seeds than before, the overall performance suffers when increasing the maximum force. We recorded 8 random seeds for each actuator.

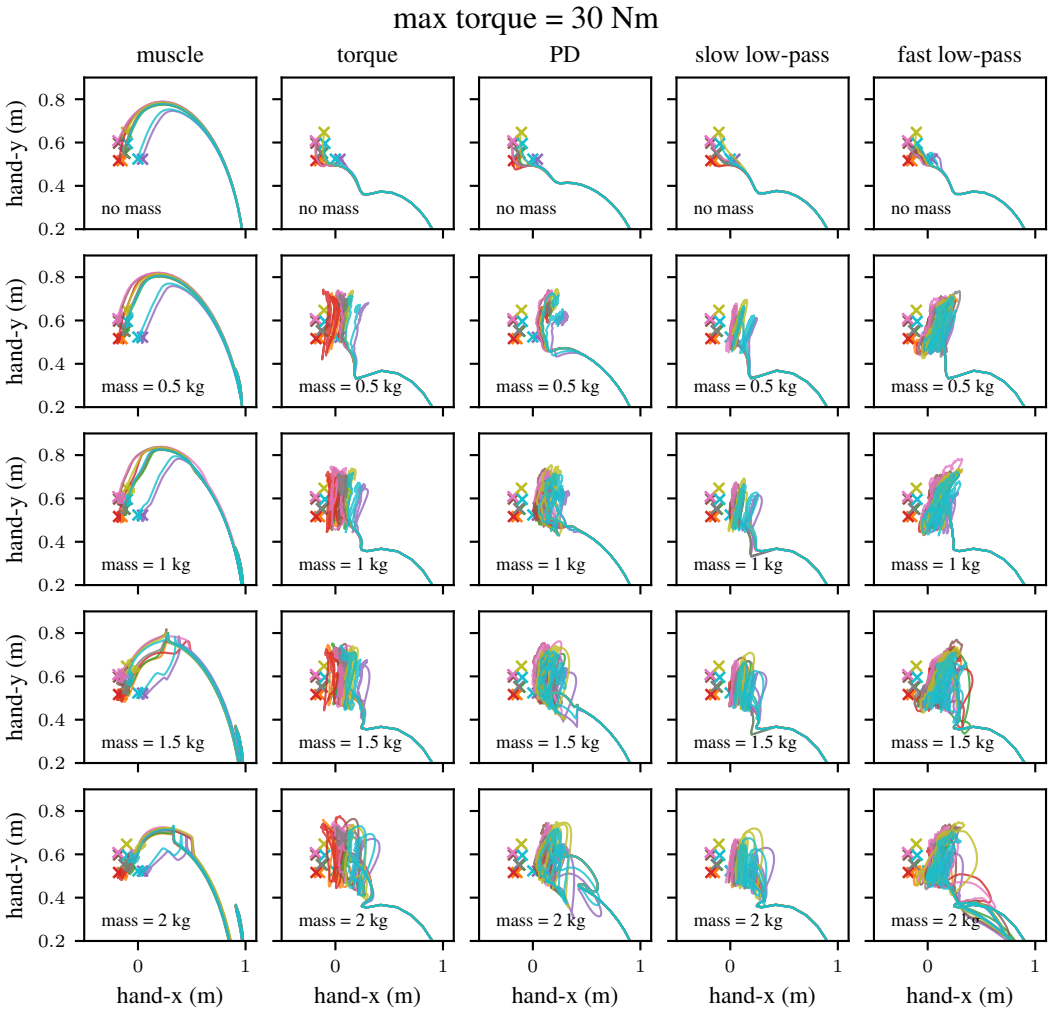

Figure 10: **The muscle actuator is more robust for all considered masses than the alternative.** We conducted perturbation experiments for all actuator models during which chaotic loads of differing masses were attached to the robot which were not present during training. The muscle actuator performs well up to 1.5 kg, when deviations start to get bigger. It does not reliably reach the goal for $m = 2$ kg. The torque actuator exhibits strong lateral oscillations for all masses and slight undershooting of the goal position. The PD-controller oscillates less for small masses, but undershoots the goals by a larger amount, as it was not tuned for this scenario. The low-pass filtered actuators perform similar to the pure torque case. For each experiment we used the best performing policy of each learning curve in Fig. 8 at the end of training. Ten goals were randomly chosen and used for all experiments.

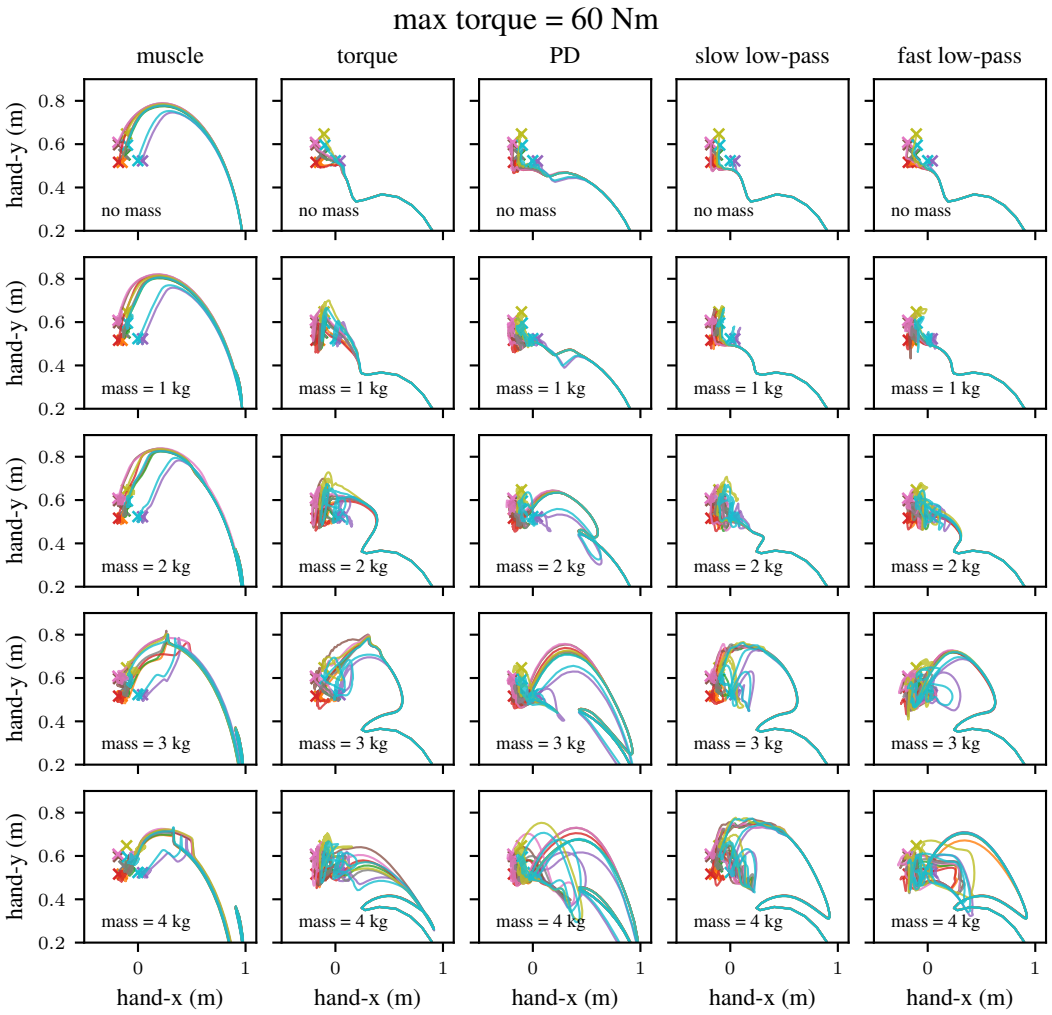

Figure 11: **Torque control is more robust with larger torque limits, but is still outperformed by muscles.** We repeat the experiment in Fig. 10 with a larger maximum torque limit of 60 Nm. All torque-variants seem to perform better than in the low torque limit case. For $m = 1$ kg, the PD-controller and the low-pass filter versions slightly outperform the pure torque actuator. Nevertheless, the muscle reacts more robustly for all considered masses. For each experiment we used the best performing policy of each learning curve in Fig. 9 at the end of training. The same ten goals as in Fig. 10 were used.

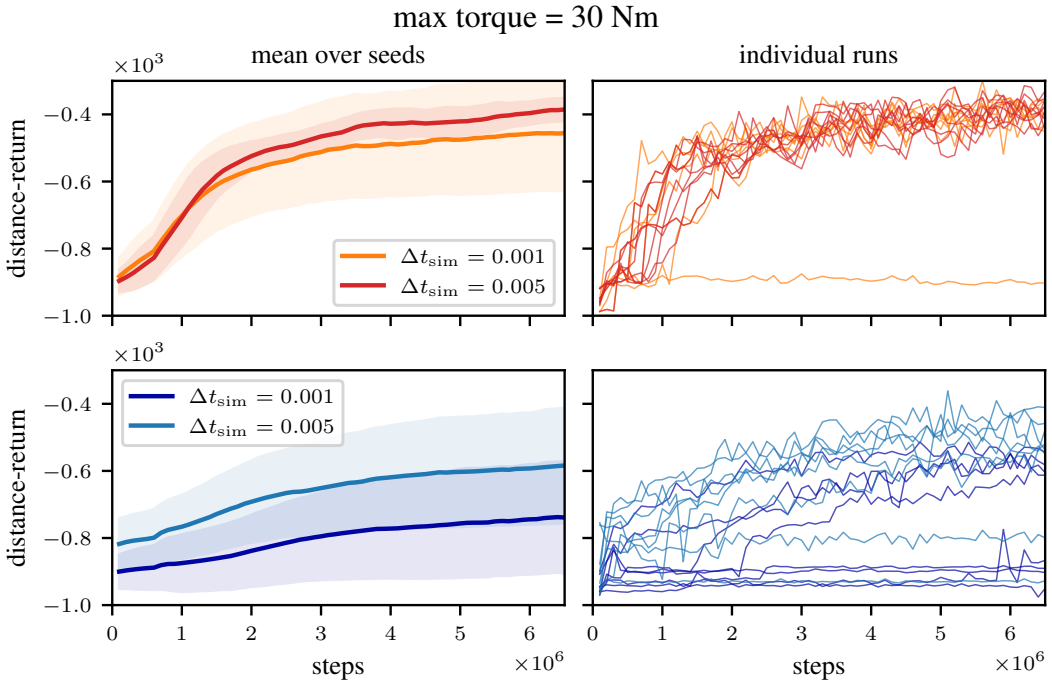

Figure 12: We present physical simulation time step ablations for ArmMuJoCo and precise point-reaching. While the Δt_{sim} was varied, the frameskip parameter was adjusted to achieve the same control time step $\Delta t_{\text{control}} = 0.01$ in all cases. Top: With the exception of a single unlucky run, the simulation time step does not seem to affect the performance of the muscle actuator in this task. Bottom: Surprisingly, the torque actuator performs much worse with a *smaller* simulation time step. As simulation accuracy increases with a smaller time step, we do not suspect this to be the result of numerical instability. A setting of $\Delta t_{\text{sim}} = 0.005$ was used for the all other MuJoCo experiments. We recorded 8 random seeds for each actuator.

the exception of one unlucky run, the muscle actuator performance seems to be unchanged under the more accurate simulation setting. The torque actuator, in contrast, seems to perform worse. As simulation accuracy increases with a smaller time step, we do not suspect this to be the result of numerical instability. As this only reinforces prior results, we conclude that the improved muscle actuator data-efficiency is not a result of numerical instability.

F Additional experiments (OC/MPC)

F.1 Nonlinearity in muscle model

In our study, we conclude that the nonlinear muscle properties can be beneficial for learning in terms of data-efficiency and robustness. To show-case the influence of individual properties, we performed additional smooth point-reaching and squatting experiments. The four major properties that differ between the torque actuator morphology and the muscle actuator morphology are the nonlinear activation dynamics, the nonlinear force-length, the nonlinear force-velocity relation and the nonlinear lever arms (see also Fig. 1 in the main paper). We switched each of these properties separately off to test which nonlinear muscle property contributes the most to the beneficial behavior. The results can be seen in Fig. 13. As shown in this figure, switching off the nonlinear force-velocity relation (no Fv) has the strongest impact and leads to results that are even worse than the torque actuator optimization. Additionally, the nonlinear activation dynamics (no actdyn) has some influence on the performance of the data-efficiency results. With these results, we would like to give a first indication that indeed the non-linearity of different muscle properties are beneficial for the data-efficiency in learning anthropomorphic tasks.

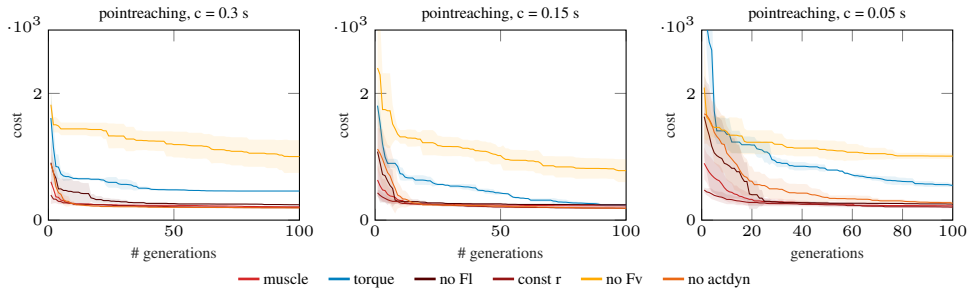

Figure 13: **Cost value for point-reaching while switching off different muscle properties.** Plotting the mean and standard deviation (shaded area) for 5 repeated runs for the two main actuator morphologies (muscle in red, torque in blue). Additionally, different morphologies are tested where muscle properties are switched off separately: We switched off the force-length relation (no Fl), set moment arms to be constant (const r), switched off the force-velocity (no Fv) and excluded the activation dynamics (no actdyn).

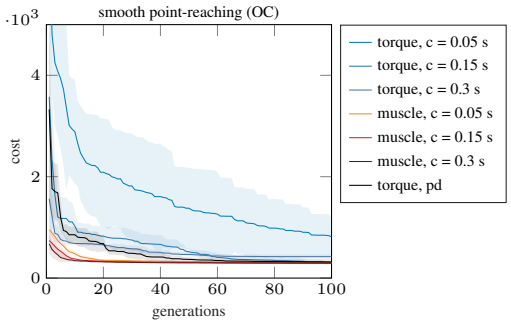

Figure 14: **Cost value for smooth point-reaching with additional baseline using PD control for torque actuator.** Plotting the mean and standard deviation (shaded area) for 5 repeated runs for the three main actuator morphologies (muscle in red, torque in blue, torque with PD controller in black).

F.2 Proportional-derivative torque control for learning

In the results presented in the main paper, we mainly compared the muscle actuator morphology to an idealized torque actuator without embedding any additional knowledge, e.g. position control which is typically used with a PD controller. Nevertheless, we consider the comparison with the PD control action space as a valuable baseline comparison. Therefore, we performed additional experiments for the smooth point-reaching task, where we added this additional baseline using PD control on top of the torque actuator morphology. Similar to the RL experiments (E.4), we use an identical PD formulation to Peng et al. [27]:

$$u(t) = k_p \left(\hat{q}(t) - q(t)\right) + k_d \left(\hat{\dot{q}} - \dot{q}\right), \tag{29}$$

with the joint angles q, the joint velocities $\dot{q}$, the desired position $\hat{q}$ and the desired velocity $\hat{\dot{q}}$. We also set $\hat{\dot{q}} = 0$, similar to [27] and our original cost function for smooth point-reaching (Eq. 18). In contrast to the RL experiments where we directly learn the desired angles for the control signal $u(t)$ for PD controller, here, with OC, we instead learn the k_p and k_d parameters. We allow for changes in these parameters every $c = 0.3$ s, whereas the control signal was updated continuously. Fig. 14 shows that the data-efficiency is slightly improved using a PD controller for the torque-actuated case in the smooth point-reaching task but it does not reach the performance of the muscle actuator.

F.3 Additional robustness experiments

In this section, we present additional robustness experiments for perturbing the arm model in the point-reaching task while adding unknown weights to the lower arm. In contrast to the main paper, we do not only show the perturbation using 1 kg, but varied the unknown weight up until 5 kg in 1 kg steps. The resulting angle trajectories are shown in Fig. 15. We see that both actuators are able to counteract unknown perturbation weights with 1 kg. For larger weights, the perturbations result in

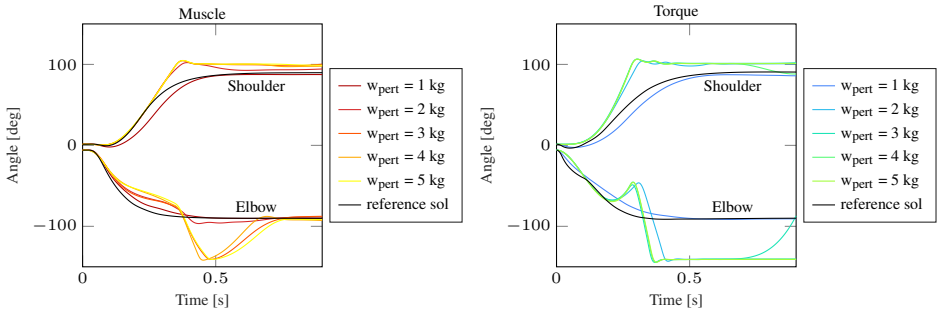

Figure 15: **Muscle morphology is more robust towards unknown weight perturbations.** Plotting the angle trajectories of the shoulder and elbow angle over time for the two actuator morphologies (left: muscle, right: torque) while varying the unknown weight (between 1 and 5 kg in 1 kg steps).

overshoots in the elbow joint angle which can be corrected in the muscle-actuated case, whereas the torque actuator struggles to counteract these perturbations. Summed up, the muscle morphology is more robust towards perturbations for a wide range of different unknown weights.

References

[1] D. Kistemaker, A. J. Van Soest, and M. F. Bobbert. Is equilibrium point control feasible for fast goal-directed single-joint movements? *Journal of Neurophysiology*, 95(5):2898–912, may 2006. ISSN 0022-3077. doi:10.1152/jn.00983.2005.

[2] H. Geyer and H. Herr. A muscle-reflex model that encodes principles of legged mechanics produces human walking dynamics and muscle activities. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 18(3):263–73, jun 2010. ISSN 1558-0210. doi:10.1109/TNSRE.2010.2047592.

[3] A. Bayer, S. Schmitt, M. Günther, and D. F. B. Haeufle. The influence of biophysical muscle properties on simulating fast human arm movements. *Computer Methods in Biomechanics and Biomedical Engineering*, 20(8):803–821, 2017. ISSN 1476-8259. doi:10.1080/10255842.2017.1293663.

[4] E. Todorov, T. Erez, and Y. Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, Oct. 2012. doi:10.1109/IROS.2012.6386109.

[5] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D. S. Seljebotn, and K. Smith. Cython: The best of both worlds. *Computing in Science & Engineering*, 13(2):31–39, 2011.

[6] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym, 2016.

[7] S. Schmitt. demoa-base: A Biophysics Simulator for Muscle-driven Motion, 2022. URL https://doi.org/10.18419/darus-2550.

[8] H. Hatze. A myocybernetic control model of skeletal muscle. *Biological cybernetics*, 25(2): 103–119, 1977.

[9] R. Rockenfeller, M. Günther, S. Schmitt, and . Götz, Thomas. Comparative sensitivity analysis of muscle activation dynamics. *Computational and Mathematical Methods in Medicine*, 2015: 1–16, 2015. doi:10.1155/2015/585409.

[10] R. Rockenfeller and M. Günther. Inter-filament spacing mediates calcium binding to troponin: A simple geometric-mechanistic model explains the shift of force-length maxima with muscle activation. *Journal of Theoretical Biology*, 454:240–252, 2018. ISSN 10958541. doi:10.1016/j.jtbi.2018.06.009.

[11] I. Wochner and S. Schmitt. arm26: A Human Arm Model, 2022. URL https://doi.org/10.18419/darus-2871.

[12] J. R. Walter, I. Wochner, M. Jacob, K. Stollenmaier, P. Lerge, and S. Schmitt. allmin: A Reduced Human All-Body Model, 2022. URL https://doi.org/10.18419/darus-2982.

[13] D. F. B. Haeufle, M. Günther, A. Bayer, and S. Schmitt. Hill-type muscle model with serial damping and eccentric force–velocity relation. *Journal of biomechanics*, 47(6):1531–1536, 2014.

[14] M. Hammer, M. Günther, D. F. B. Haeufle, and S. Schmitt. Tailoring anatomical muscle paths: a sheath-like solution for muscle routing in musculoskeletal computer models. *Mathematical Biosciences*, 311:68–81, 2019.

[15] N. Hansen, S. D. Müller, and P. Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary computation*, 11(1):1–18, 2003.

[16] M. M. Wierzbicka, A. W. Wiegner, and B. T. Shahani. Role of agonist and antagonist muscles in fast arm movements in man. *Experimental Brain Research*, 63(2):331–340, 1986.

[17] D. A. Kistemaker, A. K. J. Van Soest, and M. F. Bobbert. Is equilibrium point control feasible for fast goal-directed single-joint movements? *Journal of Neurophysiology*, 95(5):2898–2912, 2006.

[18] M. J. Powell. The bobyqa algorithm for bound constrained optimization without derivatives. *Cambridge NA Report NA2009/06, University of Cambridge, Cambridge*, 26, 2009.

[19] A. Abdolmaleki, J. T. Springenberg, Y. Tassa, R. Munos, N. Heess, and M. Riedmiller. Maximum a posteriori policy optimisation. In *International Conference on Learning Representations*,

2018. URL https://arxiv.org/abs/1806.06920.

[20] F. Pardo. Tonic: A deep reinforcement learning library for fast prototyping and benchmarking. *arXiv preprint arXiv:2011.07537*, 2020.

[21] P. Schumacher, D. Häufle, D. Büchler, S. Schmitt, and G. Martius. Dep-rl: Embodied exploration for reinforcement learning in overactuated and musculoskeletal systems, 2022. URL https://arxiv.org/abs/2206.00484.

[22] Ł. Kidziński, S. P. Mohanty, C. F. Ong, Z. Huang, S. Zhou, A. Pechenko, A. Stelmaszczyk, P. Jarosik, M. Pavlov, S. Kolesnikov, et al. Learning to Run challenge solutions: Adapting reinforcement learning methods for neuromusculoskeletal environments. In *The NIPS'17 Competition: Building Intelligent Systems*, 2018. URL http://arxiv.org/abs/1804.00361.

[23] D. M. Jessop and M. T. Pain. Maximum velocities in flexion and extension actions for sport. *Journal of human kinetics*, 50(1):37–44, 2016.

[24] D. F. Haeufle, I. Wochner, D. Holzmüller, D. Driess, M. Günther, and S. Schmitt. Muscles reduce neuronal information load: quantification of control effort in biological vs. robotic pointing and walking. *Frontiers in Robotics and AI*, 7:77, 2020.

[25] J. R. Walter, M. Günther, D. F. Haeufle, and S. Schmitt. A geometry-and muscle-based control architecture for synthesising biological movement. *Biological Cybernetics*, 115(1):7–37, 2021.

[26] M. G. Pandy, F. E. Zajac, E. Sim, and W. S. Levine. An optimal control model for maximum-height human jumping. *Journal of Biomechanics*, 23(12):1185–1198, 1990.

[27] X. B. Peng and M. van de Panne. Learning locomotion skills using deeprl: Does the choice of action space matter? In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 1–13, 2017.

[28] G. F. Franklin, J. D. Powell, and A. Emami-Naeini. *Feedback Control of Dynamic Systems (8th Edition) (What's New in Engineering)*. Pearson, 2018. ISBN 0134685717. URL https://www.amazon.com/Feedback-Control-Dynamic-Systems-Engineering/dp/0134685717?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0134685717.

[29] D. Reda, T. Tao, and M. van de Panne. Learning to locomote: Understanding how environment design matters for deep reinforcement learning. In *Motion, Interaction and Games*, MIG '20, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450381710. doi: 10.1145/3424636.3426907. URL https://doi.org/10.1145/3424636.3426907.