Beyond Inverted Pendulums: Task-optimal Simple Models of Legged Locomotion

Yu-Ming Chen[1], Jianshu Hu[2] and Michael Posa[1]

Abstract—Reduced-order models (ROM) are popular in online motion planning due to their simplicity. A good ROM for control captures critical task-relevant aspects of the full dynamics while remaining low dimensional. However, planning within the reduced-order space unavoidably constrains the full model, and hence we sacrifice the full potential of the robot. In the community of legged locomotion, this has lead to a search for better model extensions, but many of these extensions require human intuition, and there has not existed a principled way of evaluating the model performance and discovering new models. In this work, we propose a model optimization algorithm that automatically synthesizes reduced-order models, optimal with respect to a user-specified distribution of tasks and corresponding cost functions. To demonstrate our work, we optimized models for a bipedal robot Cassie. We show in simulation that the optimal ROM reduces the cost of Cassie's joint torques by up to 23% and increases its walking speed by up to 54%. We also show hardware result that the real robot walks on flat ground with 10% lower torque cost. All videos and code can be found at https://sites.google.com/view/ymchen/research/optimal-rom.

Index Terms—Reduced-order models, model optimization, humanoid and bipedal locomotion, optimization and optimal control, real time planning and control

I. INTRODUCTION

State-of-the-art approaches to model-based planning and control of legged locomotion can be categorized into two types [1]. One uses the full-order model of a robot, and the other uses a reduced-order model (ROM). With the full model, we can leverage our full knowledge about the robot to achieve high performance [2]–[4]. However, this comes with the cost of heavy computation load, and it also poses a challenge in formal analysis, because modern legged robots have many degrees of freedom. To manage this complexity, the community of legged robots has embraced the use of reduced-order models.

Most reduced-order models adopt constraints (assumptions) on the full model dynamics while capturing the task-relevant part of the full-order dynamics. For example, the linear inverted pendulum (LIP) model [5], [6] assumes that the robot is a point mass that stays in a plane, which greatly reduces energy efficiency and limits the speed and stride length of the robot. The spring loaded inverted pendulum (SLIP) model [7] is a point mass model with spring-mass dynamics, which implies zero centroidal angular momentum rate and zero ground impacts at foot touchdown event. Therefore, when we plan for motions only in the reduced-space, we unavoidably

impose limitations on the full dynamics. This restricts a complex robot's motion to that of the low-dimensional model and necessarily sacrifices performance of the robot.

The above limitations of the reduced-order models have long been acknowledged by the community, resulting in a wide array of extensions that universally rely on human intuition, and are often in the form of mechanical components (a spring, a damper, a rigid body, the second leg, etc) [8]–[15]. The success of such model extensions which enable high-performance real time planning on hardware are listed as follows. Chignoli et al. [16] used a single rigid body model and assumed small body pitch and roll angle, in order to formulate a convex planning problem. Xiong et al. [8] extended LIP with a double-support phase while maintaining the zero ground impact assumption, so the model is still linear and conducive to a LQR controller [17]–[19]. Gibson et al. [20], [21] used the angular-momentum-based LIP which has better prediction accuracy than the traditional LIP. Dai et al. and Herzog et al. [22], [23] combined the centroidal momentum model with full robot configurations, and its real time application in planning was made possible thanks to Boston Dynamics' software engineering [24].

While some of the above extensions have improved the robot performance, it remains unclear which extension provides more performance improvement than the others, and we do not have a metric to improve the model performance with. Additionally, it has been shown that not all model extensions can significantly improve the performance of robots. For example, allowing the center of mass height to vary provides limited aid in the task of balancing [25], [26]. In our work, we aim to automatically discover the most beneficial extension of the reduced-order models by directly optimizing the models given a user-specified objective function and a task distribution.
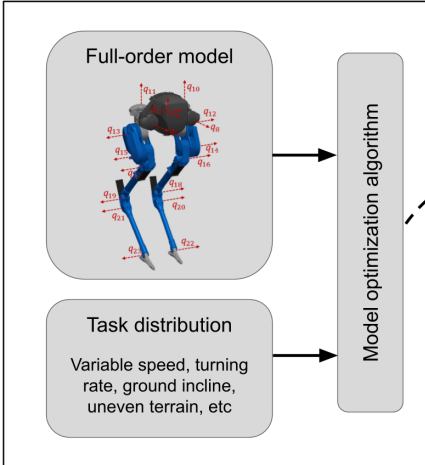
A. Related Work

Several researchers have sought to enhance the performance of the reduced-order model by mixing it with a full model in the planning horizon of Model Predictive Control (MPC). Li et al. [27] divided the horizon into two segments, utilizing a full model for the immediate part and a reduced-order model for the distant part. Subsequently, Khazoom et al. [28] systematically determined the optimal scheduling of these two models. Norby et al. [29] blended the full and reduced-order models while adaptively switching between the two. Our work is different from these existing works in that we directly optimize a reduced-order model instead of reasoning about scheduling two models to improve the overall model performance.

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

[2]The author is with the UM-SJTU Joint Institute, Shanghai Jiao Tong University, Shanghai, China. hjs1998@sjtu.edu.cn

Offline model generation (Section III)   **Real time planning and control** (Section IV)
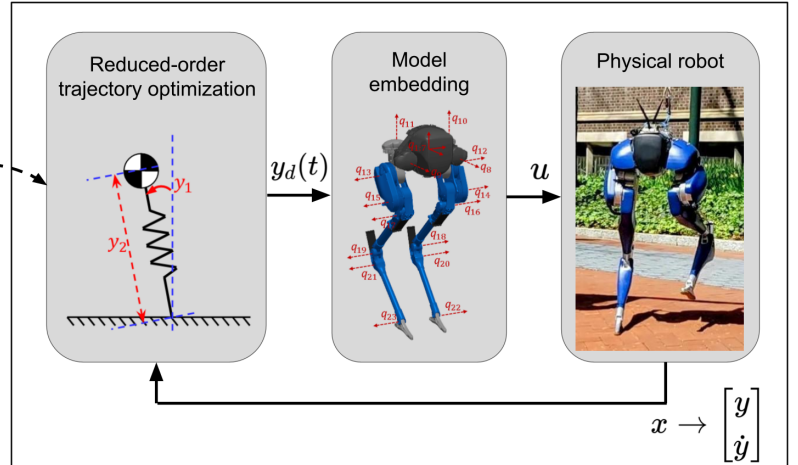

Fig. 1. An outline of the synthesis and deployment of optimal reduced-order models (ROM). Offline, given a full-order model and a distribution of tasks, we optimize a new model that is effective over the task space (Section III). Online, we generate new plans for the reduced-order model and track these trajectories on the true, full-order system (Section IV). This diagram also shows the bipedal robot Cassie (in the rightmost box) and its full model. Cassie has five motors on each leg – three located at the hip, one at the knee and one at the toe. Additionally, there are 2 leaf springs in each leg, and the spring joints are visualized by q_{16} to q_{19} in the figure. The springs are a part of the closed-loop linkages of the legs. We model these linkages with distance constraints, so there are no rods visualized in the model.

Classical approaches to finding a ROM often minimize the error between the ROM and the full model. Pandala et al. [30] attempted to close this gap implicitly in a learning framework, modeling the difference between the two models as a disturbance to the reduced dynamics. Our prior work [31] looked for an Integrable Whole-body Orientation model by minimizing the angular momentum error between the reduced-order and full-order model. Yamane [32] linearized the full model and reduced the dimension via principal component analysis (PCA), resulting in a ROM optimally approximating the full dynamics in a neighborhood of the linearization point. In contrast to these works, the approach presented in this paper leverages the fact that control on the full robot can be used to exactly embed low-dimensional models, and thus we judge the quality of such a model by a user-specified cost function instead of the modeling error. This definition and mechanism for assessing a ROM align with the state-of-the-art methods in model-based planning and control of legged robots – plan trajectories/inputs in the ROM space first and then track these reduced-order trajectories/inputs on the robot.

B. Contributions

The contributions of this paper are:

1) We propose a bilevel optimization algorithm to automatically synthesize new reduced-order models, embedding high-performance capabilities within low-dimensional representations. (This contribution was presented in the conference form [33] of this paper.)
2) We improve the model formulation of the prior work [33], and improve the algorithm efficiency by using the Envelope Theorem to derive the analytical gradient of an optimization problem. We provide more examples of model optimization, with different sizes of task space and basis functions.

3) We design a real time MPC controller for the optimal ROM, and demonstrate that the optimal model is capable of achieving higher performance in both simulation and hardware experiment of a bipedal robot Cassie.
4) We evaluate and compare the performances of reduced-order models in both simulation and hardware experiments. We analyze the performance gain and discuss the lessons learned in translating the model performance of an open-loop system to a closed-loop system.

C. Organization

The paper is organized as follows. Section II introduces the models of the Cassie robot and the background for this paper. Section III introduces our definition of a reduced-order model, formulates the model optimization problem, provides an algorithm that solves the problem, and finally demonstrates model optimization with a few examples. Section IV introduces an MPC for a specific class of ROMs used in Section V. Section V compares and analyzes the performance improvement in trajectory optimization, in simulation and in hardware. Section VI discusses the hybrid nature of legged robot dynamics and introduces the MPC for a general ROM. Finally, we discuss some of the lessons learned during the journey of realizing better performance on the robot in Section VII, and conclude the paper in Section VIII. The link to all videos and code for the examples is provided in the Abstract.

II. BACKGROUND

A. Reduced-order Models of Legged Locomotion

Modern legged robots like the Agility Robotics Cassie have many degrees of freedom and may incorporate passive dynamic elements such as springs and dampers. To manage this complexity and simplify the design of planning and control, reduced-order models have been widely adopted in the research community.

One observation, common to many approaches, lies in the relationship between foot placement, ground reaction forces, and the center of mass (CoM) [34]. While focusing on the CoM neglects the individual robot limbs, controlling the CoM position has proven to be an excellent proxy for the stability of a walking robot. CoM-based simple models include the LIP [5], [6], SLIP [7], hopping models [35], inverted pendulums [36], [37], and others. Since these models are universally low-dimensional, they have enabled a variety of control synthesis and analysis techniques that would not otherwise be computationally tractable. For example, numerical methods have been successful at finding robust gaits and control designs [38]–[41], and assessing stability [42].

Many of the aforementioned reduced-order models feature massless legs, eliminating any foot-ground impact during the swing foot touchdown event. When dealing with a robot or a model incorporating a foot of non-negligible mass, zero impacts necessitate zero swing foot velocity at touchdown. This constraint ensures the velocity continuity before and after the touchdown event.

A popular approach to using reduced-order models on legged robots is to first plan with the ROM to get desired ROM trajectories and desired foot steps, and then use a low-level controller to track the planned trajectories. This workflow is depicted in the right half of Fig. 1. As planning speed improves (e.g., through model complexity reduction), real time solving of the planning problem with a receding horizon (MPC) becomes feasible.

B. Models of Cassie

The bipedal robot Cassie (Fig. 1) is the platform we used to test our model optimization algorithm. Here we briefly introduce its model. Let the state of Cassie be $x = [q, v] \in \mathbb{R}^{45}$ where $q \in \mathbb{R}^{23}$ and $v \in \mathbb{R}^{22}$ are generalized position and velocity, respectively. We note that q and v have different dimensions, because the floating base orientation is expressed via quaternion. The conversion between $\dot{q}$ and v depends only on q [43]. The standard equations of motion are

$$M(q)\dot{v} = f_{cg}(q, v) + Bu + J_h(q)^T \lambda + \tau_{app}(q, v) \quad (1)$$

where M is the mass matrix which includes the reflected inertia of motors, f_{cg} contains the velocity product terms and the gravitational term, B is the actuation selection matrix, u is the actuator input, J_h is the Jacobian of holonomic constraints associated with the constraint forces λ, and τ_{app} includes the other generalized forces applied on the system such as joint damping forces. The forces λ contain ground contact forces and constraint forces internal to the four-bar linkages of Cassie. In simulation, the ground forces are calculated by solving an optimization problem based on the simulator's contact model [44]. In trajectory optimization, the forces λ are solved simultaneously with x and u while satisfying the dynamics, holonomic and friction cone constraints. Furthermore, we assume the swing foot collision with the ground during walking is perfectly inelastic in the trajectory optimization, so the robot dynamics is hybrid. Combining the discrete impact dynamics (from foot collision) with Eq. (1), we derive the hybrid equations of motion

$$\begin{cases} \dot{x} = f(x, u, \lambda), & x^- \notin S \\ x^+ = \Delta(x^-, \Lambda), & x^- \in S \end{cases} \quad (2)$$

where x^- and x^+ are pre- and post-impact state, Λ is the impulse of swing foot collision, f is the continuous-time dynamics, Δ is the discrete mapping at the touchdown event, and S is the surface in the state space where the event must occur [45], [46].

Cassie's legs contain four four-bar linkages – two around the shin links and the other two around the tarsus links. We simplify the model by lumping the mass of the rods of the tarsus four-bar linkages into the toe bodies, while the shin linkages are modeled with fixed-distance constraints. To simplify the model further, we assume Cassie's springs are infinitely stiff (or equivalently no springs), in which case $q \in \mathbb{R}^{19}$ and $v \in \mathbb{R}^{18}$. This assumption has been successfully deployed by other researchers [47], and it is necessary for the coarse integration steps in the trajectory optimization[3] in Section II-C. We also use this assumption in Section V when comparing the ROM performances between the trajectory optimization and simulation.

C. Trajectory Optimization

This paper will heavily leverage trajectory optimization within the inner loop of a bilevel optimization problem. We briefly review it here, but the reader is encouraged to see [49] for a more complete description. Generally speaking, trajectory optimization is a process of finding state $x(t)$ and input $u(t)$ that minimize some measure of cost h while satisfying a set of constraints C. Following the approach taken in prior work [50], [51], we explicitly optimize over state, input, and constraint (contact) forces $\lambda(t)$,

$$\begin{aligned} \min_{x(t), u(t), \lambda(t)} \quad & \int_{t_0}^{t_f} h(x(t), u(t)) dt \\ \text{s.t.} \quad & \dot{x}(t) = f(x(t), u(t), \lambda(t)), \\ & C(x(t), u(t), \lambda(t)) \le 0, \end{aligned} \quad (3)$$

where f is the dynamics of the system, λ are the forces required to satisfy holonomic constraints (inside $C \le 0$), and t_0 and t_f are the initial and the final time respectively. Standard approaches discretize in time, formulating (3) as a finite-dimensional nonlinear programming problem. For the purposes of this paper, any such method would be appropriate, while we use DIRCON [51] to address the closed kinematic chains of the Cassie robot. DIRCON transcribes the infinite dimensional problem in Eq. (3) into a finite dimensional nonlinear problem

$$\begin{aligned} \min_{w} \quad & \sum_{i=1}^{n-1} \frac{1}{2} \big(h(x_i, u_i) + h(x_{i+1}, u_{i+1}) \big) \delta_i \\ \text{s.t.} \quad & f_c(x_i, x_{i+1}, u_i, u_{i+1}, \lambda_i, \lambda_{i+1}, \delta_i, \alpha_i) = 0, \\ & \qquad\qquad\qquad\qquad\quad i = 1, ..., n-1 \\ & C(x_i, u_i, \lambda_i) \le 0, \qquad\quad i = 1, ..., n \end{aligned} \quad (4)$$

---

[3]Reher et al. [48] showed 7 times increase in solve time when using the full Cassie model (with spring dynamics) in trajectory optimization. Additionally, Cassie's spring properties can change over time and are hard to identify accurately, which discourages the use of the spring model on Cassie.

where n is the number of knot points, f_c is the collocation constraint for dynamics, $C \leq 0$ contains all the other constraints such as the four-bar-linkage kinematic constraints, δ_i is a constant time interval between knot point i and $i+1$, and the decision variables are

$$w = [x_1, ..., x_n, u_1, ..., u_n, \lambda_1, ..., \lambda_n, \alpha_1, ..., \alpha_{n-1}] \in \mathbb{R}^{n_w},$$

where $\alpha_1, ..., \alpha_{n-1}$ are slack variables specific to DIRCON. Eq. (4) uses the trapezoidal rule to approximate the integration of the running cost in Eq. (3), simplifying the selection of decision variables at knot points for evaluating the function h.

A large-scale nonlinear optimization problem such as (4) can be difficult to solve. To improve the convergence of the optimization, we manually scale the decision variables, constraints and the cost function, and we also add regularization terms (see Appendix A for details).

D. Bilevel Optimization

Since our formulation can be broadly categorized as bilevel optimization [52], we here briefly review its basics.

The basic structure of our bilevel optimization problem is written as

$$\min_\theta \left[\sum_i \min_w \Psi_i(w, \theta) \right] \tag{5}$$

The goal is to minimize the outer-level objective function $\sum_i \Psi_i(w_i^*(\theta), \theta)$ with respect to θ, where $w_i^*(\theta)$ is obtained by minimizing the inner-level objective $\Psi_i(w, \theta)$ parameterized by θ. Bilevel optimization has recently been used in various applications such as meta-learning [53], reinforcement learning [54], robotics [55], [56], etc.

Solving a bilevel program is generally NP-hard [57]. There are two types of methods to approach bilevel optimization. The first type is constraint-based [58], [59], where the key idea is to replace the inner level optimization with its optimality condition (such as the KKT conditions [60]), and finally solve a "single-level" constrained optimization. However, those methods are difficult to apply to the problem of this paper, because our inner-level is a trajectory optimization, and replacing it with its optimality condition will introduce a large number of dual variables and co-states, dramatically increasing the size of the single-level optimization. The second type is gradient-based [61], [62]. The idea is to maintain and solve the inner-level optimization, and then update the outer-level decision variable by differentiating through the inner-level solution using graph-unrolling approximation [62], [63] or implicit function theorem [64]. Compared to constraint-based methods, gradient-based methods maintain the bilevel structure and make bilevel optimization more tractable and efficient to solve.

In this paper, we use the Envelope Theorem [65], [66] and exploit the fact that our problem uses the same objective functions in the outer level and the inner level. This structure enables us to develop a more efficient gradient-based method (the second type) to solve our problem. Specifically, the gradient of the outer-level objective does not require differentiating the solution of the inner-level optimization with respect to the parameters. This leads to two numerical advantages
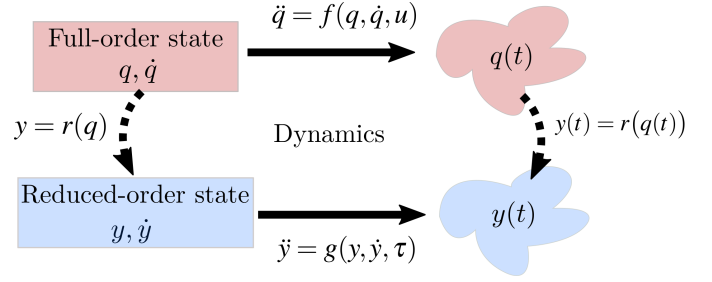

Fig. 2. Relationship of the full-order and reduced-order models. The generalized positions q and y satisfy the embedding function r for all time, and the evolution of the velocities $\dot{q}$ and $\dot{y}$ respects the dynamics f and g, respectively.

of our method over existing gradient-based methods. First, our method bypasses the computationally intensive implicit theorem, which requires the inverse of Hessian of the inner-level optimization. Second, our method leverages the inner-loop solver's understanding of active and inactive constraints, avoiding implementing the algorithm ourselves and avoiding tuning parameters such as the active set tolerance.

III. MODEL OPTIMIZATION

In this section, we propose a definition of reduced-order models, along with a notion of quality (or cost) for such models. We then introduce a bilevel optimization algorithm to optimize within our class of models.

A. Definition of Reduced-order Models

Let q and u be the generalized position and input of the full-order model, and let y and τ be the generalized position and input of the reduced-order model. We define a reduced-order model μ of dimension n_y by two functions – an embedding function $r : q \mapsto y$ and the second-order dynamics of the reduced-order model $g(y, \dot{y}, \tau)$. That is,

$$\mu \triangleq (r, g), \tag{6}$$

with

$$y = r(q), \tag{7a}$$
$$\ddot{y} = g(y, \dot{y}, \tau), \tag{7b}$$

where $\dim y < \dim q$ and $\dim \tau \leq \dim u$. As an example, to represent SLIP, r is the center of mass position relative to the foot, g is the spring-mass dynamics, and $\dim \tau = 0$ as SLIP is passive. Additionally, we note that the choices of r and g are independent of each other. For example, LIP and SLIP share the same r, but they have different dynamics function g (one has zero vertical acceleration and the other is the spring-mass dynamics).

The embedding function r can explicitly include the left or right leg of the robot (e.g. choosing left leg as support leg instead of right leg), in which case there will be two reduced-order models. In this paper, we assume to parameterize over left-right symmetric reduced-order models. As such, we explicitly optimize over a model corresponding to left-support, which will then be mirrored to cover both left and right-support phases. The details of this mirroring operation can be found in Appendix B.

Fig. 2 shows the relationship between the full-order and the reduced-order models. If we integrate the two models forward in time with their own dynamics, the resultant trajectories will still satisfy the embedding function r at any time in the future.

B. Problem Statement

As shown in the left half of Fig. 1, the goal is to find an optimal model μ^*, given a distribution Γ over a set of tasks. The distribution could be provided *a priori* or estimated via the output of a higher-level motion planner. The tasks might include anything physically achievable by the robot, such as walking up a ramp at different speeds, turning at various rates, jumping, running with a specified amount of energy, etc. The goal, then, is to find a reduced-order model that enables low-cost motion over the space of tasks,

$$\mu^* = \underset{\mu \in M}{\operatorname{argmin}} \; \mathbb{E}_\gamma \left[\mathcal{J}_\gamma(\mu) \right], \tag{8}$$

where M is the model space, $\mathbb{E}_\gamma$ takes the expected value over Γ, and $\mathcal{J}_\gamma(\mu)$ is the cost required to achieve the tasks $\gamma \sim \Gamma$ while the robot is restricted to a particular model μ.

With our model definition in Eq. (6), the problem in Eq. (8) is infinite dimensional over the space of embedding and dynamics functions, r and g. To simplify, we parametrize r and g with basis functions $\{\phi_{e,i} \mid i = 1, \ldots, n_e\}$ and $\{\phi_{d,i} \mid i = 1, \ldots, n_d\}$ with linear weights $\theta_e \in \mathbb{R}^{n_y \cdot n_e}$ and $\theta_d \in \mathbb{R}^{n_y \cdot n_d}$. Further assuming that the dynamics are affine in τ with constant multiplier, r and g are given as

$$y = r(q; \theta_e) \qquad = \Theta_e \phi_e(q), \tag{9a}$$
$$\ddot{y} = g(y, \dot{y}, \tau; \theta_d) = \Theta_d \phi_d(y, \dot{y}) + B_y \tau, \tag{9b}$$

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, we choose a constant value for B_y. Observing that physics-based rigid-body models lead to state-dependent values for B_y, one can also extend this method by parameterizing $B_y(y, \dot{y})$. Moreover, while we choose linear parameterization here, any differentiable function approximator (e.g. a neural network) can be equivalently used.

Let the model parameters be $\theta = [\theta_e, \theta_d] \in \mathbb{R}^{n_t}$. Eq. (8) can be rewritten as

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \; \mathbb{E}_\gamma \left[\mathcal{J}_\gamma(\theta) \right]. \tag{O}$$

From now on, we work explicitly in θ, rather than μ. As we will see in the next section, $\mathcal{J}_\gamma(\theta)$ is an optimal cost of a trajectory optimization problem, making Eq. (O) a bilevel optimization problem. Additionally, given the parameterization in Eq. (9), the ROM dimension n_y is fixed during the model optimization.

C. Task Evaluation

We use trajectory optimization to evaluate the task cost $\mathcal{J}_\gamma(\theta)$. Under this setting, the tasks γ are defined by a cost function h_γ and task-specific constraints C_γ. $\mathcal{J}_\gamma(\theta)$ is the optimal cost to achieve the tasks while simultaneously respecting

the embedding and dynamics given by θ. We note that the cost function h_γ is a function of the full model, although we occasionally refer to the cost evaluated by this function as the ROM performance because the ROM is embedded in the full model.

The resulting optimization problem is similar to (4), but contains additional constraints and decision variables for the reduced-order model embedding,

$$
\begin{aligned}
\mathcal{J}_\gamma(\theta) \triangleq \; \min_w \quad & \sum_{i=1}^{n-1} \frac{1}{2} \big(h_\gamma(x_i, u_i) + h_\gamma(x_{i+1}, u_{i+1}) \big) \delta_i \\
\text{s.t.} \quad & f_c(x_i, x_{i+1}, u_i, u_{i+1}, \lambda_i, \lambda_{i+1}, \delta_i, \alpha_i) = 0, \\
& \qquad\qquad\qquad\qquad\quad i = 1, \ldots, n-1 \\
& g_c(x_i, u_i, \lambda_i, \tau_i; \theta) = 0, \quad i = 1, \ldots, n \\
& C_\gamma(x_i, u_i, \lambda_i) \leq 0, \qquad i = 1, \ldots, n
\end{aligned}
\tag{10}
$$

where f_c and g_c are dynamics constraints for the full-order and reduced-order dynamics, respectively. The decision variables are $w = [x_1, ..., x_n, \; u_1, ..., u_n, \; \lambda_1, ..., \lambda_n, \; \tau_1, ..., \tau_n, \; \alpha_1, ..., \alpha_{n-1}]$, noting the addition of τ_i.

The formulation of dynamics and holonomic constraints of the full-order model are described in [51], while the reduced-order constraint g_c is

$$
\begin{aligned}
g_c &= \ddot{y}_i - g(y_i, \dot{y}_i, \tau_i; \theta_d) = 0 \\
\Rightarrow g_c &= J_i \dot{v}_i + \dot{J}_i v_i - g(y_i, \dot{y}_i, \tau_i; \theta_d) = 0
\end{aligned}
\tag{11}
$$

where

$$y_i = r(q_i; \theta_e), \quad \dot{y}_i = \frac{\partial r(q_i; \theta_e)}{\partial q_i} \dot{q}_i, \quad J_i = \frac{\partial r(q_i; \theta_e)}{\partial q_i}, \quad \text{and}$$
$$\dot{v}_i = M(q_i)^{-1} \left(f_{cg}(q_i, v_i) + Bu_i + J_h(q_i)^T \lambda_i + \tau_{app}(q_i, v_i) \right).$$

The constraint $g_c = 0$ not only explicitly describes the dynamics of the reduced-order model but also implicitly imposes the embedding constraint r via the variables y and $\dot{y}$. Therefore, the problem (10) is equivalent to simultaneous optimization of full-order and reduced-order trajectories that must also be consistent with the embedding r.

For readability, we rewrite Eq. (10) as

$$
\begin{aligned}
\mathcal{J}_\gamma(\theta) = \; \min_w \quad & \tilde{h}_\gamma(w) \\
\text{s.t.} \quad & \tilde{f}_\gamma(w, \theta) \leq 0,
\end{aligned}
\tag{TO}
$$

where $\tilde{h}_\gamma$ is the cost function of Eq. (10) and $\tilde{f}_\gamma \leq 0$ encapsulates all the constraints in Eq. (10). In Section V, we will use $\tilde{h}_\gamma$ to evaluate both the open-loop and closed-loop performance.

Remark 1. *Model optimization can change the physical meaning of a ROM. Regardless, if $J_i M^{-1} B$ (which maps a full model input u to a reduced-order acceleration $\ddot{y}_i$) has full row rank, the ROM can be exactly-embedded into the full model.*

D. Bilevel Optimization Algorithm

Since there might be a large or infinite number of tasks $\gamma \sim \Gamma$ in Eq. (O), solving for the exact solution is often intractable. Therefore, we use stochastic gradient descent to solve Eq. (O) (specifically in the outer optimization, as opposed to the inner

trajectory optimization). That is, we sample a set of tasks from the distribution Γ and optimize the averaged sample cost over the model parameters θ.

The full approach to (O) is outlined in Algorithm 1. Starting from an initial parameter seed θ_0, N tasks are sampled, and the cost for each task $\mathcal{J}_{\gamma_j}(\theta)$ is evaluated by solving the corresponding trajectory optimization problem (TO).

To compute the gradient $\nabla_\theta \left[\mathcal{J}_{\gamma_j}(\theta) \right]$, we previously [33] adopted an approach based in sequential quadratic programming. It introduced extra parameters (e.g. tolerance for determining active constraints) and required solving a potentially large and ill-conditioned system of linear equations which can take minutes to solve to good accuracy. Here, we take a new approach where we apply the Envelope Theorem and directly derive the analytical gradient $\nabla_\theta \left[\mathcal{J}_{\gamma_j}(\theta) \right]$ shown in Corollary 1 (also see Section II-D).

Proposition 1 (Differentiability Condition [67]). *Assume $\tilde{h}$ and $\tilde{f}$ are continuously differentiable functions, and consider an optimization problem*

$$\tilde{\mathcal{J}}(\theta) = \min_w \quad \tilde{h}(w, \theta) \\ s.t. \quad \tilde{f}(w, \theta) \leq 0, \tag{12}$$

where $\tilde{\mathcal{J}}(\theta)$ is the optimal cost of the problem. Let $w^*(\theta)$ be the optimal solution to Eq. (12). w^* is differentiable with respect to θ if the following conditions hold:

 1) *the second-order optimality condition for Eq. (12),*
 2) *linear independence constraint qualification (LICQ), and*
 3) *strict complementarity at w^*.*

Theorem 1 (Envelope Theorem [68]). *Assume the problem in Eq. (12) satisfies the differentiability condition. The gradient of the optimal cost $\tilde{\mathcal{J}}(\theta)$ with respect to θ is*

$$\nabla_\theta \left[\tilde{\mathcal{J}}(\theta) \right] = \frac{\partial \tilde{h}(w^*, \theta)}{\partial \theta} + \lambda^{*T} \frac{\partial \tilde{f}(w^*, \theta)}{\partial \theta}, \tag{13}$$

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

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

$$\nabla_\theta \left[\mathcal{J}_\gamma(\theta) \right] = \lambda^{*T} \frac{\partial \tilde{f}_\gamma(w^*, \theta)}{\partial \theta}, \tag{14}$$

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

Proof. The proof follows directly from Theorem 1. Note that the cost function in (TO) is independent of θ, in which case the first term of Eq. (13) becomes 0. $\square$

We note that there is, in general, no guarantee on global convergence when using Eq. (13) in a gradient descent algorithm, except for simple cases where $\tilde{h}$ and $\tilde{f}$ are convex functions in (w, θ) [69]. As for local convergence towards a stationary point, the gradient descent with Eq. (13) is guaranteed to converge with a sufficiently small step size. While there is no guarantee that the differentiability condition in Proposition 1 holds everywhere (in fact we expect it to fail under certain conditions), in practice we have observed that Algorithm 1 reliably converges. Additionally, the accuracy of the gradient $\nabla_\theta \left[\mathcal{J}_\gamma(\theta) \right]$ is bounded by the accuracy of the primal and

---

Algorithm 1 Reduced-order model optimization

Input: Task distribution Γ and step size α
Output: θ^*
     Model initialization
 1: $\theta \leftarrow \theta_0$
     Model optimization
 2: **repeat**
 3:     Sample N tasks from $\Gamma \Rightarrow \gamma_j, \; j = 1, ..., N$
 4:     **for** $j = 1, \ldots, N$ **do**
 5:         Solve (TO) to get $\mathcal{J}_{\gamma_j}(\theta)$
 6:         Compute $\nabla_\theta \left[\mathcal{J}_{\gamma_j}(\theta) \right]$ by Eq. (14)
 7:     **end for**
 8:     Average the gradients $\Delta\theta = \frac{\sum_{j=1}^{N} \nabla_\theta \left[\mathcal{J}_{\gamma_j}(\theta) \right]}{N}$
 9:     *Gradient descent* $\theta \leftarrow \theta - \alpha \cdot \Delta\theta$
10: **until** convergence
11: **return** θ

---

dual solutions to (TO) [67]. In practice, we observed that the gradient was accurate enough (showing local convergence behavior) with the default optimality tolerance and constraint tolerance given by solvers like SNOPT [70].

In Algorithm 1, the sampled tasks can sometimes be infeasible for the trajectory optimization problem due to a poor choice in ROM or numerical difficulties when solving (TO). In these cases, we do not include these samples in the gradient update step. This is a reasonable approach as we expect that optimizing the ROM for nearby tasks simultaneously improves performance for the failed task by continuity. This does have the potential to break the optimization process if large regions of the task space were infeasible, but in practice we have found this sample-rejection procedure robust enough to the occasional numerical difficulties.

The model optimization in Algorithm 1 is deemed to have converged if the norm of the average gradient of the sampled costs falls below a specified threshold. This threshold can be set on a case by case basis, depending on the robot models, tasks, etc. In our experiments, we simply look at the cost-iteration plots (e.g. Fig 4) and terminate the optimization when the cost has stopped decreasing visibly.

E. Examples of Model Optimization

In the trajectory optimization problem in Eq. (TO), we assume the robot walks with instantaneous change of support. That is, the robot transitions from right support to left support instantaneously, and vice versa. We consider only half-gait periodic motion, and so include right-left leg alternation in the impact map Δ.

We solve the problems (TO) in parallel in each iteration of Algorithm 1 using the SNOPT toolbox [70]. All examples were generated using the Drake software toolbox [44] and source code is available in the link provided in the Abstract.

1) Initialization and parameterization of ROM: To demonstrate Algorithm 1, we optimize for 3D reduced-order models on Cassie. The models are initialized with a three-dimensional LIP, of which the generalized position y is shown in Fig. 3.

Example #	1	2	3	4	5
Stride length (m)	[-0.4, 0.4]			[0.3, 0.4]	[0.0, 0.42]
Pelvis height (m)	[0.87, 1.03]				0.8
Ground incline (rad)	0				[-0.35, 0.35]
Turning rate (rad/s)	0				[-0.72, 0.72]
Stride duration (s)	0.35				0.35
Parameterize (r, g)?	both (r, g)				only g
Monomial order n_ϕ	2	4	2	2	4
Dominant cost in J_γ	u	u	$\dot{v}$	$\dot{v}$	u
Cost reduction	22.8%	20.7%	27.6%	38.2%	22.4%

TABLE I

EXAMPLES OF MODEL OPTIMIZATION. THIS TABLE INCLUDES THE TASK SPACE USED TO TRAIN MODELS (UNIFORM TASK DISTRIBUTION), THE HIGHEST ORDER OF THE MONOMIALS OF BASIS FUNCTIONS, THE DOMINANT TERM OF THE COST FUNCTION J_γ, AND THE COST REDUCTION PERCENTAGE (RELATIVE TO THE COST OF THE INITIAL MODEL).

Fig. 3. The linear inverted pendulum (LIP) model. It is a point mass model of which height is restricted in a plane. The point mass and the origin of this model correspond to the center of mass and the stance foot of the robot, respectively. In the examples of this paper, we initialize the reduced-order model to the LIP model during model optimization.

For reference, the equations of motion of the 3D LIP model are

$$\ddot{y} = \begin{bmatrix} \ddot{y}_1 \\ \ddot{y}_2 \\ \ddot{y}_3 \end{bmatrix} = \begin{bmatrix} c_g \cdot y_1/y_3 \\ c_g \cdot y_2/y_3 \\ 0 \end{bmatrix}, \tag{15}$$

where c_g is the gravitational acceleration constant. This model represents a point-mass body, where the body has a constant speed in the vertical direction.

We choose basis functions such that they not only explicitly include the position of the LIP, but also include a diverse set of additional terms. That is, the basis set ϕ_e includes the CoM position relative to the stance foot, and monomials of $\{1, q_7, ..., q_{19}\}$ up to n_ϕ-th order. Similarly, the feature set ϕ_d includes the terms in LIP dynamics (i.e. $c_g y_1/y_3$ and $c_g y_2/y_3$) and monomials of $\{1, y_1, y_2, y_3, \dot{y}_1, \dot{y}_2, \dot{y}_3\}$ up to n_ϕ-th order. With these basis functions, the ROM parameters θ can be trivially initialized to match the LIP model's.

2) Optimization Examples and Result: We demonstrate a few examples of model optimization and compare their results. The examples are shown in Table I along with their detailed settings. We note that it is a common practice to fix the duration to ease the problem difficulty of (TO) when dealing with high-dimensional robots [51], [71]. Freeing the duration can certainly be a variation of the examples and we left it for future work.

The optimization results are shown in Fig. 4, where the costs are normalized by the optimal cost of (TO) without ROM embedding (i.e. without the constraints g_c). The cost function h_γ was chosen to be the weighted sum of squares of the robot input u, the generalized velocity v and acceleration
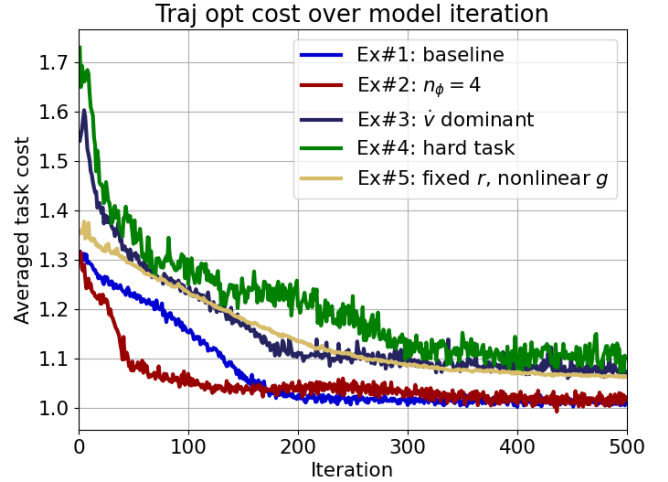

Fig. 4. The averaged cost of the sampled tasks of each model optimization iteration in Examples 1 to 5. Costs are normalized by the cost associated with the full-order model (i.e. the cost of full model trajectory optimization without any reduced-order model embedding). Therefore, the costs cannot go below 1. The costs at iteration 1 represent the averaged costs for the robots with the embedded initial reduced-order models, LIP. Note that the empirical average does not strictly decrease, as tasks are randomly sampled and are of varying difficulty.

$\dot{v}$. In Examples 1, 2 and 5, we heavily penalize the input term which is a proxy of a robot's energy consumption. For the other examples, we heavily penalize the acceleration $\dot{v}$. We observed that Cassie's motions with the initial ROM are very similar among all examples. In contrast, the motions with optimal ROMs are mostly dependent on the cost function h_γ, given the same ROM parameterization. Compared to Example 1, the optimal motion of Example 3 shows more vertical pelvis movement.

A comparison between Example 1 to Example 2 shows the effect of the order of the monomials n_ϕ in the basis function. We can see in Fig. 4 that these two examples share the same starting cost, because the initial weights on the monomials are zeros, making the trajectory optimization problems identical. Additionally we can also see that parameterizing the ROM with second-order monomials seems sufficient for the task space of Examples 1 and 2, since the final normalized cost is close to 1.

A comparison between Example 1 to Example 3 shows the effect of different choices of cost function h_γ. The initial cost of Example 3 is much higher than Example 1's, which we

can interpret as the LIP model being more restrictive under the performance metric of Example 3 than that of 1.

A comparison between Example 3 and Example 4 shows the effect of the task space. Example 4's task space is a subset of Example 3's, specifically the part of the task space with bigger stride length. We would expect the LIP model does not perform well with big stride length, and indeed Fig. 4 shows that the initial cost of Example 4 is higher than that of Example 3. Fortunately, a high initial cost provides us with a bigger room of potential improvement. As we see in Table I, Example 3 has higher cost reduction than Example 1, and Example 4 has the highest cost reduction.

In Example 5, the dimension of the task space[4] is increased compared to the other examples, and we only parameterize the ROM dynamics g. That is, the embedding function r remains to be a simple forward kinematic function – the center of mass position relative to the stance foot. In this case, the algorithm was again able to find an optimal model, and the result shows that parameterizing only the ROM dynamics g is sufficient enough for achieving near full model performance (about 5% higher than full-order model's performance).

The optimized models are capable of expressing more input-efficient motions than the LIP model, better leveraging the natural dynamics of Cassie. The ROM optimization improves the robot performance, while maintaining the model simplicity. We note that the optimal model, unlike its classical counterpart, does not map easily to a physical model, if the embedding function r contains abstract basis functions such as monomials. While this limits our ability to attach physical meaning to y and τ, it is a sacrifice that one can make to improve performance beyond that of hand-designed approaches.

IV. MPC FOR A SPECIAL CLASS OF ROM

After a ROM is optimized, we embed it in the robot via an MPC to achieve desired tasks, depicted in Fig. 1. Specifically, in this and the next section (Sections IV and V), we build upon Example 5. Example 5 limits the ROM to a fixed embedding function r, the CoM position relative to the stance foot. This physically-interpretable embedding simplifies the planner and enables a richer performance analysis in Section V. The planner for a general ROM will be introduced in Section VI.

The MPC structure is shown in Fig. 5. It contains a high-level planner in the reduced-order space (Section IV-A) and a low-level tracking controller in the full-order space (Section IV-B). The high-level planner receives the robot state and tasks, and plans for the desired ROM trajectory and the footsteps of the robot. The controller tracks these desired trajectory and footsteps, while internally using nominal trajectories to handle the system redundancy.

A. Planning with Reduced-order Models

We formulate a reduced-order trajectory optimization problem to walk n_s strides, using direct collocation method described in Section II-C to discretize the trajectory into n knot

---

[4]The dimension here is the non-degenerate dimension, meaning the task dimension with non-zero volume. In Example 5, the dimension is 3 because we vary the stride length, ground incline and turning rate.
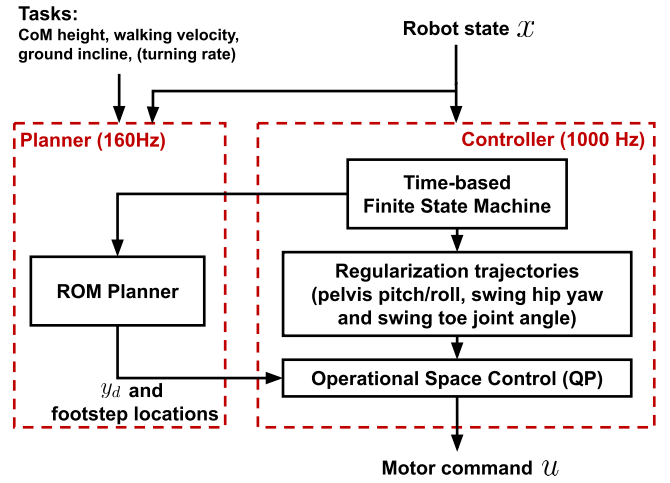

Fig. 5. The diagram of the model predictive control (MPC) introduced in Section IV. The MPC is composed of the controller process and the planner process, and it contains a time-based finite state machine which outputs either left or right support state. This finite state machine determines the contact sequence of the high-level planner and the contact mode of the low-level model-based controller. The high-level planner solves for the desired reduced-order model trajectories and swing foot stepping locations, given tasks (commands) and the finite state. For reduced-order models without body orientation (e.g. CoM model without moment of inertia), we send the turning rate command to the controller process instead of planner process. Inside the controller process, the regularization trajectories are used to fill out the joint redundancy of the robot. These regularization trajectories are derived from 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 with the desired heading angle. All desired trajectories are sent to the Operational Space Controller (OSC) which is a quadratic-programming based inverse-dynamics controller [72], [73].

points. Under the premise that the ROM embedding r is the CoM, we further assume the ROM does not have continuous inputs τ (e.g. center of pressure) but it has discrete inputs $\tau_{fp} \in \mathbb{R}^2$ which is the stepping location of the swing foot relative to the stance foot. Let $z = [y, \dot{y}] \in \mathbb{R}^{2n_y}$, and let z^- and z^+ be the reduced state of pre- and post-touchdown event, respectively. The discrete dynamics is

$$z^+ = z^- + B_{fp}\tau_{fp} \tag{16}$$

with

$$B_{fp} = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \end{bmatrix}^T.$$

The first two rows of Eq. (16) correspond to the change in stance foot reference for the COM position. The last three rows are derived from the assumption of zero ground impact at the foot touchdown event.

To improve readability, we stack decision variables into bigger vectors $z = [y, \dot{y}] \in \mathbb{R}^{2n_y}$, $Z = [z_0, z_1, ..., z_n] \in \mathbb{R}^{2n_y(n+1)}$, and $\mathrm{T}_{fp} = [\tau_{fp,1}, ..., \tau_{fp,n_s}] \in \mathbb{R}^{2n_s}$. The cost function of the planner is quadratic and expressed in terms of Z and T_{fp}. The planning problem is

$$\min_{Z, \mathrm{T}_{fp}} \quad \|Z - Z_d\|_{W_Z}^2 + \|\mathrm{T}_{fp}\|_{W_\mathrm{T}}^2$$

$$\text{s.t.} \quad \text{ROM continuous dynamics (Eq. (7b)),}$$
$$\text{ROM discrete dynamics (Eq. (16)),} \tag{17}$$
$$C_{kinematics}(Z, \mathrm{T}_{fp}) \leq 0,$$
$$z_0 = \text{current feedback reduced-order state,}$$

Trajectory y_i^{osc}	dim y_i^{osc}	cost weight W			K_p			K_d		
		x	y	z	x	y	z	x	y	z
reduced-order model	3	0.1	0	10	10	0	50	0.2	0	1
pelvis orientation	3	2	4	0.02	200	200	0	10	10	10
swing foot position	3	4	4	4	150	150	200	1	1	1
swing leg hip yaw joint	1	0.5			40			0.5		
swing leg toe joint	1	2			1500			10		

TABLE II

TRAJECTORIES AND GAINS IN THE OPERATIONAL SPACE CONTROL (OSC)

where W_Z and W_T are the weights of the norms, Z_d is a stack of desired states which encourage the robot to reach a goal location and regularize velocities, and $C_{kinematics} \leq 0$ is the constraints on step lengths and stepping locations relative to the CoM. After solving Eq. (17), we reconstruct the desired ROM trajectory $y_d(t)$ from the optimal solution Z^*, and we construct desired swing foot trajectories from T^*_{fp} with cubic splines.

B. Operational Space Controller

A controller commonly used in legged robots is the quadratic-programing-based operational space controller (QP-based OSC), which is also referred to as the QP-based whole body controller [72], [73]. Assume there are N_y number of outputs $y^{osc}(q)$, with desired outputs $y^{osc}_{i,d}(t)$, where $i = 1, 2, ... N_y$. For each output (neglecting the subscript i), we can derive the commanded acceleration as the sum of the feedforward acceleration of the desired output and a PD control law

$$\ddot{y}^{osc}_{cmd} = \ddot{y}^{osc}_d + K_p(y^{osc}_d - y^{osc}) + K_d(\dot{y}^{osc}_d - \dot{y}^{osc}).$$

At a high level, the OSC solves for robot inputs that minimize the output tracking errors, while respecting the full model dynamics and constraints (essentially an MPC but with zero time horizon). The optimal control problem of OSC is formulated as

$$\min_{\dot{v},u,\lambda,\epsilon} \quad \sum_{i=1}^{n_y} \|\ddot{y}^{osc}_i - \ddot{y}^{osc}_{i,cmd}\|^2_{W_i} + \|u\|^2_{W_u} + \|\epsilon\|^2_{W_\epsilon} \quad (18a)$$

$$\text{s.t.} \quad \ddot{y}^{osc}_i = J_i \dot{v} + \dot{J}_i v, \quad i = 1, ..., N_y \quad (18b)$$

$$\text{Dynamics constraint (Eq. (1))} \quad (18c)$$

$$\epsilon = J_h \dot{v} + \dot{J}_h v \quad (18d)$$

$$u_{min} \leq u \leq u_{max} \quad (18e)$$

$$\text{Contact force constraints} \quad (18f)$$

where $\|\cdot\|_W$ is the weighted 2-norm, (18d) contains Cassie's four-bar-linkage constraints, fixed-spring constraints and relaxed contact constraints (relaxed by slack variables ϵ), and (18f) includes friction cone constraints, non-negative normal force constraints and force blending constraints for stance leg transition.

Table II shows all trajectories tracked by the OSC and their corresponding gains and cost weights. The trajectories of the reduced-order model, pelvis orientation and swing foot position are all 3 dimensional, while the hip yaw joint and toe joint of the swing foot are 1 dimensional. The symbols (x,y,z) in Table II indicate the components of the tracking target. They do not necessarily mean the physical (x, y, z) axes for

the reduced-order model, since the model optimization might produce a physically non-interpretable model embedding r.

In the existing literature of bipedal robots, robot's floating base position (sometimes the CoM position) and orientation are often chosen to be control targets. They have 6 degrees of freedom (DoF) in total. In the case of fully-actuated robots (i.e. robots with flat feet), there is enough control authority to servo both the position and orientation. For underactuated robots, the existing approaches often give up tracking the trajectories in the transverse plane (x and y axis), because it is not possible to instantaneously track trajectories whose dimension is higher than the number of actuators (or we have to trade off the tracking performance). In this case, motion planning for discrete footstep locations is used to regulate the underactuated DoF. In our control problem, we also face the same challenge since Cassie has line feet. The total dimension of the desired trajectories in Table II is 11, while Cassie only has 10 actuators. Following the common approach, we choose not to track the second element of the ROM in OSC, because it corresponds to the lateral position of the CoM for the initial model (a competing tracking objective to the pelvis roll angle) and maintaining a good pelvis roll tracking is crucial for stable walking. Instead, the second element of ROM is regulated by the desired swing foot locations via the planner in Eq. (17), even though the OSC does not explicitly track it.

C. Hardware Setup and Solve Time

We implement the MPC in Fig. 5 using the Drake toolbox [44], and the code is publicly available in the link provided in the Abstract. In hardware experiment, the MPC planner runs on a laptop equipped with Intel i7 11800H, and everything else (low-level controller, state estimator, etc) on Cassie's onboard computer. These two computers communicate via LCM [74]. A human sends walking velocity commands to Cassie with a remote controller. Cassie is able to stably walk around with both the initial ROM and the optimal ROM (shown in the supplementary video).

The planning horizon was set to 2 foot steps with stride duration being 0.4 seconds. With cubic spline interpolation between knot points, we found that 4 knot points per stride was sufficient. IPOPT [75] was used to solve the planning problem in Eq. (17), and the solve time was on average around 6 milliseconds with warm-starts. We observed that this solve time was independent of the reduced-order models (initial or optimal) in our experiments. In contrast to the ROM, similar code required tens of seconds for the simplified Cassie model for a single foot step. As the following sections will show, Cassie's performance (with respect to the user-specified cost function) with the optimal model is better than with

	(**R1**) no ROM	(**R2**) initial ROM	(**R3**) optimal ROM
(**C1**) Traj. Opt.	X	X	X
(**C2**) Simulation		X	X
(**C3**) Experiment on real Cassie		X	X

TABLE III
EXPERIMENTS CONDUCTED IN SECTION V (MARKED WITH X)

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 IV
CRITERIA TO DETERMINE PERIODIC WALKING GAITS

the initial model. This demonstrates that the use of ROM greatly increases planning speed, and that the optimized ROM improves the performance of the robot.

V. PERFORMANCE EVALUATION AND COMPARISON

In this section, we evaluate the performance of the robot (with respect to a user-specified cost function $\tilde{h}_\gamma$ in Section III-C) in the following ROM settings:

(R1) without reduced-order model embedding,
(R2) with initial reduced-order model embedding,
(R3) with optimal reduced-order model embedding.

Additionally, the evaluation is done in the following cases:

(C1) trajectory optimization (open-loop),
(C2) simulation (closed-loop),
(C3) hardware experiment with real Cassie (closed-loop),

where (C1) is labeled as open-loop, while the others are considered closed-loop, because trajectory optimization is an optimal control method that solves for control inputs and feasible state trajectories simultaneously, without requiring a feedback controller. Table III lists the experiments conducted in this section. We note that (C1) is the same as Eq. (TO) but with a different task distribution, and that (R1) is only evaluated in (C1) because it serves as an idealized benchmark for comparison.

A. Experiment Motivations

1) Motivation for (C1): In Section III, we optimized for a reduced-order model given a task distribution. Here, one objective is to evaluate how well the model generalizes to out-of-distribution tasks. Additionally, trajectory optimization provides the ideal performance benchmark for the closed-loop system to compare to.

2) Motivation for (C2): Trajectory optimization is used in Eq. (TO) to find the optimal model based on the open-loop performance. (C2) evaluates how well the cost reduction in open-loop can be translated to the closed-loop system with the MPC from Section IV.

3) Motivation for (C3): (C3) evaluates how well the performance improvement can be translated to hardware.

B. Experiment Setups

In all experiments in this section, we use the initial and the optimal ROM from Example 5. Additionally, for all performance evaluations, we use the cost function $\tilde{h}_\gamma$ from Example 5 which mainly penalizes the joint torques.

1) (C1) Trajectory Optimization: We evaluate the open-loop performance by running the full-model trajectory optimization in Eq. (TO) over a wide range of tasks (stride length, turning rate, ground incline, etc). As a special case, (C1) combined with (R1) corresponds to Eq. (TO) without the constraint $g_c = 0$.

2) (C2) Simulation: We use Drake simulation [44]. 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. Similar to (C1), we evaluate the performance at different tasks. For each desired task, we run a simulation for 12 seconds and extract a periodic walking gait based on a set of criteria listed in Table IV. Then we compute the cost and the actual achieved tasks (stride length, turning rate, etc) of that periodic gait.

3) (C3) Hardware Experiment: Some heuristics are introduced to the MPC in order to stabilize Cassie well. For example, we add a double-support phase to smoothly transition between two single-support phases by linearly blending the ground forces of the two legs. This is critical for Cassie, because unloading the springs of the support leg too fast when transitioning into swing phase can cause foot oscillation and bad swing foot trajectory tracking. The double-support phase duration is set to 0.1 seconds, and the swing phase duration is decreased to 0.3 seconds, compared to the nominal 0.35 seconds of stride duration in the trajectory optimization.

The hardware setup is described in Section IV-C. During the experiment, we send commands to walk Cassie around and make sure that the safety hoist does not interfere with Cassie's motion. After the experiment, we apply the criteria listed in Table IV to extract periodic gait for performance evaluation.

C. Turning and Sloped Walking in Simulation

The goal of this section is to evaluate model performance in simulation. We use the initial and the optimal model from Example 5 of Section III-E. The training task distribution of this model covers various turning rates, ground inclines and positive stride lengths, shown in Table I. During performance evaluation (both (C1) and (C2)), we increase the task space size to two times of that of training stage in order to examine the optimal model's performance on the both seen (from training) and unseen tasks.

To visualize the performance improvement, we compare the cost landscapes between the initial and the optimal model. For (C1) and (C2), we first derive the cost landscapes of both models using the cost function $\tilde{h}_\gamma$ and then superimpose them in terms of cost ratio (i.e. ratio of (R3)'s cost to (R2)'s cost). The cost landscape comparisons are shown in Fig. 6. The red-blue color bar represents different levels of performance improvement in terms of $\tilde{h}_\gamma$. Green color corresponds to the tasks acquired by the optimal model (i.e. the task that the initial
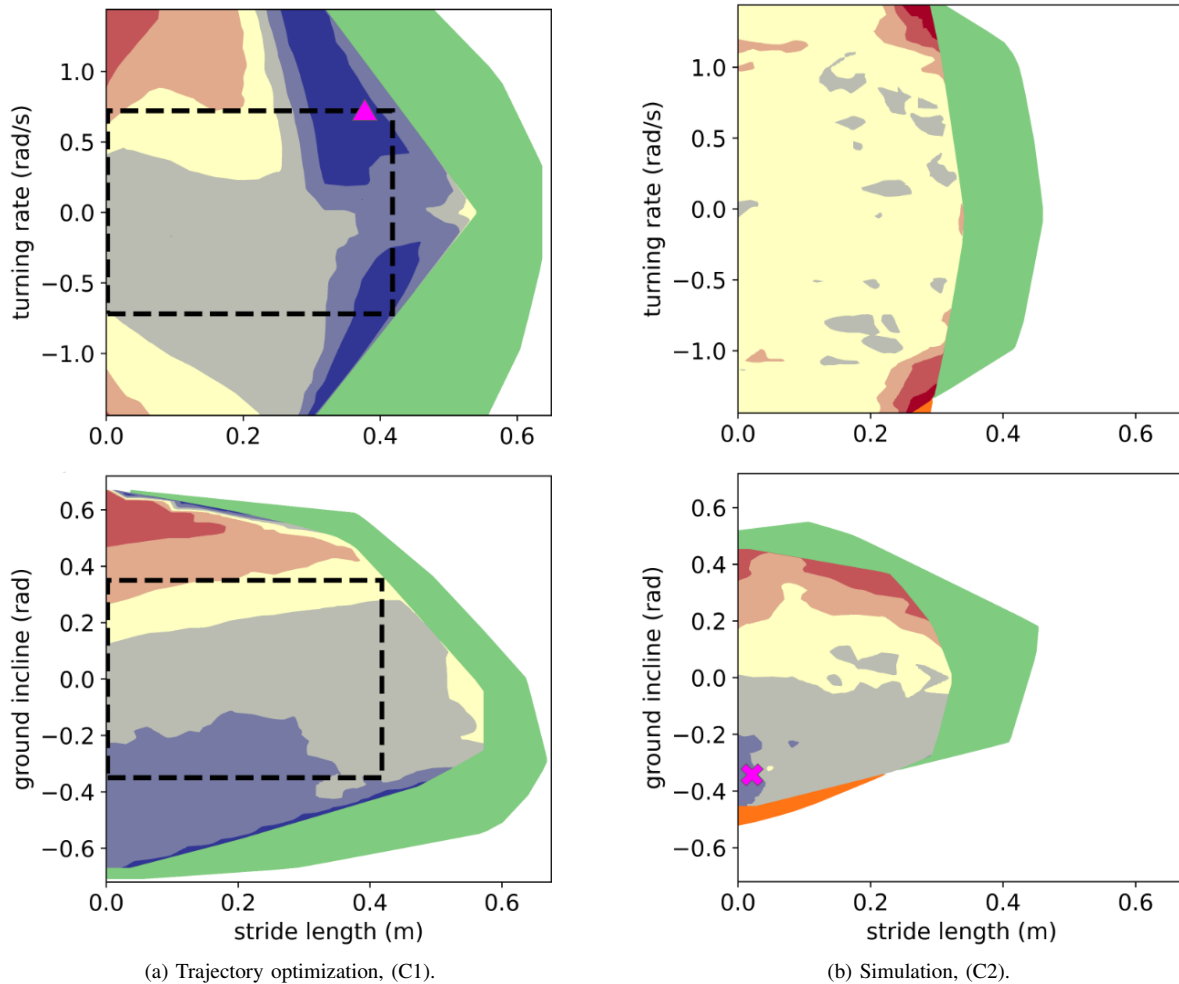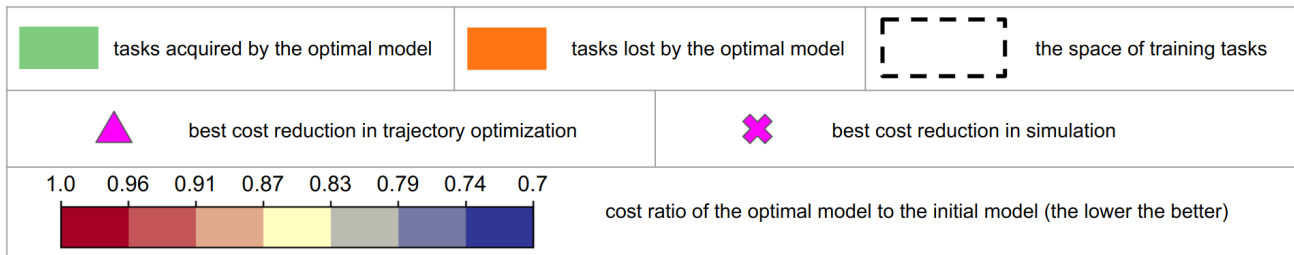
Fig. 6. Cost comparison between the initial model (R2) and the optimal model (R3). Each plot shows the ratio of the optimal model's cost to the initial model's cost. For these examples, the ROM reduces the cost across the entire task space. The color scheme red-to-blue illustrates the degree to which the ROM shows improvement, with red corresponding to a minimal improvement and blue to a 30% reduction. The scales of the axes are the same between the trajectory optimization (C1) and the simulation (C2) for ease of comparisons.

model cannot execute). Orange color corresponds to the task lost by using the optimal model. We see in Fig. 6 that the optimal ROM outperforms the LIP across all depicted tasks, as the cost ratios are smaller than 1, indicated by the color bar. In trajectory optimization, the maximum cost reduction is 30%, occurring at a stride length of approximately 0.38 meters and a turning rate of 0.71 rad/s, indicated by the pink triangle in Fig. 6a. On the other hand, the maximum cost reduction in simulation is 23%, observed at a stride length of approximately 0.02 meters and a -0.34 radians ground incline, indicated by the pink cross in Fig. 6b. This reduction in cost implies, for instance, that Cassie is able to complete the same task with

23% less joint torque in simulation.

Besides the improvement in terms of cost ratio, we also observe in simulation that the optimal ROM gained new task capability (indicated by the green area). For example, Cassie is capable of walking 54% faster on a slope of 0.2 radian when using the optimal ROM. Cassie also gets better in climbing steeper hills. At 0.1m stride length, Cassie can climb up a hill with 32% steeper incline. Overall, the task region gained is much bigger than the lost in simulation.

Comparing the cost landscape between the trajectory optimization and simulation, we can see that the task region
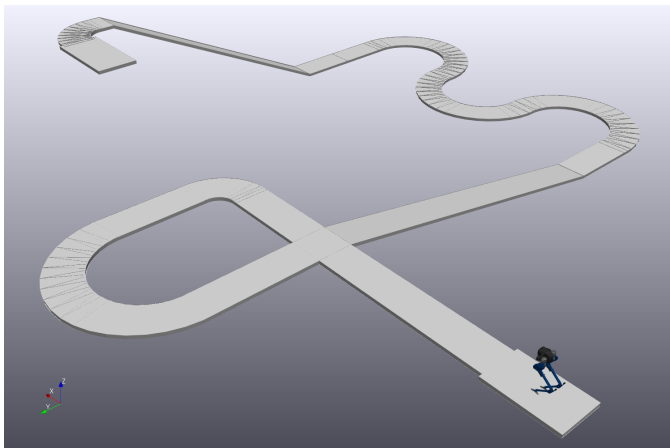
Fig. 7. A track designed to showcase the performance difference between the LIP and the optimal ROM in simulation. The video of Cassie finishing the track can be found in the supplementary materials.

	LIP	Optimal ROM	Speed Improvement
Straight line (5m)	5.72 (s)	4.05 (s)	41%
Fast 90-degree turn	1.65 (s)	1.2 (s)	38%
Downhill (20%)	11.67 (s)	8.4 (s)	39%
S-turns	20.37 (s)	14.47 (s)	41%
Uphill (50%)	failed	17.73 (s)	-

TABLE V
COMPLETION TIME FOR SOME SEGMENTS OF THE COURSE.

gained are similar[5]. Cassie in general can walk faster at different turning rates and on different ground inclines. We also observed that the cost landscapes of the open- and closed-loop share a similar profile in ground incline. Both show bigger cost reduction in walking downhill than uphill. In contrast, the landscapes in turning rate look different between the open and closed loop. This is partially because there is only one stride (left support phase) in trajectory optimization while we average the cost over 4 strides in the simulation. Additionally, there is a difference in stride lengths between the open- and closed-loop, showing a control challenge in stabilizing around high-speed nominal trajectories. This gap can be mitigated by increasing the control gains in simulation. However, we avoid using unrealistic high gains because they do not work on hardware.

In Fig. 8, the dashed-line boxes represent the training task space used in the model optimization stage. There is not a strong correlation between the cost ratio and the training space. However, we can observe that the performance of the optimal ROM generalizes well to the unseen task (region outside the dashed-line box). For example, the cost reduction ratio stays around 16% at different turning rates in simulation.

Lastly, we designed a track shown in Fig. 7 for Cassie to finish as fast as possible to showcase the capability of the optimal ROM. The track includes various segments requiring Cassie to turn by different angles and walk on different sloped grounds. To enable Cassie to race through the track, we implemented a high-level path-following controller that sends commands (such as walking velocity) to the MPC. We tuned the path-

[5]In the case of (C1), there was not a clear threshold for defining the failure of a task. We picked a cost threshold at which the robot's motion does not look abnormal.
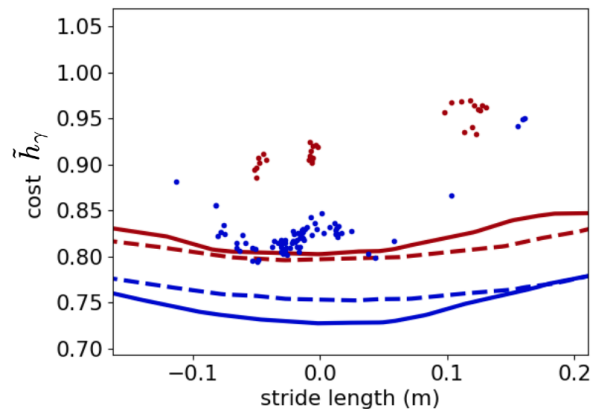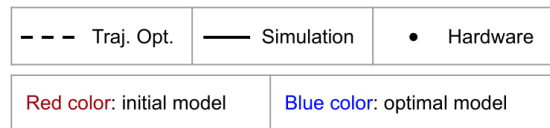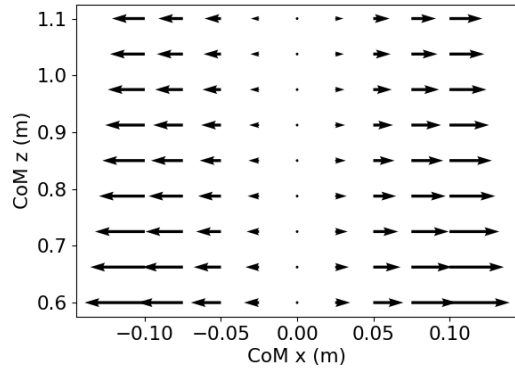

Fig. 8. Cost comparison between the initial model (R2) and the optimal model (R3). The cost $\tilde{h}_\gamma$ is the cost function in Eq. (TO). For trajectory optimization and simulation, we densely sampled the tasks and interpolated the costs. For the hardware experiment, we plot the costs of collected data points directly in the figure without interpolation. To collect the data on hardware, we used a remote control to walk Cassie around, and we applied a moving window of 4 foot steps to extract periodic gaits according to Table IV. We note that there was about 2 cm of height variation in the hardware experiment, but we normalized every extracted data point to the same height using the height-cost relationship from the trajectory optimization to make fair comparisons.

following controller parameters, so that Cassie can finish many segments as fast as possible without falling off the track. The parameters were tuned separately for each model. We observe that, during high-speed walking, the optimal model can track the commanded velocity better than the LIP model, and that the actual top speed is achieved understand different commanded velocities. Quantitatively, Cassie on average can walk about 40% faster with the optimal ROM, shown in Table V. This speed improvement reflected the periodic-walking result shown in the cost landscape plots in Fig. 6. Additionally, we observed that for the task of 50% ground incline Cassie with LIP exhibited many stop-and-go motions and eventually fell, while Cassie with the optimal ROM was able to complete the incline steadily.
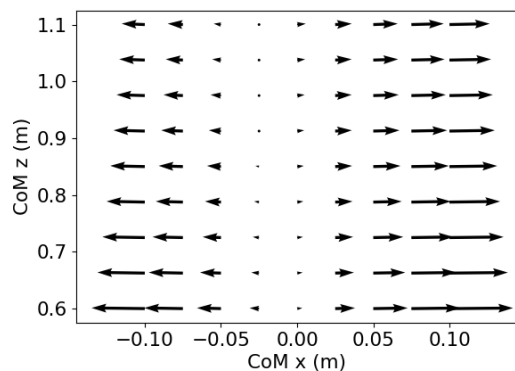
D. Straight-line Walking on Hardware

In this section, we aim to evaluate the model performance on the real robot. We evaluate the cost for different stride lengths while fixing the pelvis height (0.95 m), and then plot the costs of both the initial and the optimal model directly in Fig. 8. We also conducted the same experiment but in trajectory optimization and simulation for comparison. Additionally, in order to maximize the controller robustness for the hardware experiment, we constrained the center of pressure (CoP) close to the foot center during the model optimization stage, although this limits the potential performance gain of the optimal ROM.
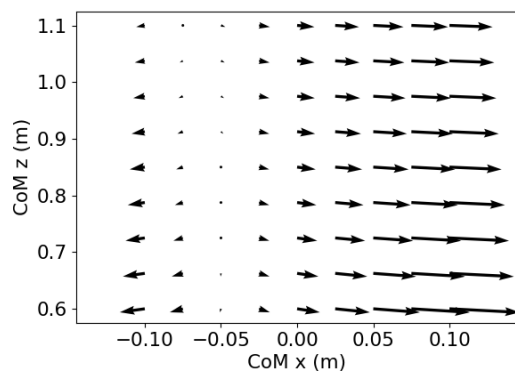
In Fig. 8, we can see that the performance of the optimal ROM (evaluated by Cassie's joint torque squared in this case)

(a) Initial model (R2). 2D slice at CoM position = 0m in y axis (when the CoM is right above the stance foot).
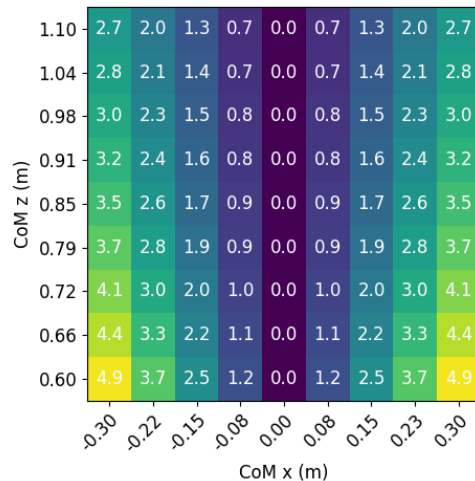

(b) Optimal model (R3). 2D slice at CoM position = 0m in y axis (when the CoM is right above the stance foot).
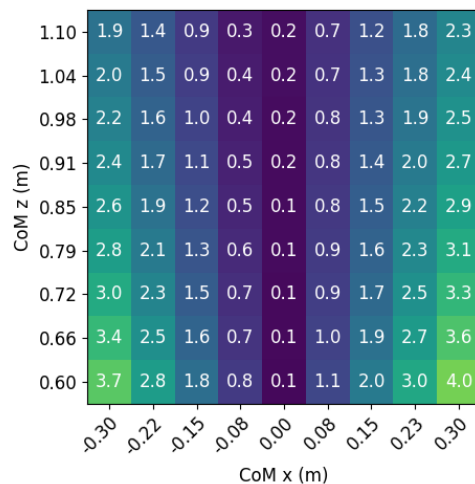

(c) Optimal model (R3). 2D slice at CoM position = -0.2m in y axis (when the CoM is to the right of the left stance foot, for example).

Fig. 9. The vector fields of the ROM dynamics g over the CoM x and z position. In this example, the dynamics is the acceleration of the CoM, which is a function of the CoM position and velocity defined in Eq. (7b). The first plot is the initial model's dynamics, while the latter two are that of the optimal model at two different slices of CoM y position. In all plots, the CoM velocity is 0. We note that the size of the vectors only reflects the relative magnitude. The absolute magnitude of the vectors for the first two plots are shown in Fig. 10, although the scales of the x axis are different.

(a) Initial model (R2). 2D slice at CoM position = 0m in y axis (when the CoM is right above the stance foot).

(b) Optimal model (R3). 2D slice at CoM position = 0m in y axis (when the CoM is right above the stance foot).

Fig. 10. The magnitude of ROM dynamics g over the CoM x and z position. The settings of Fig. 10a and 10b are the same as Fig. 9a and 9b, respectively. The magnitude plots show that the optimal model has smaller CoM accelerations, which implies smaller ground reaction forces, particularly in the x axis (implied by the vector fields in Fig. 9). We observed in the experiments that these force vectors align more closely with the normal direction of the ground.

is better than that of the initial ROM. On hardware, the performance improvement is around 10% for low-speed and medium-speed walking. As a comparison, the cost improves by up to 8% in open-loop and improves by up to 14% in simulation using the same optimal ROM. We can also observe that the hardware costs are higher than those of open-loop and simulation, which might result from additional torques required to track the desired trajectories due to sim-to-real modeling errors. Nonetheless, the improvement percentages are fairly similar across the board. This demonstrates that the model performance was successfully transferred to the hardware via the MPC, despite the modeling error in the full-order model.

E. Optimal Robot Behaviors

In order to understand the source of the performance improvement, we look at the motion of the robot and the center of mass dynamics (the ROM dynamics g). The discussions in this section are based on the straight-line walking experiments in Section V-D.

We observe in the full-model trajectory optimization that the average CoP stays at the center of the support polygon when we use LIP on Cassie (R2). In contrast, the CoP moves toward the rear end of the support polygon where there is no ROM embedding (R1). Interestingly, this CoP shift emerges when using the optimal model (R3) in the open-loop trajectory, and we observe the same behavior in both simulation and hardware experiment. On hardware, we confirm this by visualizing the projected CoM on the ground when Cassie walks in place. The projected CoM is close to CoP, since there is little centroidal angular momentum for walking in place. The projected CoM of the hardware data indeed shifts towards the back of the support polygon when using the optimal model.

To understand why the projected CoM moves backward, we plot the ROM dynamic function g in Fig. 9 for both the initial model and the optimal model. In the case of the initial model (LIP), we know that the dynamics should be symmetrical about the z axis, specifically the acceleration should be 0 at x = 0 (Fig. 9a). This vector field profile, however, looks different in the case of the optimal model. As we can see from Fig. 9b, the area with near-zero acceleration shifts towards the -x direction (i.e. to the back of the support polygon), and interestingly it also slightly correlates to the height of the CoM. The higher the CoM is, the further back the region of zero acceleration is. Additionally, we know that the LIP dynamics in the x-z plane is independent of the CoM y position. That is, no matter which slice of the x-z plane we take, the vector field should look identical. For the optimal model, the dynamics in the x-z plane is a function of the CoM y position. The further away the CoM is from the stance foot (bigger foot spread), the further back the zero acceleration region is.

Aside from the vector field plots of the CoM dynamics, we also visualize the absolute value of the vectors in Fig. 10. We can see that the magnitude of the CoM acceleration in general becomes smaller when using the optimal model. This implies that the total ground reaction force is smaller with the optimal model, even if the robot walks at the same speed. Given the same walking speed, the robot decelerates and accelerates less in the x axis when using the optimal model (i.e., the average speed is the same, but the speed fluctuation becomes smaller after using the optimal model). We hypothesize that the decrease in ground force magnitude partially contributes to the decrease in the joint torque in the case of Cassie walking. The fact that there is less work done on the CoM during walking with the optimal ROM might have led to the decrease in torque squared which is a proxy for energy consumption.

The experiments in Section V demonstrate two things. First, the optimal behavior and the performance are transferred from the open-loop training (left side of Fig. 1) to a closed-loop system (right side of Fig. 1). Second, the optimal reduced-order model improves the real Cassie's performance, while the

low dimensionality permits a real time planning application.

VI. MPC FOR A GENERAL ROM

Sections IV and V limit the ROMs to a predefined embedding function r which simplifies the planner and enables real time planning results. In this section, we present an MPC formulation for a general ROM, where full-order states at swing foot touchdown events were used to provide physical meaning to the resulting plan. Given an ongoing steady improvement in computational and algorithmic speed, we believe this general MPC will soon be solvable in real time on hardware.

A. Hybrid Nature of the Robot Dynamics

Shown in Eq. (2), the dynamics of the full model is hybrid – it contains both the continuous-time dynamics and discrete-time dynamics due to the foot collision. In contrast, many existing reduced-order models assume zero ground impacts at foot touchdown. This is partially due to the fact that the exact embedding of a reduced-order discrete dynamics does not always exist. For example, we could have two pre-impact states x of the full model that correspond to the same reduced-order states, but the post-impact states of the full model map to two different reduced-order states. In this case, the reduced-order discrete dynamics is not an ordinary (single-valued) function. Therefore, in order to capture the exact full impact dynamics in the planner, it is necessary to mix the reduced-order model with the discrete dynamics from the full-order model. We note that the traditional approaches to reduced-order planning and embedding must also grapple with approximations of the impact event.

In addition to the issue above, the mix of reduced- and full-order models also seems necessary if we do not retain the physical interpretability of the embedding r when planning for the optimal footstep locations in the planner. This results in a low-dimensional trajectory optimization problem, a search for $y_j(t)$ and $\tau_j(t)$, with additional decision variables $x_{-,j}, x_{+,j}$, representing the pre- and post-impact full-order states. The index j refers to the j-th stride. The constraints relating the reduced-order state to the full-order model and the impact dynamics are

$$
y_j(t_j) = r(q_{-,j}; \theta_e), \qquad \dot{y}_j(t_j) = \frac{\partial r(q_{-,j}; \theta_e)}{\partial q_{-,j}} \dot{q}_{-,j},
$$

$$
y_{j+1}(t_j) = r(q_{+,j}; \theta_e), \quad \dot{y}_{j+1}(t_j) = \frac{\partial r(q_{+,j}; \theta_e)}{\partial q_{+,j}} \dot{q}_{+,j}, \quad (19)
$$

$$
\text{and} \qquad C_{hybrid}(x_{-,j}, x_{+,j}, \Lambda_j) \leq 0,
$$

where t_j is the impact time (ending the j-th stride), C_{hybrid} represents the hybrid guard S and the impact mapping Δ without left-right leg alternation[6] [2].

B. Planning with ROM and Full-order Impact Dynamics

Similar to Section IV, we formulate a reduced-order trajectory optimization problem to walk n_s strides. However, we replace the discrete footstep inputs T_{fp} with the full

---

[6]The impact mapping Δ can be simplified to identity if we assume no impact (i.e. swing foot touchdown velocity is 0 in the vertical axis).

robot states $x_{-,j}$ and $x_{+,j}$. The key difference between the planning problems in Section IV and this section is that here we introduce new variables

- full-order states $x_{-,j}$, $x_{+,j}$ and ground impulses Λ_j (to capture the exact full-order impact dynamics), and
- the ROM's continuous-time input τ.

To improve readability, we stack decision variables into bigger vectors $\mathrm{T} = [\tau_1, ..., \tau_n] \in \mathbb{R}^{n_\tau n}$ and $X = [x_{-,1}, ..., x_{-,n_s}, x_{+,1}, ..., x_{+,n_s}] \in \mathbb{R}^{2n_x n_s}$.

Costs are nominally expressed in terms of $[y, \dot{y}]$ and τ, though the pre- and post-impact full-order states can also be used to represent goal locations. In addition to the constraints in Eq. (19), we impose constraints $C_{kinematics}(X) \leq 0$ on the full model's kinematics such that the solution obeys joint limits, stance foot stays fixed during the stance phase, and legs do not collide with each other.

The planning problem with the general ROM is

$$\min_{w} \quad \|\mathrm{T}\|_{W_\mathrm{T}}^2 + \|Z - Z_{reg}\|_{W_Z}^2 + \|X - X_{reg}\|_{W_X}^2$$
$$\text{s.t.} \quad \text{Reduced-order dynamics (Eq. (7b)),}$$
$$\text{Hybrid constraints (Eq. (19)),} \qquad (20)$$
$$C_{kinematics}(X) \leq 0,$$
$$x_0 = \text{current feedback full-order state,}$$

where $w = [Z, \mathrm{T}, x_0, X, \Lambda_1, ..., \Lambda_{n_s}] \in \mathbb{R}^{n_w}$, W_T, W_Z and W_X are the weights of the norms, Z_{reg} is the regularization state for the reduced model, and X_{reg} is the regularization state for the full state and contains the goal location of the robot. After solving Eq. (20), we reconstruct the desired ROM trajectory $y_d(t)$ from the optimal solution Z^*. Different from Section IV, the optimal solution w^* here also contains the desired full-order states X^* at impact events, from which we derive not only the desired swing foot stepping locations but also the desired trajectories for joints such as swing hip yaw and swing toe joint. Additionally, since there are full states in the planner, we can send the turning rate command directly to the ROM planner.

Fig. 11 visualizes the pre-impact states in the case where the robot walks two meters with four strides, connected by the hybrid events and continuous low-dimensional trajectories $y_j(t)$. Although there is no guarantee that the planned trajectories $y_j(t)$ are feasible for the full model except those at the hybrid events, we were able to retrieve $q(t)$ from $y_j(t)$ through inverse kinematics, meaning the embedding existed empirically. We note that classical models like LIP also provide no guarantees [76]. For instance, there is no constraint on leg lengths in the ROM which could lead to kinematic infeasibility. The formulation in Eq. (20) preserves an exact representation of the hybrid dynamics, but results in a significantly reduced optimization problem.

C. Implementation and Experiments

We implement the MPC using Eq. (20) for both simulation and hardware experiments (the hardware setup is the same as Section IV-C). In simulation, we were able to transfer the open-loop performance to closed-loop performance with this new MPC. However, on hardware, the off-the-shelf solvers IPOPT and SNOPT were not capable of solving the planning
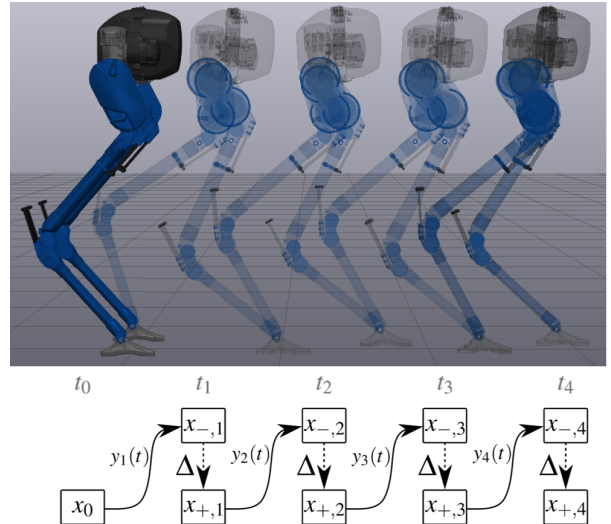

Fig. 11. An example of the real time planner in Eq. (20). Given a task of covering two meters in four steps starting from a standing pose, we rapidly plan a trajectory for the reduced-order model. The high-dimensional model is used to capture the hybrid event at stepping, as illustrated in the diagram.

problem in Eq. (20) fast enough or well enough to enable a high-performance real time MPC. With IPOPT, the planner simply did not run fast enough. With SNOPT, even though the solve time can be decreased down to 30ms with loose optimality tolerance and constraint tolerance, we sacrificed the solution quality too much to achieve big stride length on Cassie. Nonetheless, we believe that the general MPC will soon be solvable within reasonable time constraints on hardware, as computer technology advances steadily. Additionally, a well-engineered custom solver can also help enable real time planning. Boston Dynamics has shown their success in the nonlinear MPC solver with the centroidal momentum model and full model configurations[7] [79].

VII. DISCUSSION

A. Model Parameterization

1) Trade-off between planning speed and performance: A fundamental trade-off exists between planning speed and model performance [27], [29]. As the model (i.e. r and g) becomes more expressive, we see slower planning speeds and better performances. This model expressiveness increases as we

- increase the dimension of the model,
- increase the number of basis functions (or number of neurons in a neural network), or
- use universal function approximators such as monomial basis functions (polynomials) instead of physically-interpretable functions (particular set of functions).

Additionally, under the same model expressiveness, we observe a trade-off between the task space size and the model performance. Larger task spaces require the model to be less

---

[7]The centroidal momentum model [77], [78] describes the actual dynamics without imposing constraints on the full model. However, this momentum model lacks positional representations, thus requiring the incorporation of full model configurations in planning.

specialized, which might result in a lower average performance.

2) Linear models (with linear basis functions): We found that, for some choice of task space, linear models perform almost as well as the full model. This linear reduced-order dynamics transforms the MPC in Eq. (17) into a quadratic optimization problem, allowing for rapid planning. This linear model also renders a closed-form solution and makes it suitable for existing techniques in robust control design and stability analysis. For challenging or complex task spaces, linear basis functions sacrifice significant performance when compared with those of higher degree. We emphasize that our method can be used to optimize models of any chosen degree, and leave such selection to the practitioner.

3) Alternative basis functions: Beside different orders of monomials, we also experimented with trigonometric monomials (e.g. $\sin^a(x)\cos^b(x)$ where $a, b \in \mathbb{N}$). However, we found no notable difference with this basis set. Since quadratic basis approaches optimal performance in model optimization in Section III-E, we leave a broader exploration of choices of basis functions as a possible future work.

4) Physical Interpretability of ROMs: Classical ROMs often maintain some level of physical interperability, because they are built from mechanical components like springs and masses. Our approach, which uses more general representations, does sacrifice this connection to human intuition. However, we have found it beneficial to manually select the embedding function r. This has the benefit of ensuring that the reduced-order state remains human interpretable, which is useful for specifying objectives for planning and control in Section IV. One might also imagine restricting the space of ROM dynamics g to maintain physical connections (e.g. specifying nonlinear or velocity-dependent springs, inspired by human studies [80]), though we leave this to future work.

B. Performance Gap Between Open-loop and Closed-loop

The proposed approach to model optimization uses full-model trajectory optimization. This has a few advantages. First, it allows us to embed the reduced-order model into the full model exactly via constraints. Second, it is more sample efficient than the approaches in reinforcement learning (such as [30]). However, using trajectory optimization leaves a potential performance gap between the offline training and online deployment, because trajectory optimization is an open-loop optimal control method which does not consider any controller heuristics by default. For instance, our walking controller constructs the swing foot trajectory with cubic splines, and we found the open-loop performance can be transferred to the closed-loop better after we add this cubic spline heuristic as a constraint in the trajectory optimization.

While we have seen the performance of the robot improve, we also observed that the solver would exploit any degree of freedom of the input and state variables during the model training stage. For example, the CoP turned out to play an important role in improving the performance (reducing joint torques) of Cassie in Section V. If we had not regularized the CoP (or the CoM motion) during the model optimization stage,

the solver would have moved the CoP all the way to the edge of the support polygon. Although this exploitation can potentially lead to a much bigger cost improvement, it also hurts the robustness. Under sensor noises and model uncertainty, tracking the planned trajectories of this optimal model cannot stabilize the robot well, and thus the performance cannot be transferred to the hardware. One principled way of fixing this issue is to optimize the robustness of the trajectory alongside the user-specified cost function [81], [82].

For the general optimal ROM MPC in Section VI, one place where the performance gap can enter is the choice of the cost function in Eq. (20). In our past experiments, we simply ran a full model trajectory optimization and used this optimal trajectory for the regularization term in Eq. (20). It worked well, although there was a slight improvement drop. To mitigate the gap, one could try inverse optimal control to learn the MPC cost function given data from the full model trajectory optimization.

Since our robot has one-DOF underactuation caused by line feet design, it is not trivial to track the desired trajectory of the ROM within the continuous phase of hybrid dynamics. We noticed in the experiment that there was a noticeable performance difference between whether or not we tracked the first element of the desired ROM trajectory. Therefore, we conjecture that if we use a robot with a finite size of feet, we could translate the open-loop performance to the closed-loop performance better and more easily.

Lastly, we found that empirically it is easier to transfer model performance from open-loop to closed-loop when we fix the embedding function r, although the open-loop performance improvement is usually much bigger (sometimes near full-model's performance) when we optimize both the embedding function r and dynamics g. As a concrete example, in some model optimization instances the optimal ROM position y can be insensitive to the change of CoM position, which makes it difficult to servo the CoM height. In the worst case, this insensitivity could lead to substantial CoM height movement and instability of the closed-loop system.

C. Limitations of Our Framework

In our bilevel optimization approach, the initial ROM must be feasible for the inner-level trajectory optimization to obtain a meaningful gradient for the outer-level optimization. This means that we must initialize the ROM to one capable of walking, potentially limiting our ability to use stochastic initialization to explore the entire ROM space. Despite this potential drawback, we note that random task sampling in Algorithm 1 can help escape certain local minima (effectiveness depends on the cost landscape of the model optimization). Future work could explore the role of this initialization, for instance by evaluating performance when starting from multiple existing hand-designed ROMs.

Our approach requires the user to determine the dimension of the ROM. Increasing the dimension theoretically strictly improves model performance, at the cost of MPC computational speed. As a result, this defines a Pareto optimal front, without a simple way to automatically determine the dimension. That said, there are recent works which attempt to select between

models of varying complexity [27]–[29], which we believe might be applied to our framework.

D. Generality of Our Framework

This paper focuses on applying the optimal ROMs to the hardware Cassie, but throughout the project we observe that LIP performs reasonable well for Cassie, particularly over relatively simple task domains such as straight-line walking. We hypothesize that this is due, in part, to the fact that Cassie's legs are relatively light. As an experiment, we investigated the effect of foot weight on the performance improvement. When increasing the foot's mass to 4kg (the robot weighs 40kg in total), we observed that the LIP cost relative to the full model's increased from 1.3 to 1.8 and offered a greater room for improvement, resulting in 40% torque cost reduction for tasks similar to Example 1's. Beside this investigation, we also saw more than 75% of cost reduction for the five-link planar robot in our prior work [33].

Furthermore, the proposed framework is agnostic to types of robots and tasks (e.g. quadrupeds and dexterous manipulators). This has implications all over robotics, given the need for computational efficiency and the prevalence of reduced-order models in locomotion and manipulation.

VIII. Conclusion

In this work, we directly optimized the reduced-order models which can be used in an online planner that achieved performance higher than that of the traditional physical models. We formulated a bilevel optimization problem and presented an efficient algorithm that leverages the problem structure. Examples showed improvements up to 38% depending on the task difficulty and the performance metric. The optimal reduced-order models are more permissive and capable of higher performance, while remaining low dimensional. We also designed two MPCs for the optimal reduced-order models which enable Cassie to accomplish tasks with better performance. In the hardware experiment, the optimal ROM showed 10% of improvement on Cassie, and we investigated the source of performance gains for this particular model. We demonstrated that the use of ROM greatly reduces planning time, and that the optimized ROM improves the performance of the robot beyond the traditional ROMs.

Although the model optimization approach presented in this paper has the advantage of optimizing models agnostic to low-level controllers, it does not guarantee that the performance improvements from these optimal models can be transferred to the robot via a feedback controller as discussed in Section VII-B. One ongoing work is to fix the above issue by optimizing the model in a closed-loop fashion, so that the model optimization accounts for the controller heuristics and maintains the closed-loop stability. This would also potentially ease the process of realizing the optimal model performance on hardware. Additionally, discussed in Section VI-A, an approximation is necessary if we were to find a low dimensional representation of the full-order impact dynamics. In this work, we only circumvented the hybrid problem by either using a physically-interpretable ROM or mixing the full impact dynamics with the ROM. Finding an optimal low-dimensional discrete dynamics for a robot still remains an open question.

Appendix A
Heuristics in Trajectory Optimization

Solving the trajectory optimization problem in Eq. (4) or (TO) for a high-dimensional robot is hard, since the problem is nonlinear and of large scale. Even although there are off-the-shelf solvers such as IPOPT [75] and SNOPT [70] designed to solve large-scale nonlinear optimization problem, it is often impossible to get a good optimal solution without any heuristics, since there are many local optima. In this section, we will talk from our experience about the heuristics that might help to solve the problem faster and also find a solution with a lower cost and closer to the global optimum. That said, we have no objective manner in which to assess proximity to global optimality, and thus this is a purely observational criterion.

Let the nonlinear problem be

$$\begin{aligned} \min_{w} \quad & \tilde{h}(w) \\ \text{s.t.} \quad & \tilde{f}(w) \le 0 \end{aligned} \tag{21}$$

where w contains all decision variables, $\tilde{h}$ is the cost function, and $\tilde{f}$ is the constraint function. It turned out that scaling either w, $\tilde{h}$ or $\tilde{f}$ could help to improve the condition of the problem.

- w: Sometimes the decision variables are in different units and can take values of different orders. For example, joint angles of Cassie are roughly less than 1 (rad), while its contact forces are usually larger than 100 (N). In this case, we can scale w by some factor s, such that the decision variables of the new problem are $w_{scaled,i} = s_i w_i$ for $i = 1, 2, ..., n_w$. After the problem is solved, we scale the optimal solution of the new problem back by $w_i^* = \frac{1}{s_i} w_{scaled,i}^*$.
- $\tilde{f}$: The constraints $\tilde{f}$ can take various units just like w. Similarly, we can scale each constraint individually. Note that scaling constraints affects how well the original constraints are satisfied, so one should make sure that the constraint tolerance is still meaningful.
- $\tilde{h}$: In theory, scaling the cost does not affect the optimal solution. However, it does matter in the solver's algorithm. It is desirable to scale the cost so that it is not larger than 1 around the area of interest.

For more detail about scaling, we refer the readers to Chapter 8.4 and Chapter 8.7 of [83]. In addition to scaling the problem, the following heuristics could also be helpful:

- Provide the solver with a good initial guess.
- Add small randomness to the initial guess: This helps to avoid singularities.
- Add regularization terms to the cost function: This could remove local minima in the cost landscape and can also speed up the solve time. Adding regularization terms is similar to the traditional reward shaping of Reinforcement Learning [84] and the policy-regularized MPC [85].

- Add intermediate variables (also called slack variables [86]): This can sometimes improve the condition number of the constraint gradients with respect to decision variables. One example of this is reformulating the trajectory optimization problem based on the single shooting method into that based on the multiple shooting method by introducing state variables [87].
- Use solver's internal scaling option: In the case of SNOPT [88], we found setting *Scale option* to 2 helps to find an optimal solution of better quality. Note that this option increases the solve time and demands a good initial guess to the problem.

APPENDIX B
MIRRORED REDUCED-ORDER MODEL

A. Model definition

The model representation in Eq. (7) could be dependent on the side of the robot. For example, when using the LIP model as the ROM, we might choose the generalized position of the model y to be the CoM position relative to the left foot (instead of the right foot). In this case, we need to find the reduced-order model for the right support phase of the robot. Fortunately, we can derive this ROM by mirroring the robot configuration q about its sagittal plane (Fig. 12) and reusing the ROM of the left support phase. We refer to this new ROM as the *mirrored reduced-order model*.

Let q_m and v_m be the generalized position and velocity of the "mirrored robot", shown in Fig. 12. Mathematically, the mirrored ROM μ_m is

$$\mu_m \triangleq (r_m, g), \tag{22}$$

with

$$y_m = r_m(q) = r(q_m) = r(M(q)), \tag{23a}$$
$$\ddot{y}_m = g(y_m, \dot{y}_m, \tau), \tag{23b}$$

where r_m is the embedding function of the mirror model, and M is the mirror function such that $q_m = M(q)$ and $q = M(q_m)$. We note that the two models, in Eq. (6) and (22), share the same dynamics function g.

B. Time derivatives of the embedding function

Feedback control around a desired trajectory often requires the first and the second time derivatives information. Here, we derive these quantities for the mirrored ROM in terms of the original embedding function r in Eq. (7a) and its derivatives.

1) $\dot{y}_m$: Let J_m be the Jacobian of the mirrored model embedding r_m with respect to the robot configuration q, such that

$$\dot{y}_m = J_m \dot{q}. \tag{24}$$

The time derivatives of y_m is

$$\dot{y}_m = \frac{\partial r(M(q))}{\partial M(q)} \frac{\partial M(q)}{\partial q} \dot{q} = J(q_m) \frac{\partial M(q)}{\partial q} \dot{q} = J(q_m) \dot{q}_m \tag{25}$$

where $J(q_m)$ is the Jacobian of the original model embedding r evaluated with q_m. From Eq. (24) and (25), we derive

$$J_m = J(q_m) \frac{\partial M(q)}{\partial q} \tag{26}$$

where $\frac{\partial M(q)}{\partial q}$ is a matrix which contains only 0, 1, and -1.
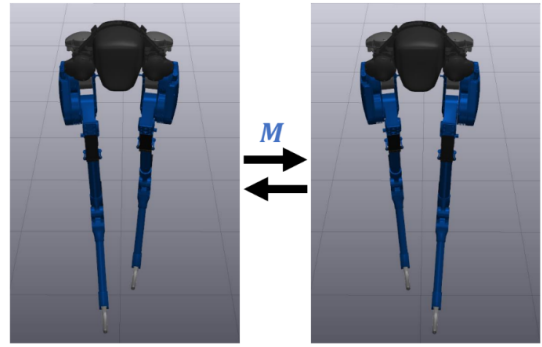

Fig. 12. The mirror function M mirrors the robot configuration about the sagittal plane. This function is necessary in planning and control when the embedding function (Eq. (7a)) of the reduced-order model only represents one side of the robot. For example, the embedding function of the LIP model was chosen to be the CoM relative to the left foot (and not the right foot).

2) $\ddot{y}_m$: The i-th element of $\ddot{y}_m$ is

$$\ddot{y}_{m,i} = \frac{d}{dt} \left(\frac{\partial r_i(q_m)}{\partial q_m} \dot{q}_m \right) = \frac{d}{dt} \sum_j \frac{\partial r_i(q_m)}{\partial q_{m,j}} \dot{q}_{m,j}$$

$$= \sum_j \frac{d}{dt} \left(\frac{\partial r_i(q_m)}{\partial q_{m,j}} \right) \dot{q}_{m,j} + \sum_j \frac{\partial r_i(q_m)}{\partial q_{m,j}} \frac{d}{dt} \left(\dot{q}_{m,j} \right)$$

$$= \sum_{jk} \frac{\partial^2 r_i(q_m)}{\partial q_{m,j} \partial q_{m,k}} \dot{q}_{m,j} \dot{q}_{m,k} + \frac{\partial r_i(q_m)}{\partial q_m} \frac{d}{dt} \left(\dot{q}_m \right).$$

where r_i is the i-th element of the embedding function. The above equation can be expressed in the vector-matrix form

$$\begin{aligned} \ddot{y}_m &= \dot{q}_m^T \nabla^2 r(q_m) \dot{q}_m + J(q_m) \ddot{q}_m \\ &= \dot{J}(q_m, v_m) \dot{q}_m + J(q_m) \ddot{q}_m \\ &= \dot{J}(q_m, v_m) \dot{q}_m + J_m \dot{v} \quad (\because \text{Eq. (26)}) \end{aligned} \tag{27}$$
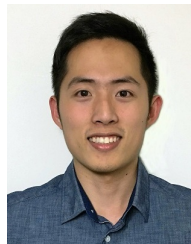
where $\dot{J}(q_m, v_m)$ is the time derivatives of the J (of the original model) evaluated with the mirrored position q_m and velocity v_m.

REFERENCES

[1] P. M. Wensing, M. Posa, Y. Hu, A. Escande, N. Mansard, and A. Del Prete, "Optimization-based control for dynamic legged robots," *IEEE Transactions on Robotics*, 2023.

[2] 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.

[3] H. Zhao, A. Hereid, W.-l. Ma, and A. D. Ames, "Multi-contact bipedal robotic locomotion," *Robotica*, vol. 35, no. 5, pp. 1072–1106, 2017.

[4] 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.

[5] 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.

[6] 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.

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

[8] 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.

[9] X. Xiong, J. Reher, and A. D. Ames, "Global position control on underactuated bipedal robots: Step-to-step dynamics approximation for step planning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2825–2831, IEEE, 2021.

[10] M. Kasaei, A. Ahmadi, N. Lau, and A. Pereira, "A robust model-based biped locomotion framework based on three-mass model: From planning to control," in *2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pp. 257–262, IEEE, 2020.

[11] T. Takenaka, T. Matsumoto, and T. Yoshiike, "Real time motion generation and control for biped robot-1 st report: Walking gait pattern generation," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1084–1091, IEEE, 2009.

[12] T. Sato, S. Sakaino, and K. Ohnishi, "Real-time walking trajectory generation method with three-mass models at constant body height for three-dimensional biped robots," *IEEE transactions on industrial electronics*, vol. 58, no. 2, pp. 376–383, 2010.

[13] S. Shimmyo, T. Sato, and K. Ohnishi, "Biped walking pattern generation by using preview control based on three-mass model," *IEEE transactions on industrial electronics*, vol. 60, no. 11, pp. 5137–5147, 2012.

[14] S. M. Kasaei, N. Lau, and A. Pereira, "A reliable hierarchical omnidirectional walking engine for a bipedal robot by using the enhanced lip plus flywheel," in *Human-Centric Robotics: Proceedings of CLAWAR 2017: 20th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines*, pp. 399–406, World Scientific, 2018.

[15] S. Faraji and A. J. Ijspeert, "3lp: A linear 3d-walking model including torso and swing dynamics," *the international journal of robotics research*, vol. 36, no. 4, pp. 436–455, 2017.

[16] 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.

[17] A. Shaiju and I. R. Petersen, "Formulas for discrete time lqr, lqg, leqg and minimax lqg optimal control problems," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 8773–8778, 2008.

[18] C. E. Garcia, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice—a survey," *Automatica*, vol. 25, no. 3, pp. 335–348, 1989.

[19] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[20] Y. Gong and J. W. Grizzle, "Zero dynamics, pendulum models, and angular momentum in feedback control of bipedal locomotion," *Journal of Dynamic Systems, Measurement, and Control*, vol. 144, no. 12, p. 121006, 2022.

[21] 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.

[22] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body Motion Planning with Centroidal Dynamics and Full Kinematics," *IEEE-RAS International Conference on Humanoid Robots*, 2014.

[23] A. Herzog, S. Schaal, and L. Righetti, "Structured contact force optimization for kino-dynamic motion generation," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2703–2710, IEEE, 2016.

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

[25] M. Posa, T. Koolen, and R. Tedrake, "Balancing and Step Recovery Capturability via Sums-of-Squares Optimization," in *Robotics: Science and Systems*, 2017.

[26] T. Koolen, M. Posa, and R. Tedrake, "Balance control using center of mass height variation: limitations imposed by unilateral contact," in *Humanoid Robots (Humanoids), 2016 IEEE-RAS 16th International Conference on*, pp. 8–15, IEEE, 2016.

[27] H. Li, R. J. Frei, and P. M. Wensing, "Model hierarchy predictive control of robotic systems," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3373–3380, 2021.

[28] C. Khazoom, S. Heim, D. Gonzalez-Diaz, and S. Kim, "Optimal scheduling of models and horizons for model hierarchy predictive control," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9952–9958, IEEE, 2023.

[29] J. Norby, A. Tajbakhsh, Y. Yang, and A. M. Johnson, "Adaptive complexity model predictive control," *arXiv preprint arXiv:2209.02849*, 2022.

[30] 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.

[31] Y.-M. Chen, G. Nelson, R. Griffin, M. Posa, and J. Pratt, "Integrable whole-body orientation coordinates for legged robots," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 10440–10447, IEEE, 2023.

[32] K. Yamane, "Systematic derivation of simplified dynamics for humanoid robots," in *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, pp. 28–35, IEEE, 2012.

[33] 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.

[34] R. J. Full and D. E. Koditschek, "Templates and anchors: neuromechanical hypotheses of legged locomotion on land," *The Journal of Experimental Biology*, vol. 202, pp. 3325–3332, 1999.

[35] M. H. Raibert, H. B. Brown, and M. Chepponis, "Experiments in Balance with a 3D One-Legged Hopping Machine," *The International Journal of Robotics Research*, vol. 3, pp. 75–92, jun 1984.

[36] M. Garcia, A. Chatterjee, A. Ruina, and M. Coleman, "The Simplest Walking Model: Stability, Complexity, and Scaling," *Journal of Biomechanical Engineering*, vol. 120, p. 281, apr 1998.

[37] A. L. Schwab and M. Wisse, "Basin of attraction of the simplest walking model," in *ASME 2001 Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 2001.

[38] K. Byl and R. Tedrake, "Metastable Walking Machines," *The International Journal of Robotics Research*, vol. 28, pp. 1040–1064, aug 2009.

[39] C. Oguz Saglam and K. Byl, "Robust Policies via Meshing for Metastable Rough Terrain Walking," in *Robotics: Science and Systems*, 2014.

[40] M. Kelly and A. Ruina, "Non-linear robust control for inverted-pendulum 2D walking," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4353–4358, IEEE, may 2015.

[41] 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.

[42] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, "Capture point: A step toward humanoid push recovery," in *2006 6th IEEE-RAS international conference on humanoid robots*, pp. 200–207, IEEE, 2006.

[43] B. Wie and P. M. Barba, "Quaternion feedback for spacecraft large angle maneuvers," *Journal of Guidance, Control, and Dynamics*, vol. 8, no. 3, pp. 360–365, 1985.

[44] R. Tedrake and the Drake Team Development, "Drake: A planning, control, and analysis toolbox for nonlinear dynamical systems," 2016.

[45] Y. Hurmuzlu and D. B. Marghitu, "Rigid body collisions of planar kinematic chains with multiple contact points," *The international journal of robotics research*, vol. 13, no. 1, pp. 82–92, 1994.

[46] J. W. Grizzle, C. Chevallereau, R. W. Sinnet, and A. D. Ames, "Models, feedback control, and open problems of 3d bipedal robotic walking," *Automatica*, vol. 50, no. 8, pp. 1955–1988, 2014.

[47] A. Hereid, O. Harib, R. Hartley, Y. Gong, and J. W. Grizzle, "Rapid bipedal gait design using c-frost with illustration on a cassie-series robot," *arXiv preprint arXiv:1807.06614*, 2018.

[48] J. Reher, W.-L. Ma, and A. D. Ames, "Dynamic walking with compliance on a cassie bipedal robot," in *2019 18th European Control Conference (ECC)*, pp. 2589–2595, IEEE, 2019.

[49] J. T. Betts, *Practical Methods for Optimal Control Using Nonlinear Programming*. SIAM Advances in Design and Control, Society for Industrial and Applied Mathematics, 2001.

[50] M. Posa, C. Cantu, and R. Tedrake, "A Direct Method for Trajectory Optimization of Rigid Bodies Through Contact," *The International Journal of Robotics Research*, vol. 33, pp. 69–81, jan 2013.

[51] M. Posa, S. Kuindersma, and R. Tedrake, "Optimization and stabilization of trajectories for constrained dynamical systems," in *2016 IEEE International Conference on Robotics and Automation*, vol. 2016-June, (Stockholm, Sweden), pp. 1366–1373, may 2016.

[52] J. Bracken and J. T. McGill, "Mathematical programs with optimization problems in the constraints," *Operations Research*, vol. 21, no. 1, pp. 37–44, 1973.

[53] L. Franceschi, P. Frasconi, S. Salzo, R. Grazzi, and M. Pontil, "Bilevel programming for hyperparameter optimization and meta-learning," in *International Conference on Machine Learning*, pp. 1568–1577, PMLR, 2018.

[54] A. Rajeswaran, I. Mordatch, and V. Kumar, "A game theoretic framework for model based reinforcement learning," in *International conference on machine learning*, pp. 7953–7963, PMLR, 2020.

[55] W. Jin, T. D. Murphey, D. Kulić, N. Ezer, and S. Mou, "Learning from sparse demonstrations," *IEEE Transactions on Robotics*, 2022.

[56] S. Pfrommer, M. Halm, and M. Posa, "Contactnets: Learning discontinuous contact dynamics with smooth, implicit representations," in *Conference on Robot Learning*, pp. 2279–2291, PMLR, 2021.

[57] A. Sinha, P. Malo, and K. Deb, "A review on bilevel optimization: from classical to evolutionary approaches and applications," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 2, pp. 276–295, 2017.

[58] K. Hatz, J. P. Schloder, and H. G. Bock, "Estimating parameters in optimal control problems," *SIAM Journal on Scientific Computing*, vol. 34, no. 3, pp. A1707–A1728, 2012.

[59] C. Shi, J. Lu, and G. Zhang, "An extended kuhn–tucker approach for linear bilevel programming," *Applied Mathematics and Computation*, vol. 162, no. 1, pp. 51–63, 2005.

[60] H. W. Kuhn and A. W. Tucker, "Nonlinear programming," in *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, (Berkeley, Calif.), pp. 481–492, University of California Press, 1951.

[61] W. Jin, Z. Wang, Z. Yang, and S. Mou, "Pontryagin differentiable programming: An end-to-end learning and control framework," *Advances in Neural Information Processing Systems*, vol. 33, pp. 7979–7992, 2020.

[62] J. Domke, "Generic methods for optimization-based modeling," in *Artificial Intelligence and Statistics*, pp. 318–326, PMLR, 2012.

[63] N. Das, S. Bechtle, T. Davchev, D. Jayaraman, A. Rai, and F. Meier, "Model-based inverse reinforcement learning from visual demonstrations," in *Conference on Robot Learning*, pp. 1930–1942, PMLR, 2021.

[64] S. G. Krantz and H. R. Parks, *The implicit function theorem: history, theory, and applications*. Springer Science & Business Media, 2002.

[65] S. Afriat, "Theory of maxima and the method of lagrange," *SIAM Journal on Applied Mathematics*, vol. 20, no. 3, pp. 343–357, 1971.

[66] A. Takayama and T. Akira, *Mathematical economics*. Cambridge university press, 1985.

[67] W. Jin, S. Mou, and G. J. Pappas, "Safe pontryagin differentiable programming," *Advances in Neural Information Processing Systems*, vol. 34, pp. 16034–16050, 2021.

[68] J. G. Riley, *Essential microeconomics*. Cambridge University Press, 2012.

[69] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[70] P. E. Gill, W. Murray, and M. A. Saunders, "Snopt: An sqp algorithm for large-scale constrained optimization," *SIAM review*, vol. 47, no. 1, pp. 99–131, 2005.

[71] Y. Gong, R. Hartley, X. Da, A. Hereid, O. Harib, J.-K. Huang, and J. Grizzle, "Feedback control of a cassie bipedal robot: Walking, standing, and riding a segway," in *2019 American Control Conference (ACC)*, pp. 4559–4566, IEEE, 2019.

[72] 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.

[73] 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.

[74] A. S. Huang, E. Olson, and D. C. Moore, "Lcm: Lightweight communications and marshalling," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4057–4062, IEEE, 2010.

[75] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[76] A. Iqbal, S. Veer, and Y. Gu, "Drs-lip: Linear inverted pendulum model for legged locomotion on dynamic rigid surfaces," *arXiv preprint arXiv:2202.00151*, 2022.

[77] D. E. Orin and A. Goswami, "Centroidal momentum matrix of a humanoid robot: Structure and properties," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 653–659, IEEE, 2008.

[78] P. M. Wensing and D. E. Orin, "Improved computation of the humanoid centroidal dynamics and application for whole-body control," *International Journal of Humanoid Robotics*, vol. 13, no. 01, p. 1550039, 2016.

[79] "2022 icra legged robots wokshop presentation by robin deits." https://youtu.be/yagQG_b_hfs. Accessed: 2022-10-24.

[80] G. Pequera, V. Yelós, and C. Biancardi, "Reducing cost of transport in asymmetrical gaits: lessons from unilateral skipping," *European Journal of Applied Physiology*, vol. 123, no. 3, pp. 623–631, 2023.

[81] H. Dai and R. Tedrake, "Optimizing Robust Limit Cycles for Legged Locomotion on Unknown Terrain," in *Proceedings of the IEEE Conference on Decision and Control*, (Maui, Hawaii), p. 8, dec 2012.

[82] J. Zhu, N. J. Kong, G. Council, and A. M. Johnson, "Hybrid event shaping to stabilize periodic hybrid orbits," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 01–07, IEEE, 2022.

[83] P. E. Gill, W. Murray, and M. H. Wright, "Practical optimization (book)," *London and New York, Academic Press, 1981. 415 p*, 1981.

[84] Y. Hu, W. Wang, H. Jia, Y. Wang, Y. Chen, J. Hao, F. Wu, and C. Fan, "Learning to utilize shaping rewards: A new approach of reward shaping," *Advances in Neural Information Processing Systems*, vol. 33, pp. 15931–15941, 2020.

[85] G. Bledt, P. M. Wensing, and S. Kim, "Policy-regularized model predictive control to stabilize diverse quadrupedal gaits for the mit cheetah," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4102–4109, IEEE, 2017.

[86] A. Hereid and A. D. Ames, "Frost: Fast robot optimization and simulation toolkit," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 719–726, IEEE, 2017.

[87] J. T. Betts, "Survey of numerical methods for trajectory optimization," *Journal of guidance, control, and dynamics*, vol. 21, no. 2, pp. 193–207, 1998.

[88] E. Philip, W. Murray, and M. A. Saunders, "User's guide for snopt version 7: Software for large-scale nonlinear programming," 2015.

Yu-Ming Chen is currently an applied scientist at The AI Institute. His research focuses on reduced-order modeling of legged robots with applications in real time planning and control. He holds a B.Sc. in Physics from National Taiwan University (2015), an M.Sc. in Robotics from the University of Michigan, Ann Arbor (2018) and a Ph.D. in Electrical and Systems Engineering from the University of Pennsylvania (2023).

Jianshu Hu is currently a PhD student in UM-SJTU Joint Institute, Shanghai Jiao Tong University. He is working on Deep Reinforcement Learning supervised by Professor Yutong Ban and Professor Paul Weng. Aiming at building an intelligent robot, he is focusing on designing sample-efficient Reinforcement Learning algorithms leveraging invariance/equivariance and data augmentation. Please check the personal website (https://jianshu-hu.github.io) for more details about his research.

Michael Posa (Member, IEEE) received the B.S. and M.S. degrees in mechanical engineering from Stanford University, Stanford, CA, USA, in 2007 and 2008, respectively, and the Ph.D. degree in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2017.,He is currently an Assistant Professor of mechanical engineering and applied mechanics with the University of Pennsylvania, Philadelphia, PA, USA, where he is a Member of the General Robotics, Automation, Sensing and Perception (GRASP) Lab. He holds secondary appointments in electrical and systems engineering and in computer and information science. He leads the Dynamic Autonomy and Intelligent Robotics Lab, University of Pennsylvania, which focuses on developing computationally tractable algorithms to enable robots to operate both dynamically and safely as they maneuver through and interact with their environments, with applications including legged locomotion and manipulation.,Dr. Posa was a recipient of multiple awards, including the NSF CAREER Award, RSS Early Career Spotlight Award, and Best Paper Awards. He is an Associate Editor for IEEE Transactions on Robotics.