2024 IEEE International Conference on Robotics and Automation (ICRA)
May 13-17, 2024. Yokohama, Japan

Reinforcement Learning for Reduced-order Models of Legged Robots

Yu-Ming Chen[1], Hien Bui[1] and Michael Posa[1]

2024 IEEE International Conference on Robotics and Automation (ICRA) | 979-8-3503-8457-4/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICRA57147.2024.10610747

Abstract— **Model-based approaches for planning and control for bipedal locomotion have a long history of success. It can provide stability and safety guarantees while being effective in accomplishing many locomotion tasks. Model-free reinforcement learning, on the other hand, has gained much popularity in recent years due to computational advancements. It can achieve high performance in specific tasks, but it lacks physical interpretability and flexibility in re-purposing the policy for a different set of tasks. For instance, we can initially train a neural network (NN) policy using velocity commands as inputs. However, to handle new task commands like desired hand or footstep locations at a desired walking velocity, we must retrain a new NN policy. In this work, we attempt to bridge the gap between these two bodies of work on a bipedal platform. We formulate a model-based reinforcement learning problem to learn a reduced-order model (ROM) within a model predictive control (MPC). Results show a 49% improvement in viable task region size and a 21% reduction in motor torque cost. All videos and code are available at https://sites.google.com/view/ymchen/research/rl-for-roms.**

I. Introduction

Model-based planning and control have shown significant success for many years in navigating legged robots [3]–[8]. For example, Boston Dynamics's bipedal robots performed parkour and natural walking [5], [9], while SRI's humanoid DURUS achieved high energy efficiency in walking [10], [11]. Moreover, having robot models makes it possible to analyze the system stability and provide stability or safety guarantees via techniques such as capturability [12], sums of squares [13], hybrid zero dynamics [14] or control barrier functions [15]–[17]. However, as these methods can be computationally costly, reduced-order models (ROMs) are frequently deployed to achieve real time planning and control, albeit at the cost of reduced performance [8], [18].

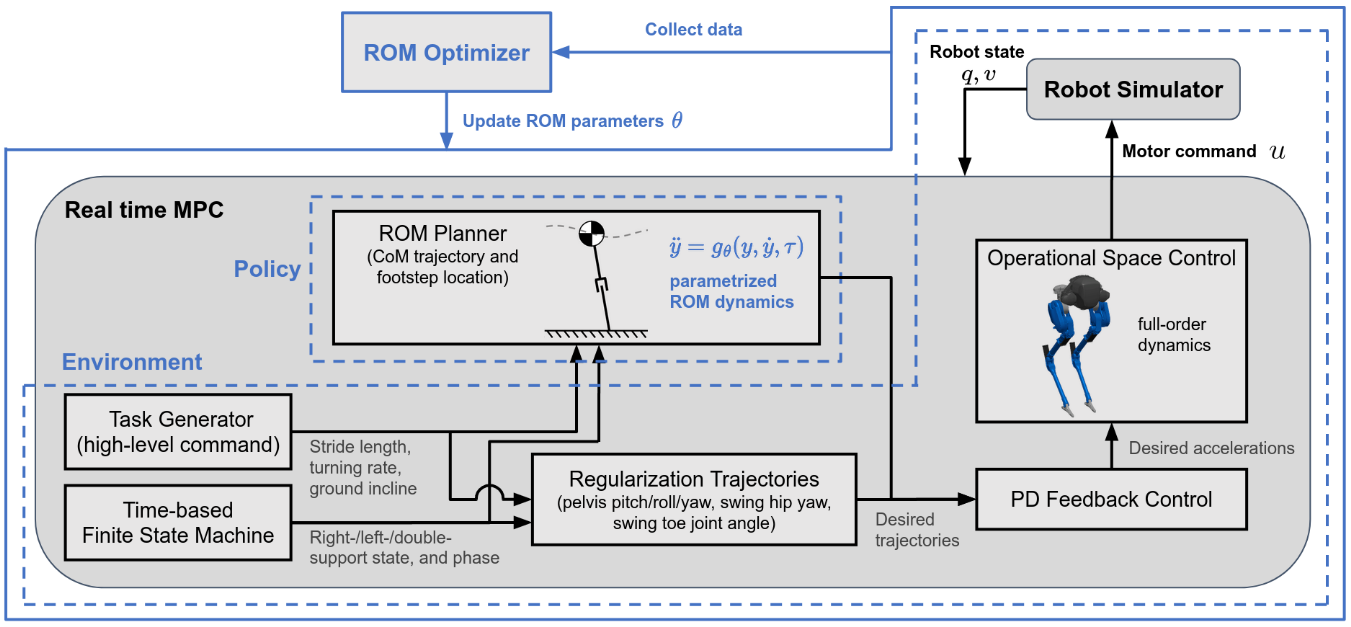
On the other hand, model-free reinforcement learning (RL) has emerged as a powerful tool for automatically synthesizing high-performance control policies [19]–[23]. Siekmann et al. highlighted the robustness achieved by a neural network (NN) policy in blind stair-walking [19], while Ma et al. developed a policy utilizing a robot's arm for fall damage mitigation and recovery [22]. These works demonstrate that neural networks, as universal approximators, enable robots to excel in specific tasks. However, model-free polices lack interpretability, rendering the existing model-based stability and safety techniques unsuitable. Additionally, the model-free methods struggle with generalizing policies to new task parameters without policy retraining, as task space parameterization is determined during the training phase. For example, the policy described in [22] effectively recovers a fallen robot but does not consider walking velocity commands. To enable the robot to walk, it is necessary to incorporate these velocity commands as additional inputs to the NN and undergo a fresh policy training process. Moreover, robots may encounter a wide array of potential tasks, such as footstep timing, adaptive limb adjustment during hardware failure, head movement for item detection, whole-body interactions between arms and legs, or collaborative furniture carrying with humans. Thus, it is impractical to enumerate all possible tasks to avoid future policy retraining.

This paper attempts to combine model-based planning and control with reinforcement learning to obtain the best of both worlds in the context of bipedal locomotion. It aims to retain the physical intepretabilty and the task flexibility of the model-based approach, while utilizing the RL capability in maximizing robot performance. Specifically, we use model predictive control (MPC) as the model-based control policy and learn a model within the MPC that maximizes the robot's performance via RL.

Within the MPC, ROMs are often used in order to achieve real time planning [6], [27], [28]. Classical approaches to reduced-order modeling often seek to find low-dimensional models which approximate some collected data (e.g. in approximating the solutions to fluid dynamics [29]). For a controlled system, such as a bipedal robot, the data is necessarily a product of the control policy, and thus the data and the ROM-induced controller are invariably intertwined. Nonetheless, recent work in locomotion has attempted this classical approach via model-based RL [30], or via robust control (treating the gap between a ROM and the full model as a disturbance [31]).

In contrast, we jointly optimize over the ROM and the induced control policy. Robot actuators are partially utilized to ensure that the system behaves like the reduced-order model, up to the limits of control performance. With this perspective, a good ROM is not one that matches some existing set of robot behaviors. Rather, we find task-optimal ROMs that maximize robot performance when deployed in conjuction with modern model-based planning and control architectures. Our prior work [18] optimizes controller-agnostic ROMs via trajectory optimization. However, the performance of the ROM may not seamlessly transfer to the robot due to the influence of specific heuristics or designs of the controller. In this paper, we improve our prior work by taking into account the control policy used on the robot while optimizing the ROM. This effectively closes the performance gap between offline model optimization and online model deployment.

[1]The authors are with the General Robotics, Automation, Sensing and Perception (GRASP) Laboratory, University of Pennsylvania, Philadelphia, PA 19104, USA {yminchen, xuanhien, posa}@seas.upenn.edu



ROM Optimizer
Collect data
Update ROM parameters θ
Robot state
q, v
Robot Simulator
Motor command u
Real time MPC
Policy
ROM Planner
(CoM trajectory and footstep location)
$\ddot{y} = g_\theta(y, \dot{y}, \tau)$
parametrized ROM dynamics
Operational Space Control
full-order dynamics
Environment
Task Generator (high-level command)
Stride length, turning rate, ground incline
Time-based Finite State Machine
Right-/left-/double-support state, and phase
Regularization Trajectories (pelvis pitch/roll/yaw, swing hip yaw, swing toe joint angle)
Desired trajectories
PD Feedback Control
Desired accelerations

Fig. 1. The diagram of our reinforcement learning framework for reduced-order models (ROMs) of legged locomotion. We learn a ROM in simulation where the robot is controlled via a real time model predictive control (MPC). The MPC follows a high-level command (task) and operates based on a time-based finite state machine (FSM), governing footstep timing in left-support, right-support, or double-support state. The MPC contains two trajectory generators – a reduced-order model planner and a regularization trajectory generator to fill out the joint redundancy of the robot. All desired trajectories are converted to desired acceleration command via PD feedback control before being sent to the Operational Space Controller (OSC), a quadratic-programming-based inverse-dynamics controller [1], [2]. In this learning framework, the ROM planner is the policy, while everything else in the closed-loop system, such as the simulation and OSC, constitutes the environment. We parameterize the policy using ROM parameters, specifically the parameters of ROM dynamics. The optimizer collects data from simulation rollouts and updates the model (policy) parameters according to a user-specified reward function. To ensure that the optimizer collects data at a consistent rate, we maintain a fixed update rate of 20Hz for the ROM planner, and we downsample the environment state (which operates at 1000Hz) to match this 20Hz rate.

A. Contributions

1) Formulate a reinforcement learning problem to directly optimize, with respect to a user-specified objective function, a reduced-order model given a controller.
2) Demonstrate that the RL algorithm improves the model performance by evaluating the model performance in detail, compared to both an existing model (LIP) and an optimal model from our prior work.
3) Demonstrate the flexibility of our model-based approach in task space parameterization in the post-training phase.

II. Background

A. Full-order Model of Cassie

Cassie [24] is a two-legged robot with closed-loop linkages and compliant components (leaf springs). It has line-feet, so there is one degree of freedom of underactuation when it stands on one foot. In the full dynamics, we model the linkages using (holonomic) distant constraints, and we account for the reflected inertia of the motors and also joint frictions. Let $q \in \mathbb{R}^{23}$ and $u \in \mathbb{R}^{10}$ be the configuration and input of the full-order model respectively. Let v be the velocity of the robot's configuration, and $x = [q, v]$ be the state of the robot.

B. ROMs for Legged Locomotion

Let y and τ be the configuration and input of the reduced-order model of a robot respectively. The reduced-order model can be defined by two functions – an embedding function and the dynamics of the ROM

$$y = r(q), \tag{1a}$$
$$\ddot{y} = g(y, \dot{y}, \tau). \tag{1b}$$

Reduced-order models for legged locomotion commonly describe the center of mass (CoM) motion, while each model imposes different constraints on the robot to derive its own CoM dynamics [32], [33]. For example, the linear inverted pendulum model (LIP) [32], [34] assumes zero vertical CoM accelerations, and the spring-loaded inverted pendulum model (SLIP) [33] enforces a spring-mass dynamics. Inspired by this, we search for an optimal CoM dynamics g, while using the same embedding function r, the CoM position relative to the stance foot, as LIP and SLIP. In this case, $\dim(y) = 3$. Furthermore, we assume $\dim(\tau) = 0$ for simplicity, although we keep τ in Eq. (1), since ROMs can have inputs in general (e.g. angle torques or the center of pressure [32]).

C. Model Predictive Control

Model predictive control (MPC) is a control technique that plans for desired input trajectories by predicting the robot's state in a receding horizon fashion [6], [27], [28]. For legged robots [24]–[26], the full models are usually too high dimensional to allow real time planning, so common ROMs introduced in Section II-B are used instead. Our MPC for

Cassie is outlined in Fig. 1. It consists of a few components – a high-level command generator, a finite state machine, trajectory generators (including a ROM planner) and a low-level controller.

1) High-level Command Generator: At the beginning of the diagram is a high-level command generator (a task generator). It outputs the desired stride length, the desired turning rate and the perceived ground incline level.

2) Finite State Machine: The MPC follows a predefined step timing dictated by the finite state machine. This determines the ground contact sequence inside the ROM planner and the contact mode inside the Operational Space Control.

3) Trajectory Generator: The primary trajectory generator creates a plan for the desired CoM state and desired footstep location given a high-level command. Specifically, the ROM planner solves an reduced-order trajectory optimization problem which minimizes an objective function while satisfying a set of constraints. The objective function predominantly minimizes the tracking error for the high-level commands. The constraints account for the initial state of the ROM, the dynamics of the ROM, and the input and state constraints. In this case, the inputs are the footstep locations, so the input constraints take care of the feasible stepping location. For more details of the planner, we refer to Section IV of [18]. Besides the planner, we also need to specify desired trajectories for the remaining degree of freedom of the robot. We generate these regularization trajectories by simple heuristics such as maintaining a horizontal attitude of the pelvis body, having the swing foot parallel to the contact surface, and aligning the hip yaw angle to the desired heading angle.

4) Low-level Controller: All desired trajectories (functions of time) from the trajectory generators are passed into PD feedback controllers where they are converted into desired accelerations. Finally, Operational Space Control is an inverse dynamics controller formulated as a quadratic programming problem [1], [2]. It takes into account the full model dynamics in the optimization problem and outputs an optimal input u that minimizes the tracking error of the desired accelerations.

III. Problem Statement

Our goal is to find the optimal model parameters that maximize the performance of the robot given a task distribution Γ. In this study, Γ is a uniform distribution over a range of stride length, turning rate and ground incline. Let $u(\theta)$ be a model-based control policy (i.e. a controller) that guides the robot to follow the trajectories of a ROM parameterized by θ. From another viewpoint, this same policy $u(\theta)$ also constrains the robot to behave like the ROM. Let $\{(x_t, u_t) \mid t = 1, 2, ..., T\}$ be the state and input trajectories of the robot completing a task $\gamma \sim \Gamma$ under the policy $u(\theta)$. We evaluate the performance for this task γ and policy $u(\theta)$ using a cost function $h(x, u)$ accumulated over the trajectories:

$$H_{\gamma,u(\theta)} = \sum_{t=1}^{T} h\left(x_t, u_t\right). \tag{2}$$

Given this evaluation metric, an optimization problem for finding the model can be formulated as

$$\min_{\theta} \mathbb{E}_{\gamma\sim\Gamma} \min_{u(\theta)} \left[H_{\gamma,u(\theta)}\right], \tag{3}$$

where the inner-level optimization minimizes the cost $H_{\gamma,u(\theta)}$ over all possible controllers $u(\theta)$ that constrain the robot to behave like a ROM parameterized by θ, and the outer-level optimization minimizes the expectation of inner-level cost over a task distribution. Eq. (3) has been proven to be solvable by our prior work [18] with successful results. However, the optimal policy $u(\theta)$ of (3) is not computable online in real time, necessitating an alternative policy $u_o(\theta)$ for online model deployment. Thus, while solving (3) leads to a ROM capable of maximal performance, this performance is not necessarily realizable via real time control, leading to reduced closed-loop performance. To improve this, in this paper, we find the model parameters θ while using the control policy $u_o(\theta)$ during offline training:

$$\min_{\theta} \mathbb{E}_{\gamma\sim\Gamma} \left[H_{\gamma,u_o(\theta)}\right]. \tag{4}$$

Specifically, this control policy $u_o(\theta)$ is the model predictive control presented in Section II-C. This seemingly simple change necessitates a significant shift in algorithmic approach, which we detail in Section IV.

IV. ROM Learning via RL

In this section, we cast the model optimization problem in Eq. (4) as a model-based reinforcement learning problem.

A. Reinforcement Learning Structure

In reinforcement learning, the system comprises a policy and an environment. The policy takes the current environment state s and outputs an action a. Given the current state s and the action a, the environment transitions to the next state s'. Each pair of state and action (s, a) results in a reward r. In this work, the policy is the ROM planner, and the environment includes everything else in the closed-loop system (Fig. 1). Given this choice, the state of the environment s includes the robot's state and input (x, u), task γ, and finite state machine information (including the phase of the current state). The policy action a are the CoM states and the foot step locations (the solution of the ROM trajectory optimization). Additionally, we limit the policy's update rate to 20Hz, so that the state and action pairs are collected at a fixed rate.

B. Policy Parameterization

We parameterize the ROM with monomials of the state of the ROM, with linear weights. That is,

$$\ddot{y} = g_\theta(y, \dot{y}, \tau) = \Theta\phi(y, \dot{y}), \tag{5}$$

where g_θ is the ROM dynamics parameterized by θ, $\Theta \in \mathbb{R}^{n_y \times n_\phi}$ is the matrix form of θ, and ϕ is the feature vector containing the monomials. Even though we use monomials here, any function approximator (e.g. a neural network) can be used to parameterize the ROM dynamics in Eq. (5). We note that the ROM parameters θ are the policy parameters, as the ROM planner is the policy in our RL framework (Fig. 1).

In this paper, we initialize the ROM to an LIP, because the LIP is effective for simple walking tasks, and this initialization speeds up the learning process. To implement this initialization, we augment the feature vector ϕ with the terms in the LIP dynamics function. Additionally, we use monomials of order up to 2. That is, ϕ includes elements such as y_1, y_0^2 and $y_0 \dot{y}_1$, where the subscripts denote the indices of each element in y. Given this, θ is of dimension 90. The parameterization choice in Eq. (5) preserves physical interpretability, since the learned model describes the CoM dynamics.

C. Rewards and the RL Problem Statement

Instead of minimizing the cost in Eq. (4), our RL formulation maximizes the return (i.e. accumulated rewards) $R = \sum_{t=1}^{T} r_t$, where r_t is the reward at time t. Therefore, to encourage minimizing the cost $h(x,u)$ and achieving a desired task γ, we design the reward function

$$r = \exp(-w \cdot h) + 0.5 \, \exp\left(-\|\gamma - \gamma_{fb}\|_W\right), \qquad (6)$$

where w and W are constant weights, γ is the desired task value, and γ_{fb} is the achieved task value (a function of the robot's state). We note that this reward is a function of the environment state s. Furthermore, the accumulated reward structure already incentivizes the robot to finish the entire episode, eliminating the need for penalty terms in case of early simulation termination (e.g. the robot falls). In this paper, we choose $h = u^T u$ (quadratic cost on motor torques), and the horizon $T = 100$ which is equivalent to 5 seconds of simulation time. Additionally, the user-specified h is ultimately the metric for model performance evaluation.

The objective of our RL problem is to maximize the expected return over the task distribution:

$$\max_{\theta} \, \mathbb{E}_{\gamma \sim \Gamma}\left[R\right]. \qquad (7)$$

There are several algorithms for solving Eq. (7), such as policy gradient and evolutionary strategy. In this work[2], we choose Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) [35]. The advantage of this approach includes easy parallelization [36] and easy hyperparameter tuning (partially due to the absence of value approximation). We use the package Optuna [37] for the CMA-ES optimizer.

Let p_θ be the probability distribution over θ in the CMA-ES algorithm. The exact problem that CMA-ES solves is

$$\max_{p_\theta} \, \mathbb{E}_{\theta \sim p_\theta}\left[\mathbb{E}_{\gamma \sim \Gamma}\left[R\right]\right]. \qquad (8)$$

The difference between Eq. (8) and (4), besides the cost-reward difference, is the stochasticity of parameters θ needed for the exploration of CMA-ES.

[2]Proximal Policy Gradient (PPO) also resulted in successful model training in our experiment, but it required more hyperparameter tuning.

Algorithm 1 Evaluation of return $\hat{R}\left(\theta, \{\gamma_j\}\right)$

```
Input: model parameters $\theta$ and sampled tasks $\{\gamma_j\}$
 1: for $j = 1, \ldots, N_\gamma$ do
 2:     Roll out an episode with the MPC in Section II-C
 3:     Compute the return $R_{\gamma_j} = \sum_{t=1}^{T} r_t$ of this rollout
 4: end for
 5: return $\frac{1}{N_\gamma} \sum_{j=1}^{N_\gamma} R_{\gamma_j}$
```

Algorithm 2 CMA-ES with curriculum learning

```
Input: initial mean $\theta_0$ and variance $\sigma_0^2$ for the parameter
    distribution $p_\theta$, and initial task set $\Gamma_d$
 1: for iter = $1, 2, \ldots$ do
 2:     if mod(iter, $N_c$) = 0 then
 3:         Grow the task set $\Gamma_d$ (Section IV-D)
 4:     end if
 5:     Randomly draw $N_\gamma$ tasks $\{\gamma_j\}$ from $\Gamma_d$
 6:     Sample $N_\theta$ parameters $\{\theta_i\} \sim p_\theta$
 7:     for each sampled $\theta_i$ do
 8:         Compute return $\hat{R}\left(\theta_i, \{\gamma_j\}\right)$
 9:     end for
10:     Update the mean and covariance of $p_\theta$ (CMA-ES)
11: end for
```

D. Curriculum Learning

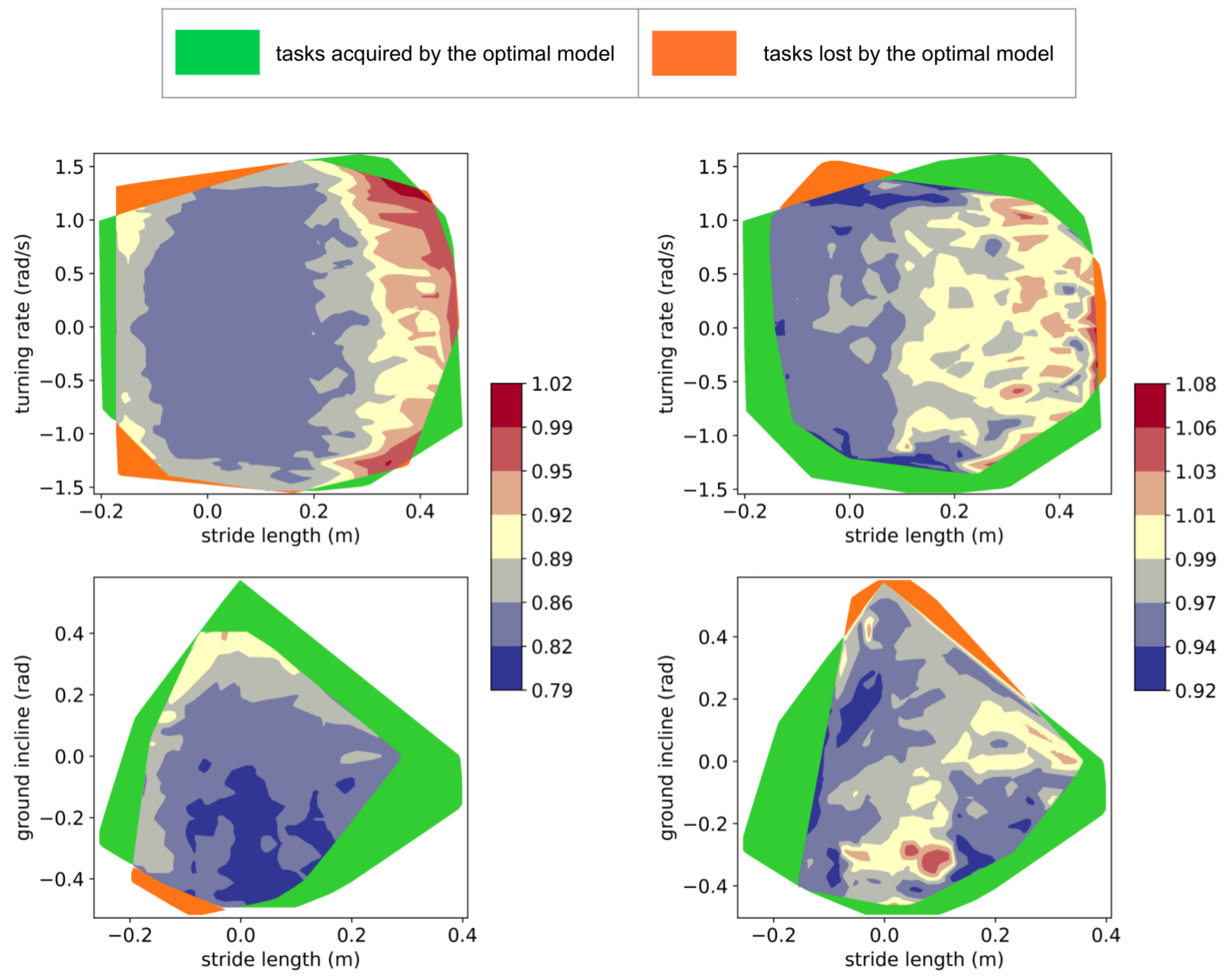
We observe that learning a policy for a large task space is difficult, so we implement a curriculum learning similar to [38] to facilitate the learning process. Our method discretizes the continuous task domain of Γ into a set Γ_d. This set Γ_d starts small and expands every N_c iterations by including the adjacent tasks of successful tasks during learning.

To evaluate the inner expected value in Eq. (8), we approximate it by randomly drawing N_γ number of tasks from the set Γ_d and averaging the returns of these tasks. Let $\hat{R}$ be this approximate expected return given model parameters θ. The evaluation of $\hat{R}$ is shown in Algorithm 1. As a part of the curriculum learning, we also adjust the number of tasks N_γ, as the size of Γ_d grows larger. Specifically, $N_\gamma = \text{floor}\left(\rho_\gamma |\Gamma_d|\right)$, where ρ_γ is the sampling percentage and $|\cdot|$ counts the number of elements in a set.

The algorithm for solving Eq. (8) with curriculum learning is outlined in Algorithm 2. In every iteration, CMA-ES samples a few model parameters θ's according to p_θ, and evaluates the value (return $\hat{R}$ in this case) for each sampled θ. Given these values, CMA-ES updates the mean and the covariance matrix of the parameter distribution p_θ.

V. Simulation Experiment

We learn a ROM in simulation using Drake [39] offline, and deploy it to the same simulation environment for detailed evaluation comparison between the initial model and the optimal model. We also compare this optimal model to the one derived using the prior approach [18]. Lastly, we showcase the flexibility in task space reparameterization in Section V-D.



(a) Comparison against LIP (initial model)

(b) Comparison against the optimal model of prior work's approach

Fig. 2. Cost landscape comparisons. Fig. 2a compares the optimal model to the initial model. Fig. 2b compares the optimal models between this paper and the prior work. For each model comparison, we create landscapes in two sets of experiments – one involving turning rate and stride length (with a constant ground incline of 0 rad), and the other involving ground incline and stride length (with a constant turning rate of 0 rad/s). Each plot shows the ratio of the optimal model's cost to the compared model's cost. The color scheme red-to-blue illustrates the degree to which the optimal ROM shows improvement. Ratio below 1 means the optimal ROM performs better than the compared model, and vice versa. Green color shows the task regions gained by the optimal model, and orange color shows the task regions lost by the optimal model.

A. *Hyperparameters*

The hyperparameters for the RL are shown in Table I. The number of parameters sampled per iteration N_θ is adaptively chosen by the CMA-ES optimizer according to the parameter dimension. The sample density ρ_γ is chosen to be 0.1 which has the benefit of speeding up the learning process compared to evaluating all discretized task samples (i.e. $\rho_\gamma = 1$). The initial standard deviation σ_0 of the parameters is set to a relatively small number, since the initial ROM already works for simple tasks. Moreover, we observe that using a larger standard deviation σ_0 does not show better performance in our experiments, and it has a drawback of potentially diverging from the optimal parameters from time to time. For the curriculum learning, we set the initial task space to be straight-line walking with stride lengths between -0.1 m and 0.2 m. The task space is discretized by 0.1 m, 0.1 rad and 0.45 rad/s in stride length, ground incline and turning rate, respectively. Lastly, we only expand the task space every 30 iterations to learn a model gradually. The entire model learning takes about 24 hours to reach convergence.

B. *Comparing the Optimal Model to LIP (Initial Model)*

We compare the performance of the initial model θ_0 and the optimal model from the learning result. To evaluate the model performance, we run the simulation for a wide range of stride lengths, ground inclines and turning rates. We then extract the periodic walking gaits according to the criteria shown in Table II which ensure variations over foot steps fall below specific thresholds. Given these periodic trajectories, we compute the cost $H_{\gamma,u(\theta)}$ and plot the landscape of the cost ratio of the optimal model to the initial model, shown in Fig. 2a. The color scheme red-to-blue illustrates the degree to which the ROM shows improvement, with red corresponding to a minimal improvement and blue to a 20% cost reduction. In addition to the ratio, we also visualize the task region where either the initial model or the optimal model fails to complete, visualized by green and orange. Green corresponds to the task regions that the optimal model gains, and orange corresponds to the regions that the optimal model loses.

In this example, the optimal ROM reduces the cost across almost the entire task space (up to 21% cost reduction). For flat ground walking tasks, the optimal model shows the largest improvement in the region of small stride lengths. For inclined walking, the optimal model achieves higher performance improvement in downhill tasks than uphill tasks. Additionally, the optimal model increases the task region size by 49% for inclined walking. For example, at an incline of -0.2 rad, the maximum stride length increases from 0.2 m to 0.37m.

C. Comparison against our Prior Work

We also conducted a set of experiments similar to Section V-B. Instead of comparing against the initial model, we compare the optimal model from Algorithm 2 to the optimal from the prior work [18]. The cost landscape comparison for this is shown in Fig. 2b.

Compared to the prior approach, the optimal model of this paper shows up to 28% increase in task space along with an average 2.7% cost improvement. For the flat ground walking tasks, the model yields a 22% larger viable task region than that of the prior approach. For the inclined walking tasks, it is 28% larger. We hypothesize that this improvement comes from the fact that in the RL framework we are able to use a Cassie model of high fidelity during training, while the prior approach was limited to a simplified Cassie model for ease of solving the inner-level optimization in Eq. (3).

We note that, besides the above numerical comparisons, the RL approach in this paper is easier to implement than the prior approach, since it does not require a careful implementation of offline trajectory optimization (inner-level optimization in Eq. (3)) in conjunction with the online control policy ($u_o(\theta)$ in Section III) for reducing the gap between the open-loop and closed-loop performances.

D. Generalization to New Task Space Parameterization

One advantage of using a model-based approach is the flexibility in specifying new tasks, as motivated in Section I. During the training stage of our experiment shown above, we train the model using common tasks including stride length, turning rate and ground incline. We demonstrate that the model can be easily extended to achieve unseen tasks by modifying the MPC diagram in Fig. 1. For example, we might want to ensure that a body-mounted sensor is oriented at a target of interest, or we might want a robot to collaboratively carry a table with a human, which requires the robot facing a different direction than the walking direction. To mimic these scenarios for Cassie, we turn Cassie's pelvis to the side while walking forward. We achieve this by simply changing the desired value of the pelvis yaw angle for the regularization trajectory generator in Fig. 1. In the accompanying video, we can see that the robot achieves this task without any more offline training. We note that the ability to adapt to the new task is a product of MPC, while the adaptability also depends on the ROM performance.

initial standard deviation σ_0	1e-3
number of sampled parameters per iteration N_θ	17
sample density ρ_γ	0.1
number of iterations for task space expansion N_c	30
stride length discretization	0.1 m
turning rate discretization	0.45 rad/s
ground incline discretization	0.1 rad

TABLE I
HYPERPARAMETERS FOR THE MODEL LEARNING

Stride length variation	< 2 cm
Side stepping variation	< 3 cm
Pelvis height variation	< 3 cm
Pelvis yaw variation	< 0.1 rad
Window size	4 consecutive footsteps

TABLE II
CRITERIA TO QUALIFY A PERIODIC WALKING GAIT

VI. CONCLUSION AND FUTURE WORK

We formulate a model-based reinforcement learning problem for reduced-order models of legged locomotion. This provides an avenue to bridge the gap between the well-established model-based control and the emerging field of reinforcement learning for legged robots, combining the performance-maximizing capability of RL with the physical interpretability and the task specification flexibility of model-based approaches. The experiments show that the optimal model reduces the torque cost by up to 21% and improves the viable task region size by up to 49% over the traditional models like LIP. We also compare this work to our prior work which uses full model trajectory optimization during the ROM optimization, and the results show an up to 28% improvement in the viable task region size along with a mild improvement in torque cost.

In this paper, we solve the RL problem using CMA-ES, but theoretically any reinforcement learning optimizer can be applied to our RL framework in Fig. 1, since our policy (ROM planner) is differentiable [40]. Future work includes exploring more optimizers to find one that results in the highest performance improvement. Additionally, we observed in our prior work [18] that parametrizing the embedding function r along with the ROM dynamics function g (i.e. both Eq. (1a) and (1b)) lead to higher performance improvement than parameterizing only g. In this paper, we simplified the problem and the RL formulation by limiting r to a CoM kinematic function, so one future direction is to parameterize both r and g, and learn the model in a similar pipeline.

VII. ACKNOWLEDGMENT

Toyota Research Institute provided funds to support this work.

References

[1] L. Sentis and O. Khatib, "Control of free-floating humanoid robots through task prioritization," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pp. 1718–1723, IEEE, 2005.

[2] P. M. Wensing and D. E. Orin, "Generation of dynamic humanoid behaviors through task-space control with conic optimization," in *2013 IEEE International Conference on Robotics and Automation*, pp. 3103–3109, IEEE, 2013.

[3] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake, "Optimization-based locomotion planning, estimation, and control design for Atlas," *Autonomous Robots*, vol. 40, no. 3, pp. 429–455, 2016.

[4] E. Ackerman, "Boston dynamics' cheetah robot now faster than fastest human," *IEEE Spectrum (Automation Blog).*, 2012.

[5] P. Marion and the team, "Flipping the script with atlas." https://blog.bostondynamics.com/flipping-the-script-with-atlas.

[6] G. Gibson, O. Dosunmu-Ogunbi, Y. Gong, and J. Grizzle, "Terrain-adaptive, alip-based bipedal locomotion controller via model predictive control and virtual constraints," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6724–6731, IEEE, 2022.

[7] X. Xiong and A. Ames, "3-d underactuated bipedal walking via h-lip based gait synthesis and stepping stabilization," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2405–2425, 2022.

[8] P. M. Wensing, M. Posa, Y. Hu, A. Escande, N. Mansard, and A. Del Prete, "Optimization-based control for dynamic legged robots," *arXiv preprint arXiv:2211.11644*, 2022.

[9] "Boston dynamics petman prototype." https://www.youtube.com/watch?v=67CUudkjEG4. Accessed: 2023-09-04.

[10] J. Reher, E. A. Cousineau, A. Hereid, C. M. Hubicki, and A. D. Ames, "Realizing dynamic and efficient bipedal locomotion on the humanoid robot durus," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1794–1801, IEEE, 2016.

[11] E. Ackerman, "Durus: Sri's ultra-efficient walking humanoid robot," *IEEE Spectrum (Automation Blog).*, 2015.

[12] T. Koolen, T. De Boer, J. Rebula, A. Goswami, and J. Pratt, "Capturability-based analysis and control of legged locomotion, part 1: Theory and application to three simple gait models," *The international journal of robotics research*, vol. 31, no. 9, pp. 1094–1113, 2012.

[13] M. A. Posa, T. Koolen, and R. L. Tedrake, "Balancing and step recovery capturability via sums-of-squares optimization," 2017.

[14] E. R. Westervelt, J. W. Grizzle, and D. E. Koditschek, "Hybrid zero dynamics of planar biped walkers," *IEEE transactions on automatic control*, vol. 48, no. 1, pp. 42–56, 2003.

[15] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European control conference (ECC)*, pp. 3420–3431, IEEE, 2019.

[16] Q. Nguyen, A. Hereid, J. W. Grizzle, A. D. Ames, and K. Sreenath, "3d dynamic walking on stepping stones with control barrier functions," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 827–834, IEEE, 2016.

[17] C. Khazoom, D. Gonzalez-Diaz, Y. Ding, and S. Kim, "Humanoid self-collision avoidance using whole-body control with control barrier functions," in *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*, pp. 558–565, IEEE, 2022.

[18] Y.-M. Chen, J. Hu, and M. Posa, "Beyond inverted pendulums: Task-optimal simple models of legged locomotion," *arXiv preprint arXiv:2301.02075*, 2023.

[19] J. Siekmann, K. Green, J. Warila, A. Fern, and J. Hurst, "Blind bipedal stair traversal via sim-to-real reinforcement learning," *arXiv preprint arXiv:2105.08328*, 2021.

[20] D. Crowley, J. Dao, H. Duan, K. Green, J. Hurst, and A. Fern, "Optimizing bipedal locomotion for the 100m dash with comparison to human running," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 12205–12211, IEEE, 2023.

[21] J. Dao, K. Green, H. Duan, A. Fern, and J. Hurst, "Sim-to-real learning for bipedal locomotion under unsensed dynamic loads," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 10449–10455, IEEE, 2022.

[22] Y. Ma, F. Farshidian, and M. Hutter, "Learning arm-assisted fall damage reduction and recovery for legged mobile manipulators," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 12149–12155, IEEE, 2023.

[23] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, no. 62, p. eabk2822, 2022.

[24] E. Ackerman, "Agility robotics introduces cassie, a dynamic and talented robot delivery ostrich," *IEEE Spectrum (Automation Blog).*, 2017.

[25] B. Robotics, "Nadia humanoid." https://www.ihmc.us/nadia-humanoid/. Accessed: 2022-09-05.

[26] Boston Dynamics, "Atlas." https://bostondynamics.com/atlas/. Accessed: 2023-08-27.

[27] M. Chignoli, D. Kim, E. Stanger-Jones, and S. Kim, "The mit humanoid robot: Design, motion planning, and control for acrobatic behaviors," in *2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids)*, pp. 1–8, IEEE, 2021.

[28] X. Xiong and A. D. Ames, "Dynamic and versatile humanoid walking via embedding 3d actuated slip model with hybrid lip based stepping," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6286–6293, 2020.

[29] B. Peherstorfer and K. Willcox, "Dynamic data-driven reduced-order models," *Computer Methods in Applied Mechanics and Engineering*, vol. 291, pp. 21–41, 2015.

[30] Y. Yang, K. Caluwaerts, A. Iscen, T. Zhang, J. Tan, and V. Sindhwani, "Data efficient reinforcement learning for legged robots," in *Conference on Robot Learning*, pp. 1–10, PMLR, 2020.

[31] A. Pandala, R. T. Fawcett, U. Rosolia, A. D. Ames, and K. A. Hamed, "Robust predictive control for quadrupedal locomotion: Learning to close the gap between reduced-and full-order models," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6622–6629, 2022.

[32] S. Kajita and K. Tani, "Study of dynamic biped locomotion on rugged terrain-derivation and application of the linear inverted pendulum mode," vol. 2, pp. 1405–1411, IEEE International Conference on Robotics and Automation (ICRA), 1991.

[33] R. Blickhan, "The spring-mass model for running and hopping," *Journal of biomechanics*, vol. 22, no. 11-12, pp. 1217–1227, 1989.

[34] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa, "The 3D linear inverted pendulum mode: a simple modeling for a biped walking pattern generation," pp. 239–246, IEEE International Conference on Intelligent Robots and Systems (IROS), 2001.

[35] N. Hansen, "The cma evolution strategy: a comparing review," *Towards a new evolutionary computation: Advances in the estimation of distribution algorithms*, pp. 75–102, 2006.

[36] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," *arXiv preprint arXiv:1703.03864*, 2017.

[37] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[38] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal, "Rapid locomotion via reinforcement learning," *arXiv preprint arXiv:2205.02824*, 2022.

[39] R. Tedrake, "Drake: Model-based design and verification for robotics," 2019.

[40] B. Amos, I. Jimenez, J. Sacks, B. Boots, and J. Z. Kolter, "Differentiable mpc for end-to-end planning and control," *Advances in neural information processing systems*, vol. 31, 2018.