Koopman Operator Based Linear Model Predictive Control for Quadruped Trotting

Chun-Ming Yang and Pranav A. Bhounsule

***Abstract*— Online optimal control of quadruped robots would enable them to adapt to varying inputs and changing conditions in real time. A common way of achieving this is linear model predictive control (LMPC), where a quadratic programming (QP) problem is formulated over a finite horizon with a quadratic cost and linear constraints obtained by linearizing the equations of motion and solved on the fly. However, the model linearization may lead to model inaccuracies. In this paper, we use the Koopman operator to create a linear model of the quadrupedal system in high dimensional space which preserves the nonlinearity of the equations of motion. Then using LMPC, we demonstrate high fidelity tracking and disturbance rejection on a quadrupedal robot. This is the first work that uses the Koopman operator theory for LMPC of quadrupedal locomotion.**

I. Introduction

Quadruped robots, due to their low center of gravity, wide base of support, and ability to move over uneven terrain and obstacles using limited footholds, provide a viable means of using mobile robots in applications such as first responders, industrial workers, and helpers at home. In this regard, the low-level or joint-level control of these robots is of paramount importance. Current model-based approaches restrict to linearized models while model-free approaches use large sample size. This paper addresses the problem of generating low-level control of quadrupeds in a generalizable, sample-efficient manner without resorting to model linearization using the Koopman operator theory.

The traditional approach of performing low-level control is to create a parametric controller and then fine tune gains either in simulation and/or hardware [1]. However, this approach may not scale to different quadrupeds and/or scenarios because it involves non-intuitive hand tuning. This can be overcome by formulating and solving a trajectory optimization (TO) problem offline by tuning a cost function subject to state and control constraints. The result of TO is a reference trajectory and/or open loop torque profile. To implement TO on hardware, one needs an additional feedback control layer to ensure high-level tracking in the presence of disturbances and noise [2].

Model predictive control (MPC) is similar to TO in that it solves an optimal control problem. However, there are two major differences: one, it solves the TO online over a finite horizon to make it computationally efficient, and two, it repeatedly updates the control using sensor measurements,

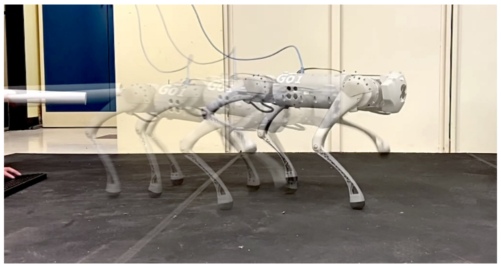

Fig. 1. Push recovery using Koopman operator-based linear model predictive control

thus providing accurate tracking even in the presence of disturbances and noise. MPC for quadruped control involves using a physics model, such as single rigid body (SRB), an idealization, for planning the reaction forces and then mapping those forces to joint torques using the Jacobian [3]. Some of the nonlinearities in SRB may be ignored or linearized to avoid formulating a nonlinear MPC problem, which is challenging to solve in real time [4]. The limitations of SRB, such as model mismatch, can be avoided by using a whole-body model for MPC, but one needs an engineering feat to make the online computation fast enough for real-time control [5]. Deep Reinforcement learning (DRL) is an model-free method that solves the optimization problem over the entire state space in an offline manner to learn a neural network that maps the control to the states. To enable seamless transfer to hardware, one could learn trajectories which are then implemented using feedback control on hardware [6]. Another approach is to use dynamic randomization, where torque profiles are generated for a wide variety of model and environmental parameters such that when the controller is transferred to hardware, it is able to cope with the sim-to-real gap [7]. However, DRL is not sample efficient. To achieve sample efficiency, one could use model-based TO to plan reference motion and model-free RL to track the reference motion [8].

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 system in 1931 by Koopman [9], but only recently tools have been devised to model controlled systems [10]. The applications of the Koopman operator are currently limited to a few simple smooth systems, such as quadcopters [11], underwater vehicles [12], autonomous cars [13], two-link planar manipulators [14], and soft robot manipulators [15].

In this paper, we use the Koopman operator to create

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.

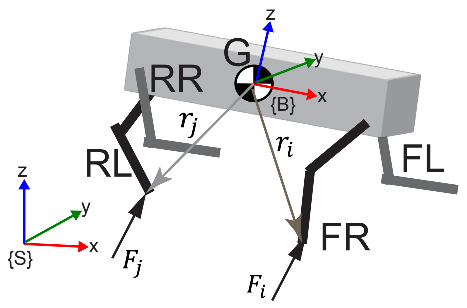

Fig. 2. Single Rigid Body Model

a relatively low degree of freedom, a linear model of a quadruped, and use Linear Model Predictive Control (LMPC) to demonstrate slip and push recovery, reference tracking in translation, and rotation. The main novelty of this work is that it is the first demonstration of the use of Koopman operator theory for modeling and controlling a quadrupedal robot. The method is generalizable to other quadruped system because we use SRB, a well known idealization of many quadruped robots, for system identification. Finally, hardware demonstration shows the efficacy of the approach to do real-time optimal control.

II. Methods

A. Single Rigid Body Model (SRB)

The quadruped system shown in Fig. 2 is modeled using SRB model [16]

$$\dot{\boldsymbol{\Theta}} = \mathbf{A}_{\Omega}^{-1}(\theta,\psi)\boldsymbol{\Omega} \tag{1}$$

$$\ddot{\mathbf{p}} = \frac{\mathbf{R}(\mathbf{F}_i+\mathbf{F}_j)}{m} - \mathbf{g} \tag{2}$$

$$\mathbf{I}\dot{\boldsymbol{\Omega}} = \boldsymbol{\Omega}\times(\mathbf{I}\boldsymbol{\Omega}) + \mathbf{r}_i\times\mathbf{F}_i + \mathbf{r}_j\times\mathbf{F}_j \tag{3}$$

where $\mathbf{p}\in\mathbb{R}^3$ and $\dot{\mathbf{p}}\in\mathbb{R}^3$ are the SRB center of mass (CoM) position and velocity both in the world frame respectively; $\boldsymbol{\Theta} = [\phi,\theta,\psi]^\top$, $\boldsymbol{\Omega}\in\mathbb{R}^3$ are orientation expressed by X-Y-Z Euler angles and angular velocities in body frame respectively. The leg ground reaction force for the foot i is $\mathbf{F}_i\in\mathbb{R}^3$ and the distance from the foot to the center of mass is $\mathbf{r}_i\in\mathbb{R}^3$; $\mathbf{A}_\Omega(\theta,\psi) = [c_\psi/c_\theta, -s_\psi/c_\theta, 0; s_\psi, c_\psi, 0; -c_\psi s_\theta/c_\theta, s_\psi s_\theta/c_\theta, 1]\in\mathbb{R}^{3\times3}$ is the matrix transforms body angular to Euler angle rate, where $c_A=\cos A, s_A=\sin A$; $\mathbf{R}\in\mathbb{R}^{3\times3}$ is the rotation matrix mapping vector from body frame to global frame; m is the mass of the robot; $\mathbf{g}\in\mathbb{R}^3$ is the gravity vector in the world frame; $\mathbf{I}\in\mathbb{R}^3$ is the inertia matrix of the torso in the body frame. Note that Eqns. 1-3 are nonlinear and they can be compactly written as: $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})+\mathbf{g}(\mathbf{x})\mathbf{u}$, where $\mathbf{x} = [\mathbf{p},\boldsymbol{\Theta},\dot{\mathbf{p}},\boldsymbol{\Omega}]^\top$ and $\mathbf{u} = [\mathbf{F}_i,\mathbf{F}_j]^\top$.

B. 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 $\boldsymbol{\Pi}(\mathbf{x})$ 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

$$[\mathcal{K}\boldsymbol{\Pi}](\mathbf{x},\mathbf{u}) = \boldsymbol{\Pi}\circ\mathbf{f}(\mathbf{x},\mathbf{u}) \tag{4}$$

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 [10], 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

$$\boldsymbol{\Pi}(\mathbf{x}_{t+1}) = \begin{bmatrix}\mathbf{A} & \mathbf{B}\end{bmatrix}\begin{bmatrix}\boldsymbol{\Pi}(\mathbf{x}_t)\\ \mathbf{u}_t\end{bmatrix} = \mathbf{K}\hat{\boldsymbol{\Pi}}(\mathbf{x_t},\mathbf{u_t}) \tag{5}$$

where $\boldsymbol{\Pi}(\mathbf{x}_t) = [\pi_1(\mathbf{x}_t),...,\pi_N(\mathbf{x}_t)]^T\in\mathbb{R}^N$ 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 [17]

$$\mathbf{K} = \arg\min_{\mathbf{K}} \|\boldsymbol{\Pi}(\mathbf{Y}) - \mathbf{K}\hat{\boldsymbol{\Pi}}(\mathbf{X},\mathbf{U})\|^2 \tag{6}$$

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

$$\mathbf{G_1} = \frac{1}{M}\sum_{i=1}^{M}\boldsymbol{\Pi}(\mathbf{y}_i)\hat{\boldsymbol{\Pi}}(\mathbf{x}_i,\mathbf{u}_i)^\top, \tag{7}$$

$$\mathbf{G_2} = \frac{1}{M}\sum_{i=1}^{M}\boldsymbol{\Pi}(\mathbf{x}_i)\hat{\boldsymbol{\Pi}}(\mathbf{x}_i,\mathbf{u}_i)^\top, \tag{8}$$

$$\mathbf{K} = \mathbf{G}_1\mathbf{G}_2^{-1} \tag{9}$$

C. Koopman Operator-Based Modeling

The SRB model equations (see Eqn. 1-3) are nonlinear. Our goal is to compute a linear model using Koopman operator theory.

We identify a set of Koopman observer functions $\boldsymbol{\Pi}(\mathbf{x}):\mathbb{R}^n\to\mathbb{R}^N$ 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. 5. 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 [11] $\bar{\boldsymbol{\Pi}} = [\underline{\mathbf{R}\boldsymbol{\Omega}},\underline{\mathbf{R}\boldsymbol{\Omega}^2},...,\underline{\mathbf{R}\boldsymbol{\Omega}^p}]^\top$ is selected to form the lifted state space, where the operator $\underline{(\cdot)}:\mathbb{R}^{l\times l}\to\mathbb{R}^{l^2}$ maps the matrix into a vector by concatenating the columns inside the matrix, then the linear SRB states can be augmented as

$$\boldsymbol{\Pi} = [1,\mathbf{p},\boldsymbol{\Theta},\dot{\mathbf{p}},\boldsymbol{\Omega},\bar{\boldsymbol{\Pi}}]^\top\in\mathbb{R}^{13+9p} \tag{10}$$

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 $\mathbf{x} = [\mathbf{p},\boldsymbol{\Theta},\dot{\mathbf{p}},\boldsymbol{\Omega}]^\top$ so we can recover it for use in the LMPC discussed in the next section.

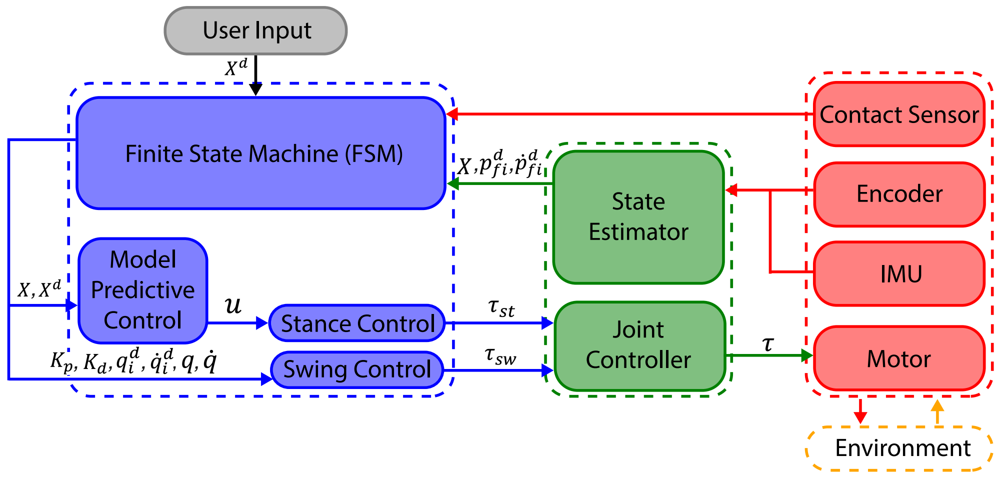

Fig. 3. Control diagram for implementation on the quadruped in simulation and hardware

D. Koopman Operator-Based Model Predictive Control

We formulate a LMPC using the Koopman operator model as follows

$$\min_{\mathbf{u}} \sum_{i=0}^{k-1} \left\|\mathbf{x}_{t+i} - \mathbf{x}_t^d\right\|_{\mathbf{Q}_i} + \left\|\mathbf{u}_{t+i}\right\|_{\mathbf{R}_i} \tag{11}$$

$$\text{s.t.} \quad \mathbf{\Pi}_{t+i} = \mathbf{A}\mathbf{\Pi}_i + \mathbf{B}\mathbf{u}_i, i = 0, 1...k-1, \tag{12}$$

$$\mathbf{x}_i = \mathbf{C}_x \mathbf{\Pi}(\mathbf{x}_i), \mathbf{u}_{\text{min}} \leq \mathbf{u}_i \leq \mathbf{u}_{\text{max}} \tag{13}$$

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. 12 and 13 recursively for k time steps, we obtain

$$\mathbf{X}_{qp} = \mathbf{A}_{qp}\mathbf{x}_0 + \mathbf{B}_{qp}\mathbf{U}_{qp} \tag{14}$$

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

$$\min_{\mathbf{U}_{qp}} \left\|\mathbf{A}_{qp}\mathbf{x}_0 + \mathbf{B}_{qp}\mathbf{U}_{qp} - \mathbf{X}_{qp}^d\right\|_{\mathbf{Q}_{qp}} + \left\|\mathbf{U}_{qp}\right\|_{\mathbf{R}_{qp}} \tag{15}$$

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

$$\min_{\mathbf{U}_{qp}} \quad \frac{1}{2}\mathbf{U}_{qp}^\top \mathbf{H}\mathbf{U}_{qp} + \mathbf{P}\mathbf{U}_{qp} \tag{16}$$

$$\text{s.t.} \quad \underline{\mathbf{c}} \leq \mathbf{C}\mathbf{U}_{qp} \leq \overline{\mathbf{c}} \tag{17}$$

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 $\mathbf{C} \in \mathbb{R}^{mk \times mk}, \underline{\mathbf{c}} \in \mathbb{R}^{mk}, \overline{\mathbf{c}} \in \mathbb{R}^{mk}$ denoted the inequality constraints of control input.

E. Controller Implementation in Simulation and Hardware

The controller is implemented on a model of Unitree Go1 quadruped in MuJoCo version 2.0.0 [18] using Ubuntu 20.04 on an Intel Core i7 and on hardware.

Each leg of the Go1 robot comprises two 0.2 m links with three degrees of freedom. The torso weighs 4.75 kg, and each leg weighs 2 kg. The torso has an IMU providing orientation, angular speed, and linear acceleration. Each joint has an absolute encoder providing joint angles. The joint torque limits are ± 33.5 Nm. An onboard Raspberry Pi 4 processes the IMU and encoder data and sends torque, position, and velocity references along with their gains to the motor controller at 1000 Hz. The motor controller operates at 10 kHz to ensure high-fidelity torque tracking. Figure 3 gives an overview of the Koopman operator-based LMPC in simulation and hardware and details follow

1) Finite State Machine: A finite state machine is used to manage the control logic, sending commands to leg controllers for Front Right (FR), Front Left (FL), Rear Right (RR), and Rear Left (RL) legs. In a trot gait, FR-RL and FL-RR legs move in pairs, transitioning between swing and stance phases every 0.2 sec. If a swing leg collides with an obstacle, detected by a contact sensor, the swing foot's position is held constant.

2) Swing Leg Controller: The role of the swing leg controller is to track the reference position of the swinging foot using joint torques on the swing leg joints. Given the swing foot reference position and velocity, $\mathbf{p}_{fi}^d, \dot{\mathbf{p}}_{fi}^d$, an analytical inverse (see [1] 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 joint torque for leg i at 1 kHz.

$$\tau_i = -\mathbf{K}_p(\mathbf{q}_i - \mathbf{q}_i^d) - \mathbf{K}_d(\dot{\mathbf{q}}_\mathbf{i} - \dot{\mathbf{q}}_i^d) \tag{18}$$

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-D). 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 [19] in about 3 ms.

4) Stance Leg Controller: The stance leg controller module uses the desired ground reaction forces to estimate the

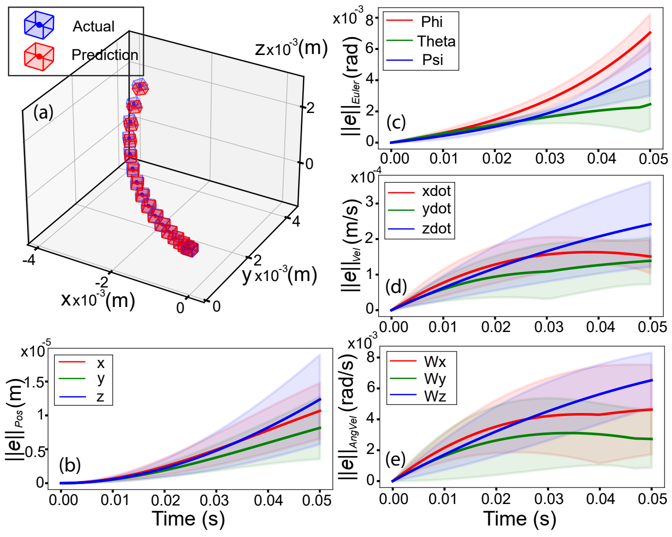

Fig. 4. Fitting the Single Rigid Body Model using the Koopman operator

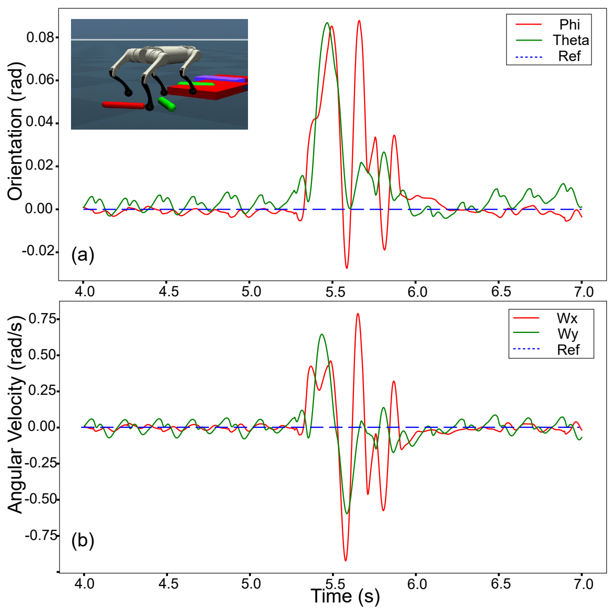

Fig. 5. Simulation: Slip and Recovery

joint torques using the Jacobian of the stance leg.

$$\tau_i = \mathbf{J}_i^\top \mathbf{f}_i \tag{19}$$

5) State Estimation: The state estimation module estimates the linear velocity of the torso by fusing the acceleration data from the IMU and the joint encoder data using an Extended Kalman Filter (EKF). The EKF uses the integrated acceleration as the process model and the estimated linear velocity using the Jacobian of the contact feet and the joint velocity as the measurement model.

III. Results

A. Koopman Operator Model Fit

To generate data for the EDMD, we integrate the SRB Eqns. 1-3 with Runge-Kutta of order 4 with a fixed step size 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-C, we perform the EDMD to obtain a linear model.

Figure 4 (a) shows fidelity of the fit on one out of fifty testing trajectories. 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. 4, where (b) for translation $\mathbf{p}$, (c) for orientation $\mathbf{\Theta}$, (d) for linear velocity $\dot{\mathbf{p}}$, (e) for angular rates $\mathbf{\Omega}$. The solid line shows the mean and the bands shows the variance. It can be seen that the errors are within $\pm 10^{-3}$ indicating a sufficiently accurate fit.

B. Koopman Operator LMPC in Simulation

To check the disturbance rejection, approximately 10 objects of height in the range of $5-7$ cm are scattered on the ground to intentionally make the robots' feet slip (see Fig. 5 inset). During the rough terrain simulation experiment, the first significant slip occurs when the robot's rear foot strikes the second object, causing the robot to lose its balance. However, the robot regains stability when the foot makes contact with the flat ground within 1 sec, recalculating the adequate force for effective disturbance rejection with orientation $\mathbf{\Theta}$ and angular velocity $\mathbf{\Omega}$ tracking errors (RMSE) of 0.012 and 0.092 respectively.

To check the reference tracking ability, we command the robot to follow a combination of reference forward and lateral speed as shown in Fig. 6 (a), the RMSE for linear velocity $\dot{\mathbf{p}}$ is 0.021. Then a combination of forward speed and turning speed as shown in Fig. 6 (d) (e) is commanded with resulting tracking RMSE for linear velocity $\dot{\mathbf{p}}$ as 0.008, and for angular velocity $\mathbf{\Omega}$ as 0.056.

C. Koopman Operator LMPC in Hardware

To evaluate the controller's disturbance rejection capabilities, the robot, while trotting, is subjected to a forceful push that causes its forward velocity to oscillate by up to 0.4 m/s (see Fig. 7). Despite this perturbation, the robot successfully recovers from a pitch deviation of 6.5 degrees within 2 sec with the tracking errors for orientation $\mathbf{\Theta}$ and angular velocity $\mathbf{\Omega}$, measured as RMSE, being 0.018 and 0.404, respectively.

To evaluate the reference tracking capability, we command the robot to follow a combination of forward and lateral reference speeds as shown in Fig. 8 (a). The ground reaction forces in Fig. 8 (b) (c), zoomed in between $4.6-5.4$ sec, demonstrates the controller's ability to manage the steep

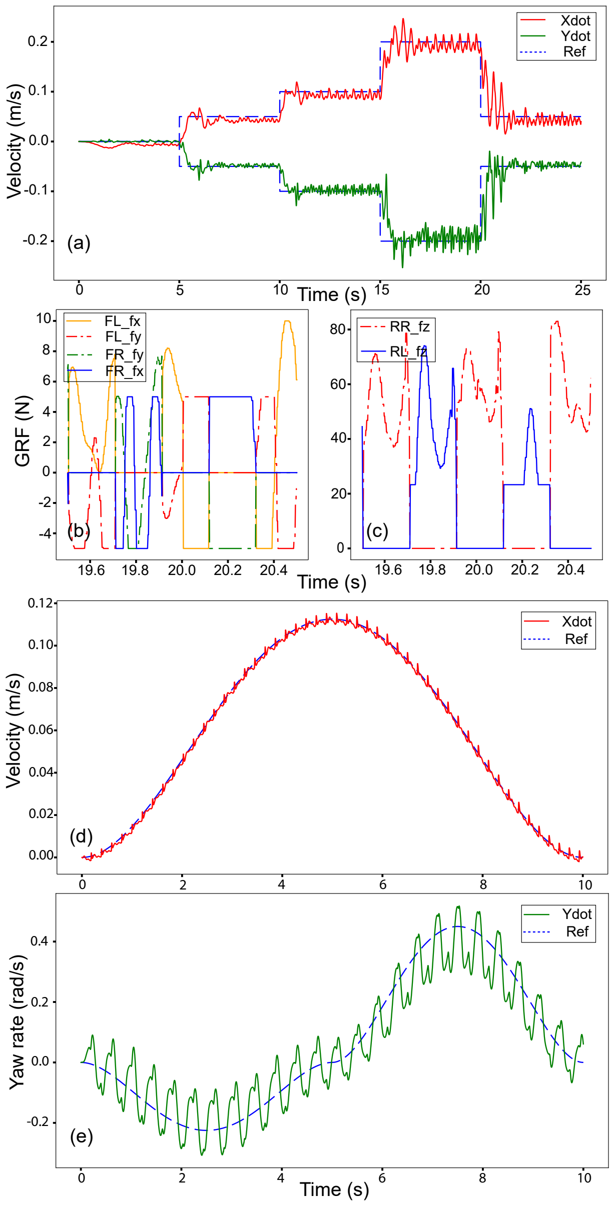

Fig. 6. Simulation: Tracking a forward and lateral speed.

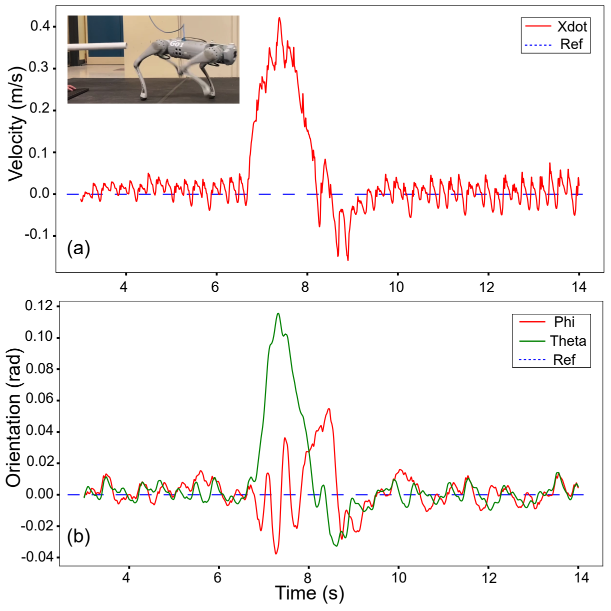

Fig. 7. Hardware: Push and Recovery

velocity change from 0 to 0.2 m/s for forward and 0 to -0.1 m/s for lateral as commanded by a step function, with the tracking error of the linear velocity $\dot{\mathbf{p}}$ quantified by RMSE of 0.026, lastly, a combination of forward and turning speed is commanded as shown in Fig. 8 (d) (e). The RMSE is 0.048 for linear velocity $\dot{\mathbf{p}}$ and 0.415 for $\mathbf{\Omega}$ angular velocity.

IV. Discussion, Conclusion, and Future Work

In this paper, we have used the Koopman operator to create a linear model of the SRB model of a quadruped. This is used with LMPC to control the trot gait of a quadruped in simulation and hardware demonstrating the efficacy of the approach.

We use the analytical model (the SRB model) instead of hardware data for system identification. Often hardware data based data identification is challenging because there may not be enough excitation to tease out the model parameters [20], but with simulated data it is always possible to generate a richer data set. Although we have used data-based EDMD to identify the linear model, analytical method such as direct encoding may also be used [21].

The SRB model enables us to reduce the dimension of the Koopman operator substantially. The full robot dynamics has 18 degrees of freedom (6 in the torso and 3 in each of the 4 legs). However, if we assume the legs are massless, then we only need to model the torso. This is done using the SRB model which has only 6 degrees of freedom. The result is that we can adequately describe the nonlinear model using only 49 observables.

The use of the Koopman operator based linear model leads to a QP over a finite horizon (LMPC) which has an analytical solution if there are no constraints. In our case, we do have control constraints and hence need to solve the problem numerically. However, we are able to solve the LMPC in 3 ms for a horizon of 6 ms enabling real-time optimal control.

We do see that our linear model remains accurate with a prediction error within $\pm 10^{-3}$ up to $0.05s$, its accuracy diminishes as time progresses. The SRB model is a control affine model and a bilinear Koopman operator-based model should better describe the dynamics [22]. But the use of bilinear model leads to a nonlinear model predictive control which is computationally challenging to solve [23]. One approach is to linearize the bilinear term at the operator point, which leads to an LMPC [24].

In conclusion, the generation of simple linear model with only 49 observables using the idealized SRB model is found to be adequate to do high-fidelity LMPC of a quadruped in hardware for reference tracking and moderate disturbances.

Our future work will explore methods to increase the robustness of the quadruped to larger disturbances, increase

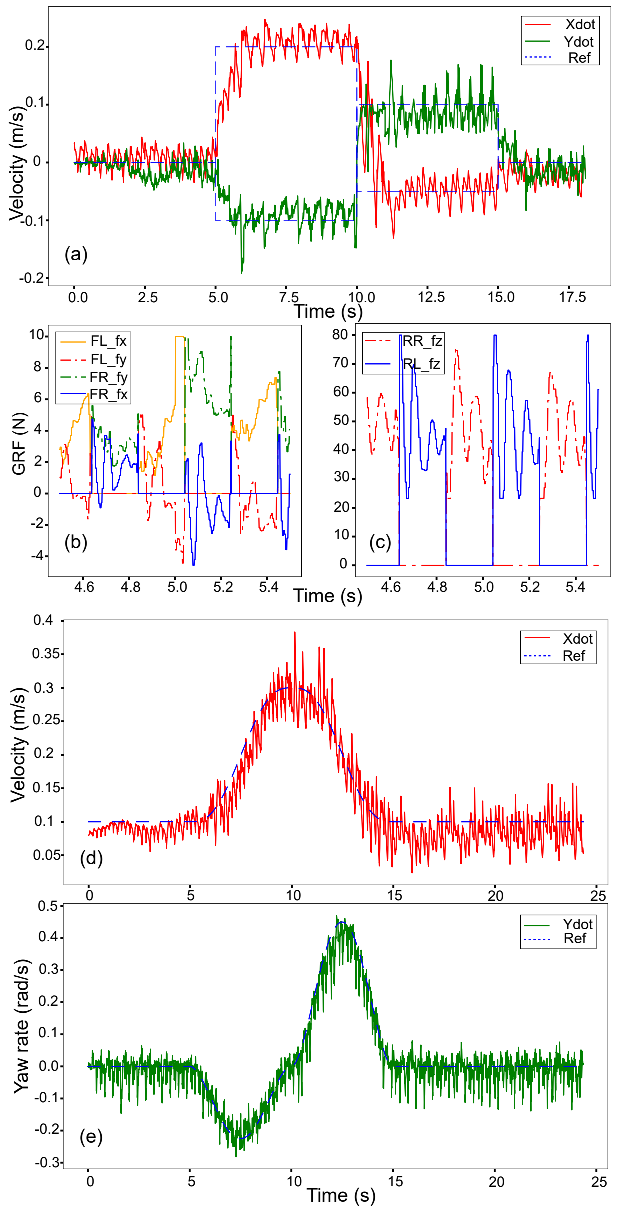

Fig. 8. Hardware: Tracking a fore-aft and side-way speed.

the range of movement of the robot, and explore multiple gaits such as bounding, pacing, trotting, and transitioning between these gaits.

References

[1] 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

[2] P. A. Bhounsule, J. Cortell, A. Grewal, B. Hendriksen, J. D. Karssen, C. Paul, and A. Ruina, "Low-bandwidth reflex-based control for lower power walking: 65 km on a single battery charge," *International Journal of Robotics Research*, 2014.

[3] D. Kim, J. Di Carlo, B. Katz, G. Bledt, and S. Kim, "Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control," *arXiv preprint arXiv:1909.06586*, 2019.

[4] Y. Ding, A. Pandala, C. Li, Y.-H. Shin, and H.-W. Park, "Representation-free model predictive control for dynamic motions in quadrupeds," *IEEE Transactions on Robotics*, vol. 37, no. 4, pp. 1154–1171, 2021.

[5] M. Neunert, M. Stäuble, M. Giftthaler, C. D. Bellicoso, J. Carius, C. Gehring, M. Hutter, and J. Buchli, "Whole-body nonlinear model predictive control through contacts for quadrupeds," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1458–1465, 2018.

[6] G. Feng, H. Zhang, Z. Li, X. B. Peng, B. Basireddy, L. Yue, Z. Song, L. Yang, Y. Liu, K. Sreenath *et al.*, "Genloco: Generalized locomotion controllers for quadrupedal robots," *arXiv preprint arXiv:2209.05309*, 2022.

[7] 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.

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

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] J. Jovic, F. Philipp, A. Escande, K. Ayusawa, E. Yoshida, A. Kheddar, and G. Venture, "Identification of dynamics of humanoids: Systematic exciting motion generation," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 2173–2179.

[21] H. Harry Asada and F. E. Sotiropoulos, "Dual faceted linearization of nonlinear dynamical systems based on physical modeling theory," *Journal of Dynamic Systems, Measurement, and Control*, vol. 141, no. 2, p. 021002, 2019.

[22] 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.

[23] 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.

[24] 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.