Koopman Operator Based Linear Model Predictive Control for 2D Quadruped Trotting, Bounding, and Gait Transition

Chun-Ming Yang and Pranav A. Bhounsule

***Abstract*— Online optimal control of quadrupedal robots would enable them to plan their movement in novel scenarios. Linear Model predictive control (LMPC) has emerged as a practical approach for real-time control. In LMPC, an optimization problem with a quadratic cost and linear constraints is formulated over a finite horizon and solved on the fly. However, LMPC relies on linearizing the equations of motion (EOM), which may lead to poor solution quality. In this paper, we use Koopman operator theory and the extended Dynamic Mode Decomposition (DMD) to create a linear model of the system in high dimensional space, thus retaining the nonlinearity of the EOM. We model the aerial phase and ground contact phases using different linear models. Then, using LMPC, we demonstrate bounding, trotting, and bound-to-trot and trot-to-bound gait transitions in level and rough terrain. The main novelty is the use of Koopman operator theory to create hybrid models of a quadrupedal system and demonstrate the online generation of multiple gaits and gait transitions.**

I. Introduction

Among legged systems, quadrupedal robots have emerged as a ubiquitous platform for designing and testing radically novel control ideas. A dominant approach is to perform online optimal control using a linearized model (e.g., model predictive control) to enable real-time adaptation. However, the linearization may introduce model artifacts that could lead to poor performance. This paper addresses the limitation by using Koopman operator theory to build linear models in high dimensional space that preserves the system's nonlinearity. Furthermore, using linear model predictive control, we demonstrate multiple gaits (bounding and trotting) and gait transitions.

The earliest work investigated simple heuristics, such as regulating foot placement for speed control and vertical force for height control, and applied it to bounding, trotting, and pronking for a quadruped [1]. However, such controllers need to be manually tuned to be robust, which is often time-consuming and not generalizable to other systems A formal method is to compute the control necessary to generate periodic gait, also known as the Poincaré limit cycle [2], and use eigenvalues to determine stability [3]. Such stability analyses have limited utility because they only consider small perturbations along the periodic gaits.

The development effective optimization tools has led to optimal control using Model Predictive Control (MPC). For instance, a quadruped bounding gait was achieved by deriving control inputs from a hierarchical nonlinear MPC [4]. However, this was limited to offline computation due to the high computational time needed for the optimization. Online optimal control can be achieved by solving a short-horizon optimal control problem with a quadratic cost and linear constraints, also known as linear model predictive control (LMPC) [5]. Here, the speed-up is achieved because the optimization is a convex quadratic program, which leads to an analytical solution. However, simplicity comes at a cost: the linear constraints are obtained by linearizing the equation of motion, which may lead to poor convergence when the system deviates far from the linearization. Another issue is that such optimization assumes a pre-determined timing which may not hold for bounding gaits and in rough terrain [6]. A solution to the latter issue is to add gaits and foothold timing as optimization variables to achieve versatile quadruped gaits, where a single optimization process plans the gait sequence, step timings, and footholds [7]. But such optimizations are challenging to solve in real-time.

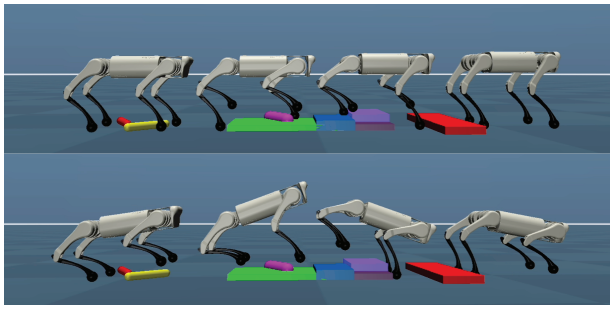

Fig. 1. Trotting (top) and bounding (bottom) on rough terrain

Deep Reinforcement Learning (DRL) is a model-free method that solves an optimization problem over the entire state space. Here, using an offline optimization, a neural network learns the mapping between sensors and actuator commands. Then during online deployment, the neural network outputs an actuator command based on the sensor value DRL has been demonstrated on trotting [8], bounding [9], pacing [10], pronking [11], and the transitions between each gait [12]. However, DRL is not sample efficient as it requires an extensive dataset for training. To achieve sample efficiency, one could use model-based optimization to plan a reference motion and model-free RL to track the reference motion [13].

The Koopman operator takes a nonlinear model $\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t)$ and converts it into a linear model in high dimensional space: $\mathbf{\Pi}(\mathbf{x}_{t+1}) = \mathbf{A}\mathbf{\Pi}(\mathbf{x}_t) + \mathbf{B}\mathbf{u}_t$, where $\mathbf{A}, \mathbf{B}$ are constant matrices and $\mathbf{\Pi}(\mathbf{x})$ is a non-linear function of $\mathbf{x}$. The original method was conceptualized for an uncontrolled

Dept. of Mechanical and Industrial Engineering, University of Illinois at Chicago, 842 W Taylor St, Chicago, IL 60607, USA. Email: jyang241@uic.edu, pranav@uic.edu The work was supported by NSF grant 2128568.

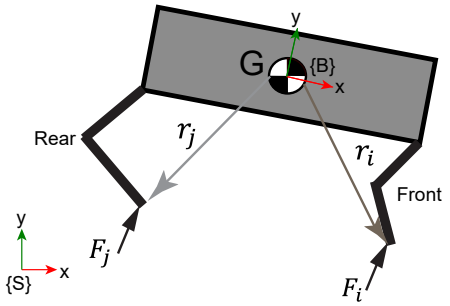

Fig. 2. 2D single rigid body model

system in 1931 by Koopman [14], but only recently tools have been devised to model controlled systems [15]. The applications of the Koopman operator are currently limited to a few simple smooth systems, such as quadcopters [16], underwater vehicles [17], autonomous cars [18], two-link planar manipulators [19], and soft robot manipulators [20].

In this paper, we use the Koopman operator to create multiple linear models that capture the aerial phase and ground contact phases. Although the models are linear, they capture the non-linearity by projecting in a high-dimensional space. Next, we use Linear Model Predictive Control (LMPC) to demonstrate bounding and trotting gaits and bound-to-trot and trot-to-bound gait transitions. Furthermore, we demonstrate gait robustness by introducing rough terrain. The main novelty is the use of Koopman operator theory to model and control hybrid systems such as a quadrupedal robot.

II. Methods

A. Quadruped Gaits Pattern

Trotting and bounding gaits are simple in that the Front Right (FR), Front Left (FL), Rear Right (RR), and Rear Left (RL) legs are used in pairs. In trotting, the legs move in diagonal pairs FR-RL and FL-RR alternating between swing and stance phases. In bounding, the front legs FR-FL and rear legs RL-RR move as pairs, transitioning through a periodic cycle that includes front stance, flight phase, rear stance, and flight phase.

B. Single Rigid Body Model (SRB)

The quadruped system shown in Fig. 2 is modeled using SRB model [5]

$$\ddot{\mathbf{p}} = \frac{\mathbf{R}(\alpha_i \mathbf{F}_i + \alpha_j \mathbf{F}_j)}{m} - \mathbf{g} \tag{1}$$

$$\ddot{\theta} = \frac{\mathbf{r}_i \times \alpha_i \mathbf{F}_i + \mathbf{r}_j \times \alpha_j \mathbf{F}_j}{I} \tag{2}$$

where $\mathbf{p} \in \mathbb{R}^2$ and $\dot{\mathbf{p}} \in \mathbb{R}^2$ are the SRB center of mass (CoM) position and velocity both in the world frame respectively; θ is the pitch angle in world frame, and $\dot{\theta}$ is pitch angular velocities in body frame. The leg ground reaction force for the foot i, j is $\mathbf{F}_{i,j} \in \mathbb{R}^2$ and the distance from the foot i, j to the center of mass is $\mathbf{r}_{i,j} \in \mathbb{R}^2$; α_i, α_j is the scalar to do the model selection, where $\alpha_i = 1, \alpha_j = 1$ for trotting model with front and rear foots contact on the ground, $\alpha_i = 0, \alpha_j = 2$ for rear stance bounding model with only rear foots contact on the ground, $\alpha_i = 2, \alpha_j = 0$ for front stance bounding model with only front foot contact on the ground, and $\alpha_i = 0, \alpha_j = 0$ for flight phase; $\mathbf{R} \in \mathbb{R}^{2\times 2}$ is the rotation matrix mapping vector from body frame to global frame; m is the mass of the robot; $\mathbf{g} \in \mathbb{R}^2$ is the gravity vector in the world frame; I is the inertia matrix of the torso in the body frame. Note that Eqns. 1-2 are nonlinear and they can be compactly written as: $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\mathbf{u}$, where $\mathbf{x} = [\mathbf{p}, \theta, \dot{\mathbf{p}}, \dot{\theta}]^\top$ and $\mathbf{u} = [\mathbf{F}_i, \mathbf{F}_j]^\top$.

C. Koopman Operator Theory

For a given nonlinear system $\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t)$; $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^n$; $\mathbf{u} \in \mathcal{U} \subseteq \mathbb{R}^m$; $\mathbf{f} : \mathcal{X} \to \mathcal{X}$, a set of nonlinear observable functions $\mathbf{\Pi}(\mathbf{x})$ exists such that the evolution of the system along these observables is characterized by linear dynamics governed by an infinite dimension operator $\mathcal{K}$, known as Koopman operator

$$[\mathcal{K}\mathbf{\Pi}](\mathbf{x}, \mathbf{u}) = \mathbf{\Pi} \circ \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{3}$$

A finite-dimensional approximation of $\mathcal{K}$, denoted as $\mathbf{K} = [\mathbf{A}, \mathbf{B}]; \mathbf{A} \in \mathbb{R}^{N\times N}; \mathbf{B} \in \mathbb{R}^{N\times m}$, is derived by employing the Extended Dynamic Mode Decomposition (EDMD) approach [15], which projects $\mathcal{K}$ onto a subspace of observable functions via least squares regression. The finite dimension approximated operator $\mathbf{K}$ can be used to represent the linear evolution of observable functions as

$$\mathbf{\Pi}(\mathbf{x}_{t+1}) = \begin{bmatrix}\mathbf{A} & \mathbf{B}\end{bmatrix} \begin{bmatrix}\mathbf{\Pi}(\mathbf{x}_t) \\ \mathbf{u}_t\end{bmatrix} = \mathbf{K}\hat{\mathbf{\Pi}}(\mathbf{x_t}, \mathbf{u_t}) \tag{4}$$

where $\mathbf{\Pi}(\mathbf{x}_t) = [\pi_1(\mathbf{x}_t), ..., \pi_N(\mathbf{x}_t)]^T \in \mathbb{R}^N$ is the dictionary of observable functions. By utilizing M snapshots of the system states and control inputs in lifted space formed by the basis function $\pi_i(\mathbf{x})$ in the dictionary, we obtain the approximated operator $\mathbf{K}$. The paired dataset $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{M-1}]$ and $\mathbf{Y} = [\mathbf{x}_2, \mathbf{x}_3, ..., \mathbf{x}_M]$ can be obtained by perturbed nonlinear dynamics $\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t)$ with a given control sequence $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_{M-1}]$, the approximation of the Koopman operator then can be obtained by solving the least square regression such that [21]

$$\mathbf{K} = \arg\min_{\mathbf{K}} \|\mathbf{\Pi}(\mathbf{Y}) - \mathbf{K}\hat{\mathbf{\Pi}}(\mathbf{X}, \mathbf{U})\|^2 \tag{5}$$

By constructing $\mathbf{G}_1$ and $\mathbf{G}_2$, an analytical solution for K may be computed

$$\mathbf{G_1} = \frac{1}{M}\sum_{i=1}^{M} \mathbf{\Pi}(\mathbf{y}_i)\hat{\mathbf{\Pi}}(\mathbf{x}_i, \mathbf{u}_i)^\top, \tag{6}$$

$$\mathbf{G_2} = \frac{1}{M}\sum_{i=1}^{M} \mathbf{\Pi}(\mathbf{x}_i)\hat{\mathbf{\Pi}}(\mathbf{x}_i, \mathbf{u}_i)^\top, \tag{7}$$

$$\mathbf{K} = \mathbf{G}_1\mathbf{G}_2^{-1} \tag{8}$$

D. Koopman Operator-Based Modeling

The 2D SRB model equations (see Eqn. 1-2) are nonlinear. Our goal is to compute a linear model using Koopman operator theory.

We identify a set of Koopman observer functions $\mathbf{\Pi}(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathbb{R}^N$ that can evolve linearly in the lifted observable space with the finite-dimensional Koopman operator $\mathbf{K} = [\mathbf{A}, \mathbf{B}]$, such that the dynamics are approximated by Eqn. 4. To perform the EDMD to find the linear Koopman predictor $\mathbf{A} \in \mathbb{R}^{N \times N}$ and $\mathbf{B} \in \mathbb{R}^{N \times m}$, a set of physics informed observable functions [16] $\bar{\mathbf{\Pi}} = [\underline{\mathbf{R}\dot{\theta}}, \underline{\mathbf{R}\dot{\theta}^2}, ..., \underline{\mathbf{R}\dot{\theta}^p}]^\top$ is selected to form the lifted state space, where the operator $\underline{(\cdot)} : \mathbb{R}^{l \times l} \rightarrow \mathbb{R}^{l^2}$ maps the matrix into a vector by concatenating the columns inside the matrix, then the linear SRB states can be augmented as

$$\mathbf{\Pi} = [1, \mathbf{p}, \theta, \dot{\mathbf{p}}, \dot{\theta}, \bar{\mathbf{\Pi}}]^\top \in \mathbb{R}^{7+4p} \tag{9}$$

Note that one of the observable is 1 here is needed to put the constant terms such as the gravity term. We also include the state $\mathbf{x} = [\mathbf{p}, \theta, \dot{\mathbf{p}}, \dot{\theta}]^\top$ so we can recover it for use in the LMPC discussed in the next section.

E. Koopman Operator-Based Model Predictive Control

Unlike [5] [6], which use MPC formulation to plan the flight phase, our approach, inspired by Raibert's controller [1], performs all calculations during the stance phase. We formulate a LMPC using the Koopman operator model as follows

$$\min_{\mathbf{u}} \sum_{i=0}^{k-1} \left\| \mathbf{x}_{t+i} - \mathbf{x}_t^d \right\|_{\mathbf{Q}_i} + \left\| \mathbf{u}_{t+i} \right\|_{\mathbf{R}_i} \tag{10}$$

$$\text{s.t.} \quad \mathbf{\Pi}_{t+i} = \mathbf{A}\mathbf{\Pi}_i + \mathbf{B}\mathbf{u}_i, i = 0, 1...k-1, \tag{11}$$

$$\mathbf{x}_i = \mathbf{C}_x \mathbf{\Pi}(\mathbf{x}_i), \mathbf{u}_{\min} \leq \mathbf{u}_i \leq \mathbf{u}_{\max} \tag{12}$$

where $\mathbf{C}_x \in \mathbb{R}^{n \times N}$ is selection matrix that pulls out the state $\mathbf{x}$ from $\mathbf{\Pi}(\mathbf{x})$; $\mathbf{Q}_i \in \mathbb{R}^{n \times n}$ and $\mathbf{R}_i \in \mathbb{R}^{m \times m}$ are user-chosen diagonal positive definite matrices.

For a given initial state $\mathbf{x}_0 \in \mathbb{R}^n$, using Eqn. 11 and 12 recursively for k time steps, we obtain

$$\mathbf{X}_{qp} = \mathbf{A}_{qp}\mathbf{x}_0 + \mathbf{B}_{qp}\mathbf{U}_{qp} \tag{13}$$

where $\mathbf{X}_{qp} \in \mathbb{R}^{n \times k}$ and $\mathbf{U}_{qp} \in \mathbb{R}^{m \times k}$ are the concatenated state and control from $1, 2, ..., k$ The cost function can then be rewritten as

$$\min_{\mathbf{U}_{qp}} \left\| \mathbf{A}_{qp}\mathbf{x}_0 + \mathbf{B}_{qp}\mathbf{U}_{qp} - \mathbf{X}_{qp}^d \right\|_{\mathbf{Q}_{qp}} + \left\| \mathbf{U}_{qp} \right\|_{\mathbf{R}_{qp}} \tag{14}$$

where $\mathbf{X}_{qp}^d \in \mathbb{R}^{n \times k}$ is the concatenated reference trajectories; $\mathbf{Q}_{qp} \in \mathbb{R}^{nk \times nk}$ and $\mathbf{R}_{qp} \in \mathbb{R}^{mk \times mk}$ are user-chosen diagonal positive weight matrix.

The QP can now be written as

$$\min_{\mathbf{U}_{qp}} \quad \frac{1}{2}\mathbf{U}_{qp}^\top \mathbf{H} \mathbf{U}_{qp} + \mathbf{P}\mathbf{U}_{qp} \tag{15}$$

$$\text{s.t.} \quad \underline{\mathbf{c}} \leq \mathbf{C}\mathbf{U}_{qp} \leq \bar{\mathbf{c}} \tag{16}$$

where $\mathbf{H} = 2(\mathbf{B}_{qp}^\top \mathbf{Q}_{qp} \mathbf{B}_{qp} + \mathbf{R}_{qp})$, $\mathbf{P} = 2(\mathbf{x}_0^\top \mathbf{A}_{qp}^\top \mathbf{Q}_{qp} \mathbf{B}_{qp} - \mathbf{X}_{qp}^{d\top} \mathbf{Q}_{qp} \mathbf{B}_{qp})$ and $\mathbf{C} \in \mathbb{R}^{mk \times mk}, \underline{\mathbf{c}} \in \mathbb{R}^{mk}, \bar{\mathbf{c}} \in \mathbb{R}^{mk}$ denoted the inequality constraints of control input.

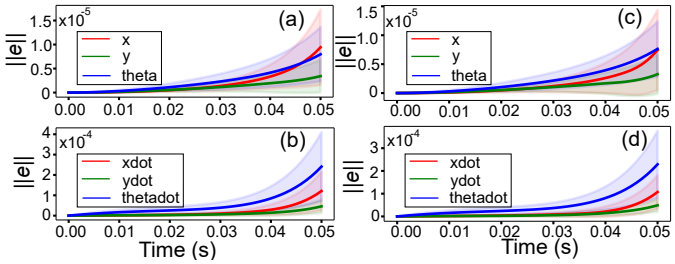

Fig. 3. 2D single rigid body model Koopman operator fitting result

F. Controller Implementation

The controller is implemented on a model of Unitree Go1 quadruped in MuJoCo version 2.0.0 [22] using Ubuntu 20.04 on an Intel Core i7 machine.

1) Finite State Machine: A finite state machine is used to manage the control logic, sending commands to leg controllers for FR, FL, RR, and RL legs. In our trot gait controller, FR-RL and FL-RR legs move in pairs, transitioning between swing and stance phases every $0.2 - 0.5$ sec. If a swing leg collides with an obstacle, detected by a contact sensor, the swing foot's position is held constant. In our bounding gait controller, FR-FL and RR-RL legs move in pairs, transitioning between front stance, flight phase, rear stance, and flight phase following a duty cycle where flight time, front stance time and rear stance time are 0.1 sec, 0.1 sec, 0.05 sec respectively. If legs collide with an obstacle during the flight phase, detected by a contact sensor, the foot's position is held constant.

2) Swing/Flight Phase Controller: The role of the swing/flight leg controller is to track the reference position of the foot in these two phases using joint torques. Given the foot reference position and velocity, $\mathbf{p}_{fi}^d, \dot{\mathbf{p}}_{fi}^d$, an analytical inverse (see [23] Sec. 3.5) is used to compute the corresponding joint reference position and velocity, $\mathbf{q}_i^d, \dot{\mathbf{q}}_i^d$. The following simple proportional-derivative controller is used to compute the leg i joint torque.

$$\tau_i = -\mathbf{K}_p(\mathbf{q}_i - \mathbf{q}_i^d) - \mathbf{K}_d(\dot{\mathbf{q}}_\mathbf{i} - \dot{\mathbf{q}}_i^d) \tag{17}$$

3) Linear Model Predictive Control: The LMPC uses the Koopman operator-based model and estimated torso states to compute desired ground reaction forces for the stance legs (see Sec. II-E). The planning horizon for the model predictive control is 6 ms or 166.67 Hz while the update horizon is 5 ms or 200 Hz. The MPC is solved online using qpSWIFT [24] in about 3 ms.

4) Stance Phase Controller: The stance leg controller module uses the desired ground reaction forces to estimate the joint torques using the Jacobian of the stance leg.

$$\tau_i = \mathbf{J}_i^\top \mathbf{f}_i \tag{18}$$

III. Results

A. Koopman Operator Model Fit

To generate data for the EDMD, we integrate the SRB Eqns. 1-2 with Runge-Kutta of order 4 with a fixed step size

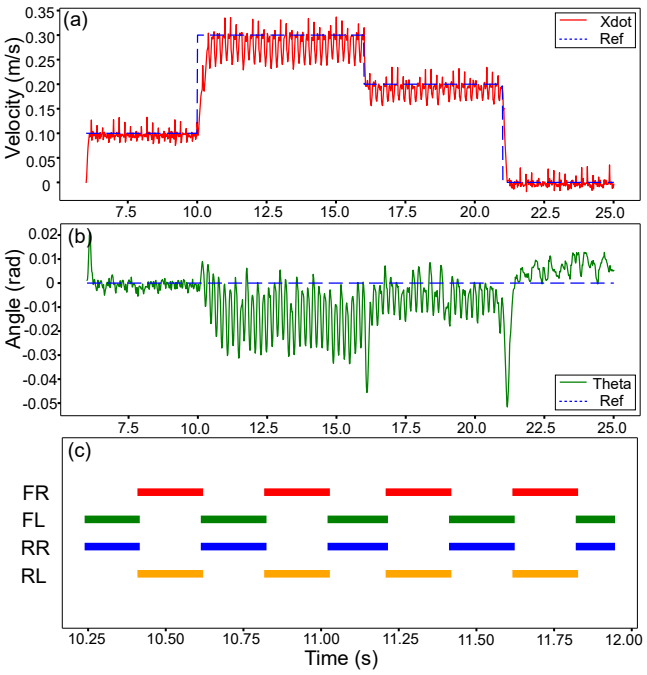

Fig. 4. Trotting forward velocity tracking

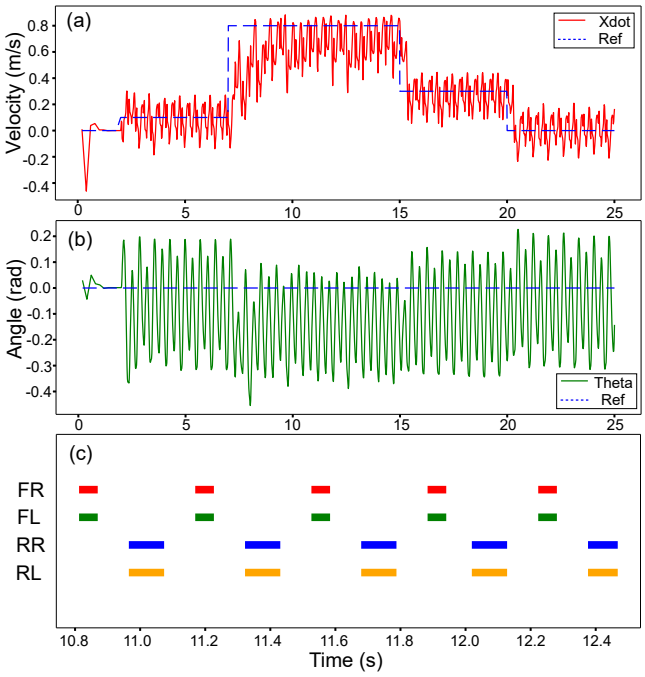

Fig. 5. Bounding forward velocity tracking

of $dt = 0.001$ sec from $t = 0$ to $t = 0.1$ sec. For each roll-out, we use a randomly generated initial conditions at $t = 0$ and random control input for every 0.001 sec. We use a total of 100 roll-outs to create a training dataset. Then using the $p = 4$ observables $\mathbf{\Pi}(\mathbf{x})$ discussed in Sec. II-D, we perform the EDMD to obtain a linear model.

Figure 3 (a)-(b), (c)-(d) shows the fitting result for trotting model and rear legs bounding models respectively. We did the extensive test by generating 50 initial conditions and using 50 random force profiles for a 0.05 sec roll-out using the SRB model, then using the same paired initial condition and force inputs we generated corresponding predictions using Koopman operator model. The linear model fitting results are evaluated by the error between the actual and predicted trajectories across 50 sets of data shown in Fig. 3, where (a) (c) for translation and orientation $\mathbf{p}, \theta$ states, (b) (d) for linear velocity and angular rates $\dot{\mathbf{p}}, \dot{\theta}$ states. The solid line shows the mean and the bands shows the variance. It can be seen that the errors are within $\pm 10^{-4}$ indicating a sufficiently accurate fit.

B. Koopman Operator LMPC in Trotting

To check the trotting gait reference tracking ability, we command the robot to follow a combination of reference forward as shown in Fig. 4 (a). The reference is parameterized by a step function, starting at 6 sec. The forward velocity is commanded to change sequentially: first to 0.1 m/s, then 0.3 m/s, followed by 0.2 m/s, and finally back to 0 m/s. At 10 sec, the command velocity undergoes a sharp increase from 0.1 m/s to 0.3 m/s, yet the controller successfully achieves stable forward velocity tracking with overall RMSE as 0.029. From Fig. 5 (b), one can see that when the forward reference velocity is relatively high, the controller temporarily allows the robot to pitch forward at a small angle to improve forward tracking performance. When the commanded forward velocity is set to zero, the pitch also returns to its neutral position. The trotting gait pattern diagram is shown in Fig. 4 (c) where the bars indicate the period of ground contact.

A rough terrain shown in Fig. 1 is designed to test the controller's disturbance rejection capability. The terrain is scattered with eight blocks, cubes, and capsules, ranging in height from 5 to 7 cm, placed on the path in front of the quadruped, which is commanded to track a forward velocity of 0.3 m/s. The experimental results are shown in Fig. 6 (a) (b). Around 3 sec and 9 sec, two slip incidents occur, causing the quadruped to experience a severe forward pitch. However, the controller successfully restores the robot's balance. Between 3 and 8 sec, the velocity tracking deviates from the reference due to the rough terrain, but the robot manages to maintain pitch stability, with a tracking error within ± 3 degrees while traversing.

C. Koopman Operator LMPC in Bounding

To check the trotting gait reference tracking ability, we command the robot to follow a combination of reference forward as shown in Fig. 5 (a). Compared to trotting, bounding allows the robot to achieve higher forward velocity tracking due to its higher gait frequency and additional flight times. Starting at 2 sec, the reference signal is defined by a step function that sequentially commands changes in forward velocity. The velocity first increases to 0.1 m/s, then jumps to 0.8 m/s, decreases to 0.3 m/s, and finally returns to 0 m/s. Notably, at 7 sec, there's a sharp increase from 0.1 m/s to 0.8 m/s in the commanded velocity. Despite this sudden change, the controller successfully maintains stable forward velocity

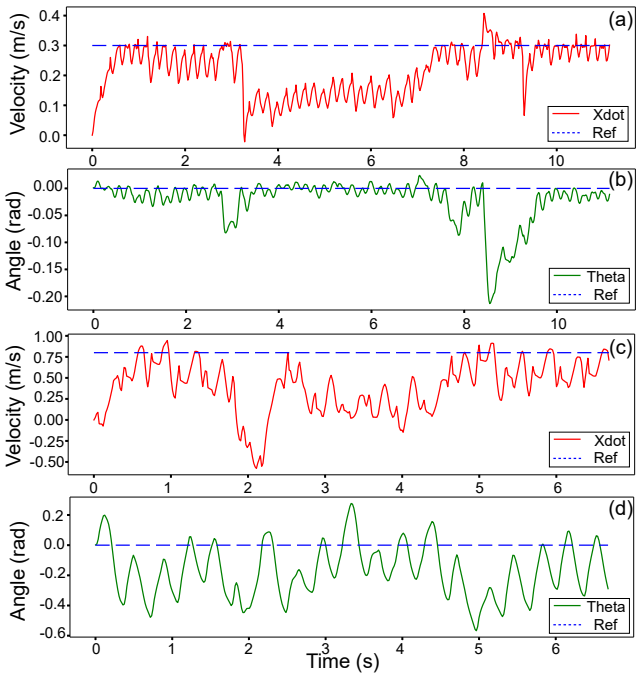

Fig. 6. Trotting (top two) and bounding (bottom two) on rough terrain

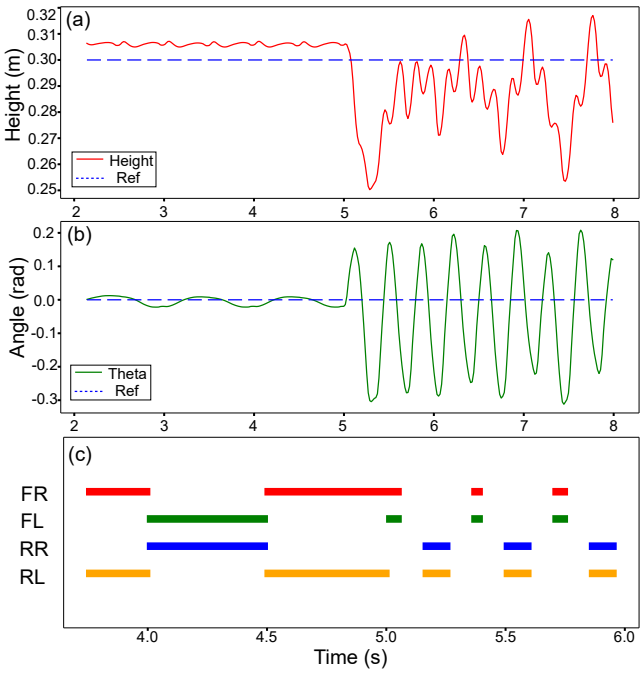

Fig. 7. Trotting transit to bounding

tracking with overall RMSE 0.189.Similar to the trotting gait, the same pitch angle pattern is observed in the bounding gait. From 7 to 15 sec, the forward velocity is commanded to be relatively high. To achieve effective forward tracking, the controller allows the pitch angle to tilt forward, and as the reference forward velocity decreases, the pitch angle returns to its neutral position. The bounding gait pattern diagram is shown in Fig. 5 (c) where the bars indicate the period of ground contact.

The rough terrain profile used in the trotting gait experiment is also used to test the disturbance rejection capability of the bounding gait. The experimental results are shown in Fig. 6 (c) (d). In the bounding experiment, the robot is commanded to move at a forward velocity of 0.75 m/s shown in Fig. 6 (c). Around 2 sec, the front foot stumbles on a block, causing a sharp decrease in forward velocity. Between 2 and 5 sec, the forward velocity tracking performance deviates due to the rough terrain; however, the controller is able to maintain a stable, periodic bounding pitch angle, as shown in Fig. 6 (d).

D. Koopman Operator LMPC in Gait Switch

In the gait switch experiment, two transitions are conducted: one from trotting to bounding, and the other from bounding to trotting, with the results shown in Fig. 7 and Fig. 8 respectively. The gait transitions occur while the robot is trotting or bounding in place. The two quadruped gaits are governed by separate finite state machines, and the transitions are based on a fixed switch time. Specifically, the transition from trot to bound occurs in 2 sec, while the transition from bound to trot takes 3 sec.

For the transition from trotting to bounding, the switch time can be placed at any point along the time axis due to the nature of the trotting gait, which always has a stance leg on the ground, providing better support polygon. On the other hand, one needs to choose the switch more carefully when transitioning from bounding to trotting. Due to the distinct phase composition of bounding, a large pitch moment may be induced if the switch occurs during bounding stance phase. For instance, if the switch happens during the front stance phase while the back foot is in the air, switching to the trotting gait will cause the back foot to kick the ground, generating a significant impulse that can disturb the robot's balance. Inspired by [1], we placed our bounding to trotting switch time in the fly phase to realize the smooth gait transition.

IV. Discussion, Conclusion, and Future Work

In this paper, we have used the Koopman operator to create a linear model of the 2D SRB model of a quadruped. This is used with LMPC to control the quadruped trotting, bounding gaits and theirs transition demonstrating the efficacy of the approach.

We do see that our linear model remains accurate with a prediction error within $\pm 10^{-4}$ up to 0.05 sec, its accuracy diminishes as time progresses. This limitation suggests that the linear model may not fully capture the complexities of the system over longer duration. To address this issue, one could use a neural network as a basis function [25], which would provide a more flexible and accurate fit. Alternatively, a bilinear Koopman operator-based model could provide a better representation of the system's dynamics, due to the SRB control affine feature [26]. But the use of bilinear model leads to a nonlinear model predictive control which is computationally challenging to solve [27]. One approach

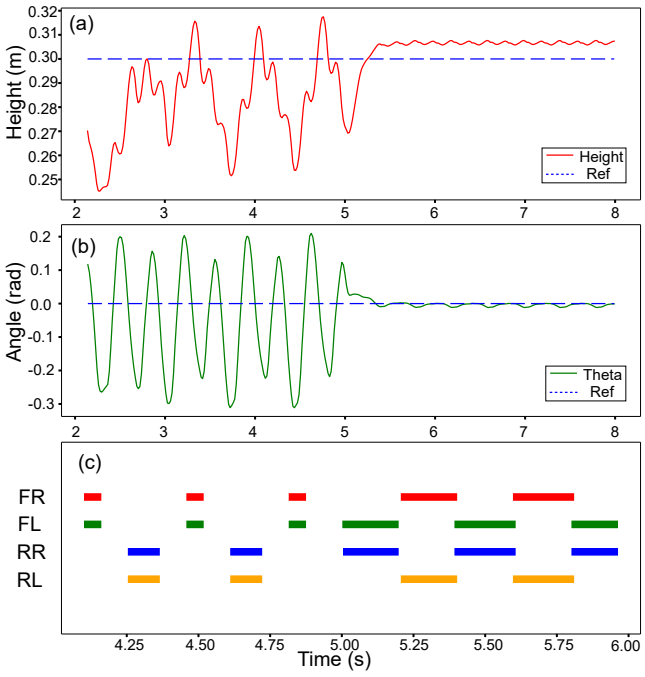

Fig. 8. Bounding transit to trotting

is to linearize the bilinear term at the operator point, which leads to an LMPC [28].

In this work, we implement simple gaits. Since most gaits are inherently periodic, the gait frequency can be easily modified by changing the cycle duration [7], while gait speed can be adjusted by altering the phase combination and frequency. For example, a flying trot can increase forward velocity by incorporating a flight phase within the duty cycle [5].

Our future work will explore methods to increase the robustness of the quadruped to larger disturbances, increase the range of movement of the robot, and using a neural network as the basis function to increase the model accuracy.

REFERENCES

[1] M. H. Raibert, "Trotting, pacing and bounding by a quadruped robot," *Journal of biomechanics*, vol. 23, pp. 79–98, 1990.

[2] S. Strogatz, *Nonlinear dynamics and chaos*. Addison-Wesley Reading, 1994.

[3] H.-W. Park, M. Y. Chuah, and S. Kim, "Quadruped bounding control with variable duty cycle via vertical impulse scaling," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 3245–3252.

[4] 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.

[5] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the mit cheetah 3 through convex model-predictive control," in *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2018, pp. 1–9.

[6] Y. Ding, A. Pandala, and H.-W. Park, "Real-time model predictive control for versatile dynamic motions in quadrupedal robots," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8484–8490.

[7] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1560–1567, 2018.

[8] S. Chen, B. Zhang, M. W. Mueller, A. Rai, and K. Sreenath, "Learning torque control for quadrupedal locomotion," in *2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)*. IEEE, 2023, pp. 1–8.

[9] G. Bellegarda and Q. Nguyen, "Robust high-speed running for quadruped robots via deep reinforcement learning," *arXiv preprint arXiv:2103.06484*, 2021.

[10] A. Seyfarth, H. Geyer, and H. Herr, "Swing-leg retraction: a simple control model for stable running," *Journal of Experimental Biology*, vol. 206, no. 15, pp. 2547–2555, 2003.

[11] S. Gangapurwala, A. Mitchell, and I. Havoutis, "Guided constrained policy optimization for dynamic quadrupedal robot locomotion," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3642–3649, 2020.

[12] J. Zhang, S. Heim, S. H. Jeon, and S. Kim, "Learning emergent gaits with decentralized phase oscillators: on the role of observations, rewards, and feedback," *arXiv preprint arXiv:2402.08662*, 2024.

[13] F. Jenelten, J. He, F. Farshidian, and M. Hutter, "Dtc: Deep tracking control," *Science Robotics*, vol. 9, no. 86, p. eadh5401, 2024.

[14] B. O. Koopman, "Hamiltonian systems and transformation in hilbert space," *Proceedings of the National Academy of Sciences*, vol. 17, no. 5, pp. 315–318, 1931.

[15] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley, "A data–driven approximation of the koopman operator: Extending dynamic mode decomposition," *Journal of Nonlinear Science*, vol. 25, pp. 1307–1346, 2015.

[16] S. S. Narayanan, D. Tellez-Castro, S. Sutavani, and U. Vaidya, "Se (3) koopman-mpc: Data-driven learning and control of quadrotor uavs," *IFAC-PapersOnLine*, vol. 56, no. 3, pp. 607–612, 2023.

[17] M. Rahmani and S. Redkar, "Enhanced koopman operator-based robust data-driven control for 3 degree of freedom autonomous underwater vehicles: A novel approach," *Ocean Engineering*, vol. 307, p. 118227, 2024.

[18] B. Kim, D. Necsulescu, and J. Sasiadek, "Model predictive control of an autonomous vehicle," in *2001 IEEE/ASME International Conference on Advanced Intelligent Mechatronics. Proceedings (Cat. No. 01TH8556)*, vol. 2. IEEE, 2001, pp. 1279–1284.

[19] L. Shi and K. Karydis, "Acd-edmd: Analytical construction for dictionaries of lifting functions in koopman operator-based nonlinear robotic systems," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 906–913, 2021.

[20] D. Bruder, X. Fu, R. B. Gillespie, C. D. Remy, and R. Vasudevan, "Data-driven control of soft robots using koopman operator theory," *IEEE Transactions on Robotics*, vol. 37, no. 3, pp. 948–961, 2020.

[21] Q. Li, F. Dietrich, E. M. Bollt, and I. G. Kevrekidis, "Extended dynamic mode decomposition with dictionary learning: A data-driven adaptive spectral decomposition of the koopman operator," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 27, no. 10, 2017.

[22] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2012, pp. 5026–5033.

[23] P. A. Bhounsule and C.-M. Yang, "A simple controller for omnidirectional trotting of quadrupedal robots: Command following and waypoint tracking," *Robotics*, vol. 12, no. 2, 2023. [Online]. Available: https://www.mdpi.com/2218-6581/12/2/35

[24] A. G. Pandala, Y. Ding, and H.-W. Park, "qpswift: A real-time sparse quadratic program solver for robotic applications," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3355–3362, 2019.

[25] E. Yeung, S. Kundu, and N. Hodas, "Learning deep neural network representations for koopman operators of nonlinear dynamical systems," in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 4832–4839.

[26] D. Bruder, X. Fu, and R. Vasudevan, "Advantages of bilinear koopman realizations for the modeling and control of systems with unknown dynamics," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4369–4376, 2021.

[27] C. Folkestad and J. W. Burdick, "Koopman nmpc: Koopman-based learning and nonlinear model predictive control of control-affine systems," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 7350–7356.

[28] S. Yu, C. Shen, and T. Ersal, "Autonomous driving using linear model predictive control with a koopman operator based bilinear vehicle model," *IFAC-PapersOnLine*, vol. 55, no. 24, pp. 254–259, 2022.