A Survey of Optimal Reduced-Order Models for Bipedal Robots

Alvin Abduh

***Abstract*—The reviewed works focus on optimizing reduced-order models (ROMs), which are widely used for their simplicity in legged locomotion, particularly bipedal robots, enabling a wide array of tools for planning, control, and analysis while remaining low dimensional, albeit at the cost of limiting the full model and ultimately sacrificing performance. We review three proposed model optimization algorithms that automatically synthesize reduced-order models that remain low dimensional and retain capabilities of the full model. The first two approaches both solve a bilevel optimization problem. Where the inner-level is a trajectory optimization problem and the outer-level is a gradient descent problem. The first algorithm uses sequential programming, while the second algorithm uses the envelope theorem to solve the reduced-order model optimization problem. The third approach uses model-based reinforcement learning to synthesize a ROM. We observe their results on the bipedal robot Cassie, showing in simulation a significant improvements in joint torque costs of 23% and walking speed of 54% and a 49% improvement in viable task region size.**

I. Motivation

State-of-the-art approaches for planning and control of legged locomotion use model-based approaches or model-free Reinforcement Learning approaches. The field of model-based planning and control for robots can be categorized into two types: the use of a full-order model and the use of a reduced-order model. The full-order model, while capable of delivering high performance, presents challenges in formal analysis due to the numerous degrees of freedom inherent in legged robots. To mitigate this complexity, the legged robot community has widely adopted the use of reduced-order models. These reduced-order models typically impose constraints on the full model dynamics while encapsulating the task-relevant aspects of the full-order dynamics. For instance, the Linear Inverted Pendulum (LIP) model, which assumes the robot to be a point mass remaining within a plane, which significantly lowers energy efficiency and restricts the robot's speed and stride length. Similarly, the Spring Loaded Inverted Pendulum (SLIP) model, a point mass model with spring-mass dynamics, with zero centroidal angular momentum rate and zero ground impacts at the foot touchdown event. Consequently, planning motions solely within the reduced-space inevitably imposes limitations on the full dynamics, restricting a complex robot's motion to that of the low-dimensional model and inevitably compromising the robot's performance. The limitations of reduced-order models have been widely recognized within the community, leading to the development of numerous extensions. These universally rely on human intuition and often in the form of mechanical components such as springs, dampers, rigid bodies, additional legs, etc.

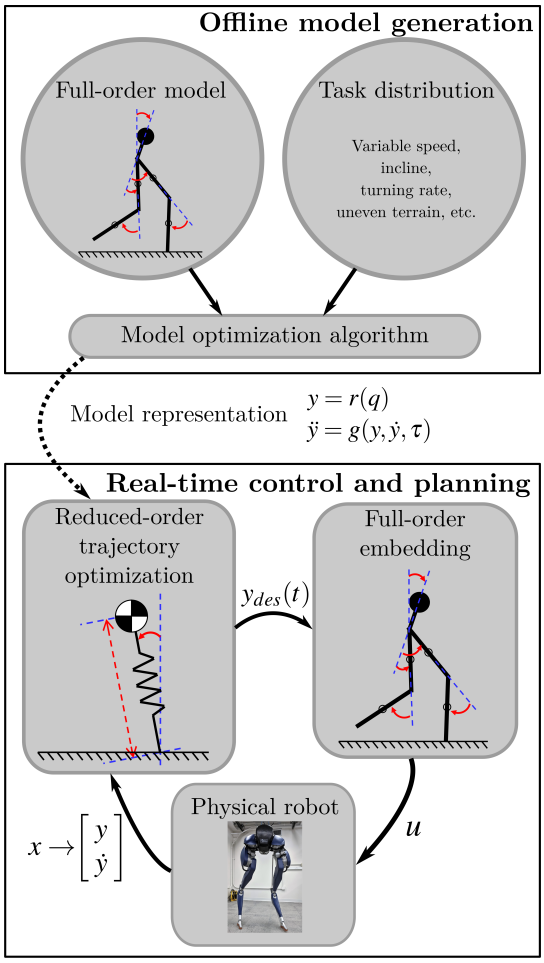

Fig. 1. Overview of optimal reduced-order model. Figure taken from [1]

While some extensions have enhanced robot performance, it remains unclear which extension offers more performance improvement than others, and a metric for improving model performance is lacking. Furthermore, it has been demonstrated that not all model extensions can significantly enhance the performance of robots. On the other hand, model-free Reinforcement Learning (RL) has emerged as a potent tool for the automatic synthesis of high-performance control policies. However, these model-free policies lack interpretability, making existing model-based stability and safety techniques unsuitable. Additionally, model-free methods struggle with generalizing policies to new task parameters without policy retraining, as task space parameterization is determined during

the training phase. Moreover, robots may encounter a wide array of potential tasks. Thus, it is impractical to enumerate all possible tasks to avoid future policy retraining. The goal is to bridge the gap between model-free and model-based planning and control with reinforcement learning to obtain the best of both worlds in the context of bipedal locomotion. This approach seeks to preserve the physical interpretability and task flexibility inherent in model-based approaches while harnessing the RL capability to maximize robot performance.

II. Problem Statement

The objective is to identify an optimal model, denoted as μ^*, given a distribution Γ over a set of tasks. A reduced-order model is defined as $\mu \triangleq (r, g)$ where q and u are the generalized position and input of the full order-model and y and τ are the generalized position and input of the reduced the order-model, here r is an embedding function $r : q \mapsto y$ and g is the reduced-order dynamics with $g(y, \dot{y}, \tau)$, with $y = r(q)$ and $\ddot{y} = g(y, \dot{y}, \tau)$. The aim is to find a reduced-order model μ^* that facilitates low-cost motion across the task space. This can be represented as:

$$\mu^* = \underset{\mu \in M}{\operatorname{argmin}} \, \mathbb{E}_\gamma[\mathcal{J}_\gamma(\mu)] \tag{1}$$

where M denotes the model space, $\mathbb{E}_\gamma$ represents the expected value over Γ, and $\mathcal{J}_\gamma(\mu)$ is the cost associated with accomplishing the tasks $\gamma \sim \Gamma$ when the robot is constrained to a specific model μ. This problem is infinite-dimensional over the space of embedding and dynamics functions r and g. The motivation for parameterizing the model is to transform the infinite-dimensional optimization problem into a finite-dimensional one, thereby making it computationally tractable. This approach allows efficient numerical optimization techniques to find the optimal model parameters, which would otherwise be infeasible with the original infinite-dimensional problem. To realize the optimization problem in (1), the most common solution would be to parameterize the embedding function r and dynamics function g, and solve the parameterized trajectory optimization problem. Assuming that the dynamics are affine in τ with a constant multiplier, r and g can be expressed as:

$$y = r(q; \theta_e) = \Theta_e \Phi_e(q), \tag{2a}$$

$$\ddot{y} = g(y, \dot{y}, \tau; \theta_d) = \Theta_d \Phi_d(y, \dot{y}) + B_y \tau, \tag{2b}$$

With the functions r and g being parameterized using basis functions $\{\Phi_{e,i} | i = 1, \ldots, n_e\}$ and $\{\Phi_{d,i} | i = 1, \ldots, n_d\}$ with linear weights $\theta_e \in \mathbb{R}^{n_y \cdot n_e}$ and $\theta_d \in \mathbb{R}^{n_y \cdot n_d}$ and where $\Theta_e \in \mathbb{R}^{n_y \times n_e}$ and $\Theta_d \in \mathbb{R}^{n_y \times n_d}$ are θ_e and θ_d arranged as matrices, $\Phi_e = [\Phi_{e_1}, \ldots, \Phi_{e,n_e}]$, $\Phi_d = [\Phi_{d_1}, \ldots, \Phi_{d,n_d}]$ and $B_y \in \mathbb{R}^{n_y \times n_\tau}$. For simplicity, B_y is assumed to be a constant value, but the method can be generalized by parameterizing $B_y(y, \dot{y})$. While a linear parametrization was chosen, any differentiable function approximator (e.g., a neural network) can be used. With the model parameters $\theta = [\theta_e, \theta_d] \in \mathbb{R}^{n_t}$, (1) can be rewritten as:

$$\begin{aligned} \theta^* = \; & \underset{\theta}{\operatorname{argmin}} \; \mathbb{E}_\gamma[\mathcal{J}_\gamma(\theta, w)] \\ & \text{subject to:} \\ & f_c(x_i, x_{i+1}, u_i, u_{i+1}, \lambda_i, \lambda_{i+1}, \delta_i, \alpha_i) = 0, \\ & i = 1, \ldots, n-1 \\ & g_c(x_i, u_i, \lambda_i, \tau_i; \theta) = 0, \quad i = 1, \ldots, n \\ & C_\gamma(x_i, u_i, \lambda_i) \leq 0, \quad i = 1, \ldots, n \end{aligned} \tag{3}$$

where $\mathcal{J}_\gamma(\theta, w)$ is the optimal cost of a trajectory optimization problem, defined as:

$$\mathcal{J}_\gamma(\theta, w) \triangleq \min_w \sum_{i=1}^{n-1} \frac{1}{2}(h_\gamma(x_i, u_i) + h_\gamma(x_{i+1}, u_{i+1}))\delta_i \tag{4}$$

Here, f_c are the dynamics constraints for the full-order dynamics, g_c are the dynamics constraints for the reduced-order dynamics, and w are the decision variables $w = [x_1, \ldots, x_n, u_1, \ldots, u_n, \lambda_1, \ldots, \lambda_n, \tau_1, \ldots, \tau_n, \alpha_1, \ldots, \alpha_{n-1}]$. The dynamics and holonomic constraints of the full-order model are given by, and the reduced-order constraint g_c is:

$$\begin{aligned} & g_c = \ddot{y}_i - g(y_i, \dot{y}_i, \tau_i; \theta_d) = 0 \\ \Rightarrow \; & g_c = J_i \dot{v}_i + \dot{J}_i v_i - g(y_i, \dot{y}_i, \tau_i; \theta_d) = 0 \end{aligned} \tag{5}$$

where $y_i = r(q_i; \theta_e)$, $\dot{y}_i = \frac{\partial r(q_i; \theta_e)}{\partial q_i} \dot{q}_i$, $J_i = \frac{\partial r(q_i; \theta_e)}{\partial q_i}$ and $\dot{v}_i = M(q_i)^{-1}(f_{cg}(q_i, v_i) + Bu_i + J_h(q_i)^T \lambda_i + \tau_{app}(q_i, v_i))$. For readability, (4) is rewritten as:

$$\begin{aligned} \mathcal{J}_\gamma(\theta, w) = \min_w \quad & \tilde{h}_\gamma(w) \\ \text{subject to} \quad & \tilde{f}_\gamma(w, \theta) \leq 0, \end{aligned} \tag{6}$$

where $\tilde{h}_\gamma$ is the cost function of (4) and $\tilde{f}_\gamma \leq 0$ encapsulates all the constraints in (3).

III. Challenges

As demonstrated in $\mathcal{J}_\gamma(\theta, w)$ which is the optimal cost for a trajectory optimization problem, we have two variables to optimize, the model parameter θ and the decision variables w. This makes (3) a bilevel optimization problem. Solving a bilevel optimization problem is generally NP-hard. This complexity arises from the hierarchical structure of the problem, where the solution to one level depends on the solution of another. This interdependence makes the problem non-convex and thus challenging to solve both theoretically and practically.

IV. Proposed Solutions

We study three possible ways to solve this bilevel optimization problem, the first proposed solution is to use gradient descent to solve the outer optimization of (3) with two different approaches of solving the inner optimization, the first using an approach in quadratic programming and the latter being the use of the envelope theorem to derive it analytically. The second approach is to use reinforcement learning to solve the bilevel optimization problem.

A. Using Approximated Quadratic Program

Starting from an initial parameter seed θ_0, N tasks are sampled. The cost for each task, denoted as $\mathcal{J}_\gamma(\theta, w)$, is evaluated by solving the corresponding trajectory optimization problem (3). To compute the gradient $\nabla_\theta[\mathcal{J}_{\gamma j}(\theta, w)]$, several approaches have been proposed. One such approach is based on sequential quadratic programming, where (4) is locally approximated with an equality-constrained quadratic program, considering only the active constraints. Let $\tilde{w}_\gamma = w - w_\gamma^*$ and $\tilde{\theta} = \theta - \theta^{(i)}$, where w_γ^* is the optimal solution of (4) and $\theta^{(i)}$ is the parameter at the i-th iteration. The approximated quadratic program is given by

$$\begin{aligned} \mathcal{J}_\gamma(\theta, w) \approx \min_{w_\gamma} \quad & \frac{1}{2}\tilde{w}_\gamma^T H_\gamma \tilde{w}_\gamma + b_\gamma^T \tilde{w}_\gamma + c_\gamma \\ \text{subject to} \quad & F_\gamma \tilde{w}_\gamma + G_\gamma \tilde{\theta} = 0 \end{aligned} \tag{7}$$

Applying the KKT conditions, we derive the following equation for the optimal solution

$$\begin{bmatrix} H_\gamma & F_\gamma^T \\ F_\gamma & 0 \end{bmatrix} \begin{bmatrix} \tilde{w}_\gamma^* \\ v_\gamma^* \end{bmatrix} = \begin{bmatrix} -b_\gamma \\ -G_\gamma \tilde{\theta} \end{bmatrix} \tag{8}$$

where v_γ^* is the optimal dual solution. Equation (8) can be further solved, with the solution rewritten as

$$\tilde{w}^* \gamma = Q_\gamma \tilde{\theta} + p_\gamma \tag{9}$$

for some $Q_\gamma \in \mathbb{R}^{n_w \times n_t}$ and $p_\gamma \in \mathbb{R}_w^n$. Since we approximate the original problem around w^* and $\theta^{(i)}$, we know that $\tilde{w}_\gamma^* = 0$ if $\tilde{\theta} = 0$; therefore, $p_\gamma = 0$. Substituting (9) into (7) and taking the gradient, we derive

$$\nabla_\theta[\mathcal{J}_{\gamma j}(\theta, w)]|_{\theta=\theta^{(i)}} = Q_{\gamma_j}^T b_{\gamma_j} \tag{10}$$

B. Using the Envelope Theorem

The second approach uses the Envelope Theorem to directly derive the analytical gradient.

Proposition 1 (Differentiability Condition). *Assume $\tilde{h}$ and $\tilde{f}$ are continuously differentiable functions, and consider an optimization problem*

$$\tilde{J}(\theta) = \min_w \tilde{h}(w, \theta) \quad \text{s.t.} \quad \tilde{f}(w, \theta) \leq 0, \tag{11}$$

where $\tilde{J}(\theta)$ is the optimal cost of the problem. Let $w^*(\theta)$ be the optimal solution to (11)*. w^* is differentiable with respect to θ if the following conditions hold:*

1) *the second-order optimality condition for* (11)*,*
2) *linear independence constraint qualification (**LICQ**), and*
3) *strict complementarity at w^*.*

Theorem 1 (Envelope Theorem). *Assume the problem in Eq. (12) satisfies the differentiability condition. The gradient of the optimal cost $\tilde{J}(\theta)$ with respect to θ is*

$$\nabla_\theta[\tilde{J}(\theta, w)] = \frac{\partial \tilde{h}(w^*, \theta)}{\partial \theta} + \lambda^* \frac{\partial \tilde{f}(w^*, \theta)}{\partial \theta}, \tag{12}$$

where λ^* is the dual solution to (12)*.*

Corollary 1. *The gradient of the optimal cost of* (4) *is*

$$\nabla_\theta[J_\gamma(\theta)] = \lambda^* \frac{\partial \tilde{f}_\gamma(w^*, \theta)}{\partial \theta}, \tag{13}$$

where w^* and λ^* are respectively the primal and the dual solution to (4)*.*

Proof. The proof follows directly from Theorem 1. Note that the cost function in (4) is independent of θ, in which case the first term of (12) becomes 0. □

Algorithm 1 Reduced-order model optimization using approximated quadratic program

Require: Task distribution Γ
Ensure: θ^*
{Model initialization}
$\theta \leftarrow \theta_0$
{Model optimization}
repeat
  Randomly sample tasks $\gamma_j \sim \Gamma for j = i, \ldots, N$.
  for $j = 1, \ldots, N$ **do**
    Solve (4) to get $\mathcal{J}_{\gamma_j}(\theta, w)$
    Compute $\nabla_\theta[\mathcal{J}_{\gamma_j}(\theta, w)]$
  end for
  Average the gradients $\Delta\theta = \frac{\sum_{j=1}^N \nabla_\theta[\mathcal{J}_{\gamma j}(\theta, w)]}{N}$
  Gradient descent $\theta \leftarrow \theta - \alpha \cdot \Delta\theta$
until convergence
return θ

Algorithm 2 Reduced-order model optimization using Envelope Theorem

Require: Task distribution Γ and step size α
Ensure: θ^*
{Model initialization}
$\theta \leftarrow \theta_0$
{Model optimization}
repeat
  Sample N tasks from $\Gamma \Rightarrow \gamma_j, j = 1, \ldots, N$
  for $j = 1, \ldots, N$ **do**
    Solve (4) to get $\mathcal{J}_{\gamma_j}(\theta, w)$
    Compute $\nabla_\theta[\mathcal{J}_{\gamma_j}(\theta, w)]$ by (12)
  end for
  Average the gradients $\Delta\theta = \frac{\sum_{j=1}^N \nabla_\theta[\mathcal{J}_{\gamma j}(\theta, w)]}{N}$
  Gradient descent $\theta \leftarrow \theta - \alpha \cdot \Delta\theta$
until convergence
return θ

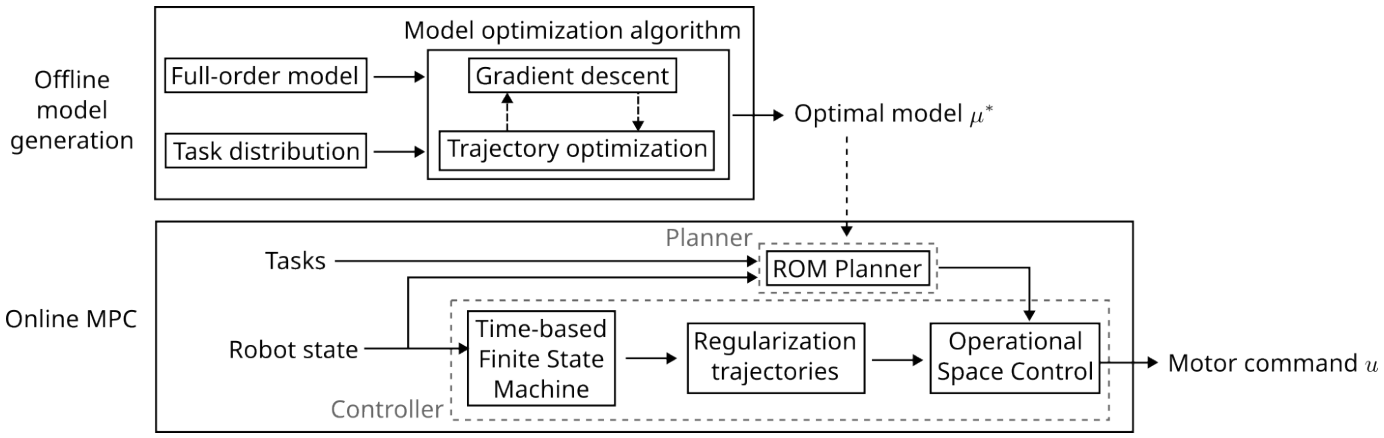

Fig. 2. Overview of the Pipeline for the synthesis and deployment of optimal reduced-order model using Algorithm 2

C. Using Reinforcement Learning

The third proposed solution involves the use of a model-based reinforcement learning approach to find the optimal model parameters. Rather than minimizing the cost as shown in equation (3), the approach focuses on maximizing the return $\sum_{t=1}^{T} r_t$, where r_t represents the reward at time t.In order to minimize the cost $h(x, u)$ and achieve a desired task γ, the reward function is defined as:

$$r = \exp\left(-w \cdot h\right) + 0.5 \exp\left(-\|\gamma - \gamma_{fb}\|_W\right), \qquad (14)$$

Here, w and W are constant weights, γ is the desired task value, and γ_{fb} is the achieved task value. The objective of the reinforcement learning problem is to maximize the expected return over the task distribution:

$$\max_{\theta} \mathbb{E}_{\gamma \sim \Gamma}[R] \qquad (15)$$

The Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) is used to solve equation (15). In the CMA-ES algorithm, let p_θ represent the probability distribution over θ. CMA-ES then solves the equation:

$$\max_{p_\theta} \mathbb{E}_\theta \sim p_\theta[\mathbb{E}_\gamma \sim \Gamma[R]] \qquad (16)$$

Given the difficulty of learning a policy for a large task space, curriculum learning is employed to facilitate the learning process. The continuous task domain Γ is discretized into a set Γ_d, which initially 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 equation (16), it is approximated by randomly drawing N_γ number of tasks from Γ_d and averaging the return. Let $\hat{R}$ represent the approximate expected return given by the model parameters θ. The evaluation of $\hat{R}$ is shown in algorithm 3. As Γ_d grows larger, the number of tasks N_γ are adjusted as $N_\gamma = (\rho_\gamma |\Gamma_d|)$, where ρ_γ is the sampling percentage and $|\cdot|$ counts the number of elements in a set. The algorithm to solve (16) is outlined in algorithm 4. In every iteration, CMA-ES samples a number of model parameters θ according to p_θ and evaluates the return for each θ. Based on these values, CMA-ES updates the mean and the covariance matrix of the parameter distribution p_θ.

Algorithm 3 Evaluation of return $\hat{R}(\theta, \{\gamma_j\})$

Require: model parameters θ and sampled tasks $\{\gamma_j\}$
1: **for** $j = 1, \ldots, N_\gamma$ **do**
2: Roll out an episode with the MPC
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 4 CMA-ES with curriculum learning

Require: initial mean θ_0 and variance σ_0^2 for the parameter distribution p_θ, and initial task set Γ_d
1: **for** $iter = 1, 2, \ldots$ **do**
2: **if** mod(iter, N_c) = 0 **then**
3: Grow the task set Γ_d
4: **end if**
5: Randomly draw N_γ tasks $\{\gamma_j\}$ from Γ_d
6: Sample N_θ parameters $\{\theta_i\} \sim p_\theta$
7: **for** each sampled θ_i **do**
8: Compute return $\hat{R}(\theta_i, \{\gamma_j\})$
9: **end for**
10: Update the mean and covariance of p_θ (CMA-ES)
11: **end for**

V. Experimental Results

A. Trajectory Optimization

They tested the model optimization algorithm 1 on two robots: a five-link planar robot and the 3D Cassie biped. They optimized two models for each robot, initialized with an LIP for the 2D model and an LIP plus a point-mass swing foot for the 4D model. The five-link robot has $q \in \mathbb{R}^7$, with the first 3 elements being the floating-base joint. The feature sets ϕ_e and ϕ_d were chosen to include elements of the LIP and the LIP with a swing foot, among other terms. The task set Γ includes walking at different speeds and ground inclines. The cost h_γ is the sum of the weighted norm of generalized velocity $\dot{q}$, input of the robot u, and

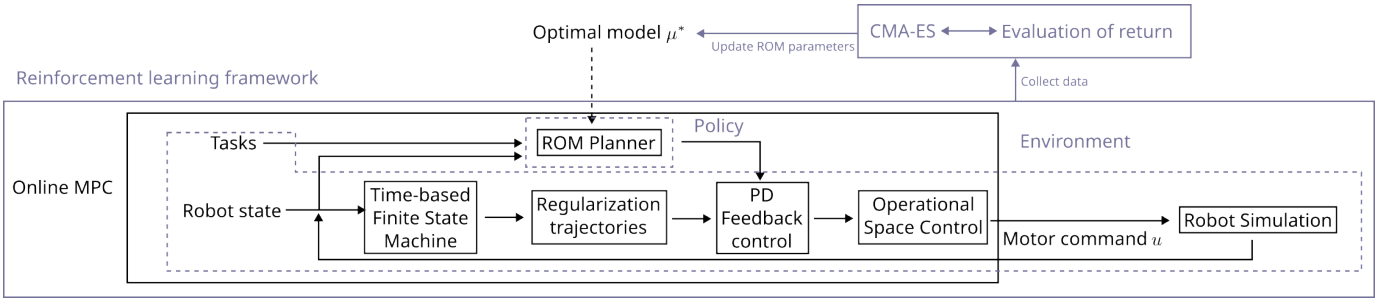

Fig. 3. Overview of the Pipeline for the synthesis and deployment of optimal reduced-order model for the reinforcement learning framework

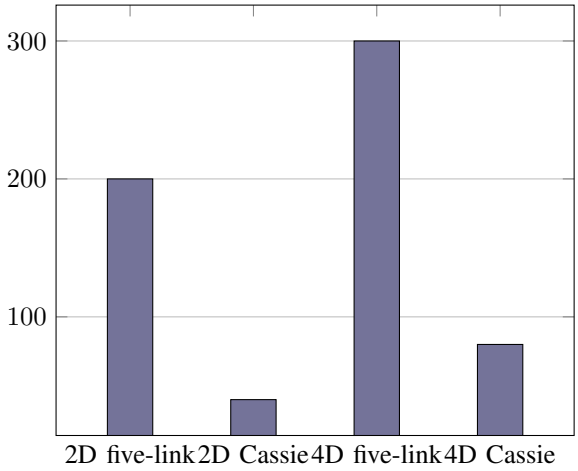

Fig. 4. The number of iterations for averaged sampled task cost to converge for Algorithm 1

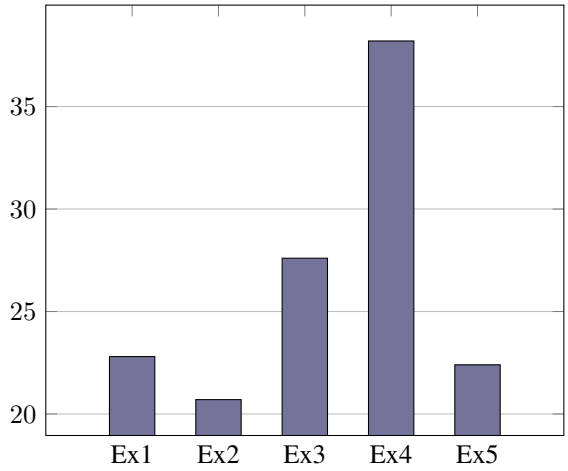

Fig. 5. Cost reduction percantage relative to initial model using Algorithm 2

input of the reduced-order model τ. The optimized model outperforms the LIP, better leveraging the natural dynamics of the five-link model. For the 3D Cassie, the generalized position $q \in \mathbb{R}^{19}$ is 19-dimensional. The feature sets ϕ_e and ϕ_d are constructed similarly to the five-link robot. The task set Γ includes walking at different speeds and ground inclines. They test their algorithm 2, by optimizing 3D reduced-order models on Cassie, initialized with a three-dimensional Linear Inverted Pendulum (LIP) model. This model represents a point-mass body moving at a constant vertical speed. They present five examples of model optimization, each varying in task space, monomial order, and dominant cost function terms. Examples 1 and 2 focus on minimizing the robot's energy consumption, with both starting from the same cost due to zero initial weights on the monomials. Example 1 uses second-order monomials, while Example 2 uses fourth-order monomials. The authors conclude that second-order monomials are sufficient for their task space. Example 3, which heavily penalizes acceleration, exhibits more vertical pelvis movement than Example 1 and has a higher initial cost, suggesting that the LIP model is more restrictive under this performance metric. Example 4, a subset of Example 3's task space with a larger stride length, also starts with a higher cost due to the LIP model's poor performance with large strides. However, this high initial cost allows for greater potential improvement. Finally, Example 5 penalizes the motor torques u in it's cost function and increases the task space dimension and only parameterizes the ROM dynamics g. The algorithm successfully finds an optimal model, demonstrating that parameterizing only the ROM dynamics can achieve near full model performance. The optimized models outperform the LIP model by expressing more input-efficient motions, thus better leveraging Cassie's natural dynamics. This optimization of the reduced-order model enhances the robot's performance while preserving model simplicity. The final, optimized model does not map easily to a simple, physical model if the embedding function, denoted as r, contains abstract basis functions such as monomials. This limits our ability to attach physical meaning to y and τ.

B. Performance evaluation

They evaluate the performance of the robot with the ROM settings: without reduced-order model embedding, with initial reduced-order model embedding and with optimal reduced-order model embedding. In the cases: trajectory optimization (open-loop), simulation (closed-loop) and hardware experiment with real Cassie (closed-loop). In all experiments the initial and the optimal ROM from Example 5 and the cost

	Speed Improvement
Straight line (5m)	41%
Fast 90-degree turn	38%
Downhill (20%)	39%
S-turns	41%

TABLE I
PERCENTAGE DIFFERENCE OF LIP AND OPTIMAL ROM SHOWN IN FIGURE 6

function $\tilde{h}_\gamma$ is used. To evaluate open loop performance they run full-model trajectory optimization over a wide range of tasks. For the simulation they use Drake. The MPC horizon is set to two footsteps, and the duration per step is fixed to 0.35 seconds which is the same as that of open-loop and evaluate the performance at different tasks, the duration of the simulation for a task is 12 seconds. For the hardware experiments heuristics are introduced to the MPC in order to stabilize Cassie well. For RL the CMA-ES optimizer adaptively selects the number of parameters sampled per iteration, N_θ. The sample density, ρ_γ, is set to 0.1 to expedite learning compared to evaluating all discretized task samples. The initial standard deviation, σ_0, of the parameters is kept small as the initial ROM is effective for simple tasks. A larger σ_0 doesn't improve performance and may cause divergence from optimal parameters. For curriculum learning, the initial task space is straight-line walking with stride lengths from -0.1 m to 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. The task space is expanded every 30 iterations for gradual model learning. The results of the performance evaluation for algorithm 2 show that the optimal ROM performs better than LIP in all tasks, with a maximum cost reduction of 30% in trajectory optimization and 24% in simulation. The optimal ROM also has an increased task capability. Which means that walking faster on slope and increased slope incline is possible. For algorithm 4 in simulation the optimal ROM reduces the cost across the task space by up to 21% and increased region size by 49%. Compared to algorithm 2, algorithm 4 has increased task space by 28% and increased cost reduction of 2.7%. To showcase the capability of the ROM synthesized by 2 Cassie has to finish a track, that consists of various segments requiring Cassie to turn by different angles and walk on different sloped grounds. Table I shows a roughly 40% faster finishing time compared to LIP, additionally on ground inclines of 50% the LIP fell where-as the optimal ROM continued.

VI. POSSIBLE FUTURE WORK

Future research could delve deeper into the trade-off between planning speed and model performance, where increased model expressiveness enhances performance but at the expense of slower planning. While linear models approximate full models well in certain task spaces, for challenging or complex task spaces, linear basis functions sacrifice significant performance when compared with those of higher degree. The possibility of optimizing models of varying degrees, depending on the complexity of the task space, is an avenue for

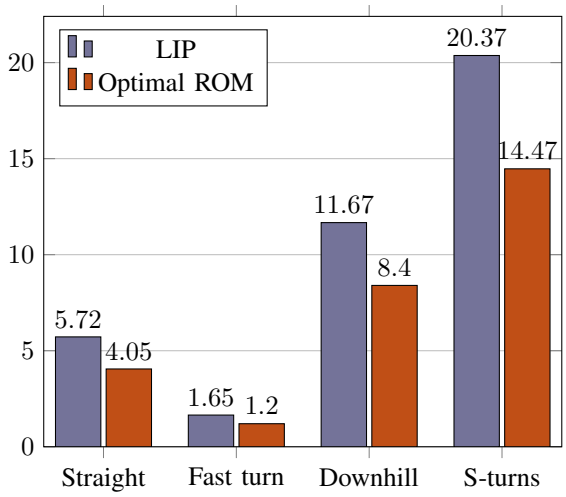

Fig. 6. Completion time in seconds of some of the segments of the simulated course using the ROM synthesized by 2

exploration. The role of initialization in bilevel optimization also presents an interesting avenue for future work. The requirement for the initial ROM to be feasible for the inner-level trajectory optimization, necessitates the initial ROM to be capable of walking, potentially limits the ability to use stochastic initialization to explore the entire ROM space. Determining the dimension of the ROM is another challenge that future research could address. While increasing the dimension theoretically enhances model performance, it comes at the expense of MPC computational speed. The user is currently required to determine the dimension of the ROM. Automatic ways to determine the dimension could be explored. Possible improvement for the reinforcement learning approach is to use different RL optimizers that improve performance. Also the embedding function r could also be parameterized.

REFERENCES

[1] Y.-M. Chen and M. Posa, "Optimal reduced-order modeling of bipedal locomotion," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8753–8760, IEEE, 2020.

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

[3] Y.-M. Chen, H. Bui, and M. Posa, "Reinforcement Learning for Reduced-order Models of Legged Robots" *arXiv preprint arXiv:2310.09873*, 2023.