Learning to Control Robot Hopping over Uneven Terrain

M.D. Lemmon, P.M. Wensing, V. Kurtz, and H. Lin

Abstract— **Learning how to traverse uneven terrain is an open challenge for robotic locomotion. Machine learning techniques such as deep reinforcement learning (DRL) have been used to train walking robots, but the resulting control policies often fail in the presence of neglected dynamics or uneven terrain. This paper demonstrates that by separating the learning problem into the identification of the robot's natural and steady-state response to terrain variations, one can efficiently adapt the step-length controller for Raibert's hopper to successfully traverse uneven terrain even when there are passive dynamics that were neglected in earlier training. The approach uses a moment-matching (MM) model (a.k.a. MM-abstraction) of the hopper's steady-state gait over a given class of terrain to construct a model-following (MF) model (a.k.a. MF-abstraction) of the hopper's natural dynamics. These abstractions are then used to demonstrate the capacity of a simple passivity-based adaptive controller to enforce robust tracking of the hopper's steady-state response to uneven terrain.**

I. INTRODUCTION

Learning to adaptively traverse uneven terrain is an open issue for robotic locomotion. While deep reinforcement learning (DRL) [1] is capable of learning robotic skills [2], the resulting control policies often fail on robotic hardware in an open and uncertain world. Moreover, DRL's reliance on offline training makes it difficult for that hardware to adapt previously learned skills to new situations. With these observations in mind, this paper examines an approach that decomposes the learning problem into the identification of the system's *natural* and *steady-state* response. Because models for these two behaviors can be identified using regression techniques, the proposed approach facilitates the *online* adaptation of a control policy to maintain hopping in the presence of terrain variation and excitation of unknown passive dynamics. This paper demonstrates the method on the step-length controller of Raibert's hopper [3].

The approach rests on a well-known behavioral decomposition for dynamical systems with compact trajectories. Any trajectory of such a "closed" dynamical system can be classified as being either recurrent or nonrecurrent [4]. The distinction is that recurrent trajectories are contained inside the system's positive limit set and nonrecurrent trajectories "connect" disjoint components of the positive limit set. What we call "steady state" response is simply a recurrent trajectory. What we call "natural" response is then the difference between a nonrecurrent trajectory and the recurrent orbit that it converges to asymptotically.

This behavioral decomposition is useful because identifying models that generate the system's steady-state and natural behavior may be posed as regression problems. In particular, steady-state responses are generated by *moment-matching* [5] reduced order models (a.k.a. MM-abstractions) of the system's *steady-state response* to a given time-varying input. The fact that MM-abstractions can be identified using regression techniques [6] means that any recursive least squares algorithm can be used to "learn" or "adapt" these abstractions in an online, data-driven manner.

The natural response of the system is then generated by a *model-following* MF abstraction formed by subtracting the MM-abstraction's output from that of the plant. Taken's theorem [7] justifies the use of delay embeddings of the MF-abstraction's outputs as the system state. With the recurrent behavior subtracted out, the MF-abstraction's qualitative behavior is determined by a hyperbolic fixed point, and so the natural dynamics can be approximated by a Taylor linearization. In particular, we use the dynamic mode decomposition with control (DMDc) algorithm [8] to identify a *linear* state-based model of the natural dynamics. Because the DMDc algorithm is based on singular value decompositions of data matrices, the MF-abstraction can also be updated recursively as new data becomes available.

The fact that the MF-abstraction is linear means that one has a range of methods for designing feedback control laws to regulate the plant's base controller. Prior work has used linear quadratic regulators (LQR) [9] and model predictive controllers (MPC) [10]. This paper uses passivity-based adaption of the hopper's step-length controller to demonstrate robust performance with respect to passive dynamics that were neglected in the initial training of the hopper.

The remainder of the paper is organized as follows. Section II reviews the prior work supporting the behavioral decomposition described above. Section III discusses the use of moment-matching abstractions in capturing the hopper's steady-state behavior. Section IV uses the DMDc algorithm to identify a linear state-based model of the hopper's natural dynamics. Section V demonstrates the use of passivity-based adaptive control on the hopper. Section VI discusses future directions.

II. BEHAVIORAL DECOMPOSITION

Any smooth dynamical system with compact orbits can have its forward trajectories decomposed as the sum of a

M.D. Lemmon, V. Kurtz, and H. Lin are with the Department of Electrical Engineering, University of Notre Dame lemmon@nd.edu, hlin1@nd.edu, vkurtz@nd.edu

P.M. Wensing is with the Department of Aerospace and Mechanical Engineering, University of Notre Dame pwensing@nd.edu

natural behavior and a *steady-state* behavior. This section reviews the background needed to describe this behavioral decomposition of the system's orbits.

Let (X, ϕ) denote a continuous-time dynamical system where $X \subset \mathbb{R}^n$ is compact and $\phi : \mathbb{R} \times X \to X$ is a homeomorphism such that for all $p \in X$ we have $\phi(0, p) = p$ and $\phi(s + t, p) = \phi(t, \phi(s, p))$ for any $s, t \in \mathbb{R}$. X is called the system's *state space* and ϕ is called the system's *transition function*. We define a *trajectory* of (X, ϕ) passing through the point $p \in X$ as the function $\gamma_p : \mathbb{R} \to X$ where $\gamma_p(0) = p$ and $\gamma_p(t) = \phi(t, p)$ for all $t \in \mathbb{R}$. Since all trajectories are contained in the compact set X, we refer to (X, ϕ) as a *closed dynamical system*.

Given a closed dynamical system, (X, ϕ) and a set $B \subset X$, then the positive limit set, $\omega(B)$, of set B consists of all points $q \in X$ for which there exists a sequence $\{p_k, t_k\}_{k=0}^{\infty}$ with $p_k \in B$ and $\{t_k\}_{k=0}^{\infty}$ increasing to infinity such that $\lim_{k\to\infty} \phi(t_k, p_k) = q$. As noted in [11], the trajectories contained in $\omega(X)$ represent what we think of as the system's *steady-state behaviors*.

The positive limit set, $\omega(X)$, is contained in the union of mutually disjoint sets that are referred to as *basic sets* [4]. If the collection of basic sets is countable, then there exists a continuous function $W : X \to [0, 1]$ such that $W(\phi(t, p)) < W(p)$ for all p not in a basic set [12]. When p lies in the jth basic set, then there is a constant c_j such that $W(\gamma_p(t)) = c_j$ for all t. This fact implies that any trajectory, γ_p, of a closed dynamical system can be classified as *recurrent* or *nonrecurrent*. Recurrent trajectories are contained within a single basic set of $\omega(X)$. Nonrecurrent trajectories pass through a point, p, outside of a basic set and these trajectories may therefore be seen as *connecting* two different basic sets, B_i, and B_j, in the sense that $\gamma_p(t) \to B_j$ as $t \to \infty$ and $\gamma_p(t) \to B_i$ as $t \to -\infty$. Recurrent trajectories may therefore be seen as the system's *steady-state behavior*. A non-recurrent trajectory, γ_p, may therefore be seen as asymptotically approaching a recurrent trajectory, say γ_q, on one of the basic sets.

Given a nonrecurrent trajectory, γ_p, that approaches a recurrent trajectory, γ_q, we let $r = p - q$ and then define the system's *natural response* as the function $\gamma_r : \mathbb{R}^+ \to X$ which takes values

$$\gamma_r(t) = \gamma_p(t) - \gamma_q(t), \text{for all } t \in \mathbb{R}^+$$

In the more traditional signals/systems terminology, γ_p for p not in a basic set is the system's *total response*, γ_q, is the associated *steady-state response* and γ_r is the *natural response*. The next two sections discuss how we identify abstractions for a dynamical system's steady-state and natural behavior.

III. MOMENT-MATCHING ABSTRACTION OF STEADY STATE BEHAVIOR

This section uses moment-matching reduced order models [5] to capture the steady-state behavior of an input-output system called the *physical plant*. These reduced order models are called *MM-abstractions*. Let us consider an input-output dynamical system denoted as Σ_c and called the *plant*. The plant has two types of inputs; an *exogenous disturbance*, $\nu : \mathbb{R}^+ \to \mathbb{R}$ and a *control input*, $u : \mathbb{R}^+ \to \mathbb{R}$. The plant's *internal state*, $\mathbf{x} :\to \mathbb{R}^+ \to \mathbb{R}^{n_c}$ and *output* $y : \mathbb{R}^+ \to \mathbb{R}$ satisfy

$$\dot{\mathbf{x}}(t) = \mathbf{F}(\mathbf{x}(t), \nu(t), u(t)), \quad y(t) = \mathbf{G}(\mathbf{x}(t)) \tag{1}$$

where $\mathbf{F} : \mathbb{R}^{n_c} \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}^{n_c}$ is Lipschitz and $\mathbf{G} : \mathbb{R}^{n_c} \to \mathbb{R}$ is continuous.

Since the behavioral decomposition applies to homogeneous systems, we will assume that there is a dynamical system, denoted as Σ_g and called the *generator*, that produces a signal $\begin{bmatrix} \nu(t) \\ u_0 \end{bmatrix}$ at time t where ν is the exogenous disturbance and $u(t) = u_0$ is a reference control input that is driving the plant. To be concrete we assume the disturbance, ν, is generated by a state-based system with an internal state, ω, satisfying

$$\dot{\omega}(t) = \mathbf{S}\omega(t), \quad \nu(t) = \mathbf{L}\omega(t) \tag{2}$$

where $\mathbf{S}$ and $\mathbf{L}$ are appropriately dimensioned constant matrices and $\omega : \mathbb{R}^+ \to \mathbb{R}^{n_a}$ is the generator state. Finally, we assume that the cascaded system, $\Sigma_c\Sigma_g$, formed by having the generator drive the plant produces compact plant state trajectories. With these assumptions, the behavioral decomposition implies that the cascaded system's response, $y : \mathbb{R}^+ \to \mathbb{R}$ converges to a steady-state response $y_{ss} : \mathbb{R}^+ \to \mathbb{R}$.

We are interested in using the plant's observed inputs, ν, and outputs, y, to construct a reduced order model, Σ_a, such that the steady state output of $\Sigma_a\Sigma_g$ equals the steady state output of $\Sigma_c\Sigma_g$. Such abstractions are said to be *moment-matching*. For the generator dynamics in equation (2) a family of moment-matching abstractions is given by [5]

$$\begin{array}{rcl} \dot{\xi}(t) & = & \mathbf{S}\xi(t) + \Delta(\nu(t) - \mathbf{L}\xi(t)) \\ \psi(t) & = & [\mathbf{G} \circ \mathbf{\Pi}]\,(\xi(t)) \end{array} \tag{3}$$

where $\xi : \mathbb{R}^+ \to \mathbb{R}^{n_a}$ and Δ is a matrix such that $\mathbf{S} - \Delta\mathbf{L}$ is Hurwitz. $\mathbf{G} \circ \mathbf{\Pi}$ is the composition of the plant's output map, $\mathbf{G}$, with a function $\mathbf{\Pi} : \mathbb{R}^{n_a} \to \mathbb{R}^{n_c}$ that lifts the abstraction's state into the plant's state space. This composed map, $\mathbf{G} \circ \mathbf{\Pi}$ is called the plant's *moment* with respect to the given generator. When this lifting map satisfies the moment-matching partial differential equation (PDE)

$$\mathbf{F}(\mathbf{\Pi}(\xi), \mathbf{L}\xi) = \frac{\partial \mathbf{\Pi}(\xi)}{\partial \xi}\mathbf{S}\xi \tag{4}$$

then we are guaranteed [5] that the plant's and abstraction's steady-state response to Σ_g are equal.

Determining the lifting function, $\mathbf{\Pi}$, however requires the solution of the moment-matching PDE in equation (4). If the plant is linear, then this PDE reduces to a Sylvester equation. But in either case one needs to know the plant's state-space realization to solve for $\mathbf{\Pi}$. This is an unrealistic expectation as the plant may be so complex that there is no tractable analytical model to work with.

In our setup, however, we know the generator's state equation is embedded in the abstraction's state equation (3). The only thing in the abstraction that depends on the unknown plant is the moment function, $\mathbf{G} \circ \mathbf{\Pi}$. The input to this moment function is the abstraction's state, ξ, which we know. The output from the moment function should be equal to that of the plant's output, y, which we can observe. So the moment function can be learned from observations of the abstraction's state and the plant's output using regression methods. This fact was used in [6] to obtain a data-driven map, $\widehat{\mathbf{G} \circ \mathbf{\Pi}}$, that approximates the true moment map, thereby avoiding the issue of trying to solve the moment matching PDE when we don't know the plant's state space realization.

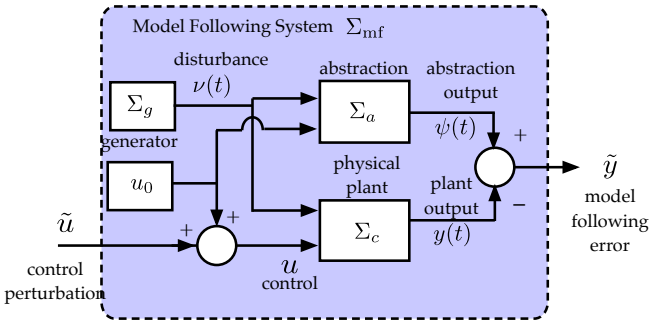

Fig. 1. Model Following System

Fig. 1 is a block diagram illustrating the relationship between the generator, Σ_g, the MM-abstraction, Σ_a, and the plant, Σ_c. The figure shows that these blocks form a *model-following* (MF) system, Σ_{mf} which has the input $\tilde{u}(t)$ that perturbs the constant reference input u_0 being fed into the MM-abstraction. The perturbed control

$$u(t) = u_0 + \tilde{u}(t)$$

is what drives the plant. The difference $\tilde{y}(t) = \psi(t) - y(t)$ is called the model following or tracking error. If the MM-abstraction is truly moment matching then we expect $\tilde{y}(t) \to 0$ as $t \to \infty$. As mentioned above, however, the MM-abstraction actually uses an approximation of the moment map and this means that the model following error may only be bounded in a neighborhood of the origin. A key issue to be addressed in the following sections involves finding the perturbed input, $\tilde{u}(t)$, that ensures the model following error is well regulated.

We now demonstrate how one can learn an MM-abstraction for Raibert's hopper [13]. The hopper consists of a spring-loaded leg connected to a head. We assume the leg is massless. The hopper's motion has a stance stage and a flight stage. The stance stage starts when the hopper's foot touches the ground. The leg spring compresses upon impact and when it reaches its shortest length, the spring constant is changed, thereby injecting energy into the spring and forcing the leg length to increase until the foot leaves the ground (take-off). Take off marks the start of the hopper's flight stage. During this stage, the hopper is airborne and it swings the leg forward to a desired position for a safe landing, after which the robot re-enters the stance stage. We assumed that the energy injected into the spring during the stance stage is constant, but that the desired leg position commanded during the flight stage is given by a step-length controller [3]. This step-length controller adjusts the desired leg position as a function of the forward velocity that the hopper was commanded to follow.

In this example, the hopper is the *plant*, Σ_c, with the slope of the terrain being traversed as the exogenous input, $\nu(t)$, and the commanded forward velocity as the reference input, u_0. For simplicity, we assume the slope of the terrain has two fundamental harmonics with known fundamental periods T_a and T_b. This means that the generator in our example has the state equations

$$\begin{aligned} \dot{\omega}(t) &= \begin{bmatrix} 0 & -\frac{1}{T_a} & 0 & 0 \\ \frac{1}{T_a} & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{T_b} \\ 0 & 0 & \frac{1}{T_b} & 0 \end{bmatrix} \omega(t) = \mathbf{S}\omega(t) \\ \nu(t) &= \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix} \omega(t) = \mathbf{L}\omega(t) \end{aligned}$$

Now that we know the generator's state space realization, we use the $\mathbf{S}$ and $\mathbf{L}$ matrices to write out the abstraction's state equations (3). While any stabilizing gain matrix Δ can be used, we chose Δ as the gains of a steady-state Kalman filter. As noted above, the only thing that remains to be determined is an approximation $\widehat{\mathbf{G} \circ \mathbf{W}}$ of the moment map. The moment map approximation is learned through a regression on the observed plant's output, y, and the abstraction's state, ξ. As suggested in [6], this problem may be solved by constructing data matrices

$$\begin{aligned} \mathbf{Y} &= \begin{bmatrix} y(t_1) & y(t_2) & \cdots & y_{t_N} \end{bmatrix}^T \\ \mathbf{X} &= \begin{bmatrix} \xi(t_1) & \xi(t_2) & \cdots & \xi(t_N) \\ 1 & 1 & \cdots & 1 \end{bmatrix}^T \end{aligned}$$

where $y(t_i)$ is the ith observation of the plant's output at time t_i for $i = 1, \ldots, N$. The $\mathbf{X}$ data matrix has columns formed by stacking the ith abstraction state $\xi(t_i)$ on top of a row of 1's. In a linear regression, we assume there is a vector $\mathbf{a}$ such that $\mathbf{Y} = \mathbf{X}\mathbf{a}$, and so the minimum mean square estimate of $\mathbf{a}$ is

$$\hat{\mathbf{a}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} \tag{5}$$

This estimate, however, may also be computed recursively. In particular, we used the Sherman-Morrison formula to make rank one updates of the inverse in equation (5) every time a new observation of the plant's output and terrain slope, ν was received. The bottom plot in Fig. 2 shows how well this recursive update of the abstraction tracks the plant's forward velocity while traversing the uneven terrain in the top plot. The plot shows the commanded velocity (dashed) u_0, the plant's actual velocity (blue) y, and the abstraction's prediction (red) ψ. From this plot we readily see that after the initial transient in which the abstraction states are converging, there is a small steady-state prediction error $\psi - y$. This steady state error is nonzero because we are using the approximate moment $\widehat{\mathbf{G} \circ \mathbf{\Pi}}$ rather than the true moment.

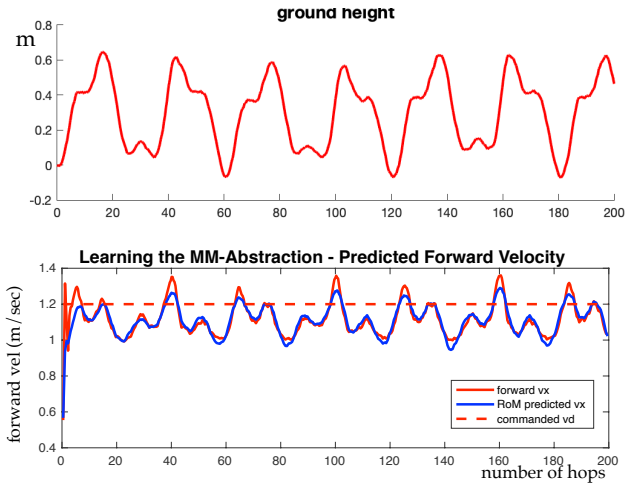

Fig. 2. Performance of MM-abstraction of Raibert's hopper traversing uneven periodic terrain.

IV. MODEL-FOLLOWING ABSTRACTION OF NATURAL RESPONSE

The MM-abstraction, Σ_a, predicts the steady-state behaviors of the plant, Σ_c, to the disturbances produced by the generator, Σ_g, assuming a constant control input, u_0. This is, however, an *open loop* prediction. Prediction errors enter due to the approximate nature of the estimated moment map, $\widehat{\mathbf{G} \circ \mathbf{\Pi}}$, and because the generator's dynamics are not exactly what was assumed in the abstraction's state equation (3). Additional errors occur because some plant modes that were neglected in training Σ_a were later excited in the real world. This means we need to "close" the loop to ensure the model-following error $\tilde{y}(t) = \psi(t) - y(t)$ remains sufficiently bounded in the presence of these unmodeled effects. One particular feedback architecture is shown in Fig. 3, which is discussed at greater length in Section V. Closing the loop requires us to construct a model for the MF-system, Σ_{mf}, as shown in Fig. 3. We call this model the *MF-abstraction* and show that a *linear* MF-abstraction can also be learned in a data driven manner.

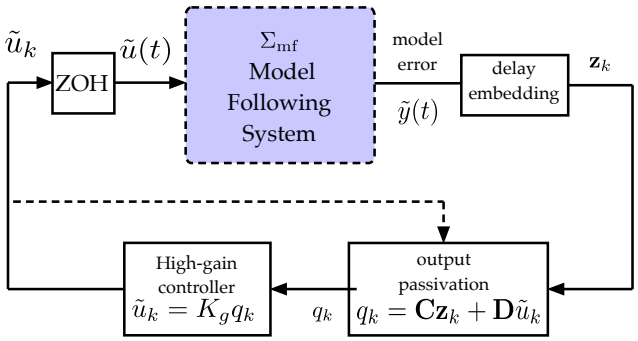

Fig. 3. Passive feedback controller used to regulate the MF-system, Σ_{mf}.

To establish the feasibility of using linear abstractions, we recall from section II that any plant trajectory may be classified as recurrent or nonrecurrent. A recurrent trajectory is contained within a basic set of the system's positive limit set. A nonrecurrent trajectory connects to different basic sets and asymptotically it converges to one of the steady-state behaviors. Since the MF-system, Σ_{mf} subtracts this steady-state behavior from the nonrecurrent trajectory, the corresponding *natural response* is governed by a hyperbolic fixed point which means the flows of the MF-system are qualitatively the same as that of its Taylor linearization. In other words, the natural dynamics of the plant can be captured by a *linear state-space* model thereby allowing one to use a number of linear control methods to regulate the MF-system's tracking error, $\tilde{y}$.

We used the dynamic mode decomposition with control (DMDc) algorithm [8] to learn a linear MF-abstraction for Σ_{mf}. This algorithm takes the state of the given system and its input to solve a regression problem yielding a linear state space model. In particular, let $\mathbf{z}(t)$ denote the internal state of the MF system at time t. Consider a finite set of times $\{t_k\}_{k=1}^N$ at which one samples the system's state and input. Let $\mathbf{z}_k$ and $\tilde{u}_k$ denote the sample state and input, respectively, at time t_k. We are interested in finding a matrix $\mathbf{K} = \begin{bmatrix} \mathbf{K}_{11} & \mathbf{K}_{12} \\ \mathbf{K}_{21} & \mathbf{K}_{22} \end{bmatrix}$ that minimizes the mean squared prediction error

$$\sum_{k=1}^{N-1} \left| \begin{bmatrix} \mathbf{z}_{k+1} \\ \tilde{u}_{k+1} \end{bmatrix} - \begin{bmatrix} \mathbf{K}_{11} & \mathbf{K}_{12} \\ \mathbf{K}_{21} & \mathbf{K}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{z}_k \\ \tilde{u}_k \end{bmatrix} \right|^2$$

where N is the length of the data record. The matrix $\mathbf{K}$ may be found from the data matrices

$$\mathbf{Z} = \begin{bmatrix} \mathbf{z}_1 & \mathbf{z}_2 & \cdots & \mathbf{z}_{N-2} & \mathbf{z}_{N-1} \\ \tilde{u}_1 & \tilde{u}_2 & \cdots & \tilde{u}_{N-2} & \tilde{u}_{N-1} \end{bmatrix}$$

$$\mathbf{Z}^+ = \begin{bmatrix} \mathbf{z}_2 & \mathbf{z}_3 & \cdots & \mathbf{z}_{N-1} & \mathbf{z}_N \\ \tilde{u}_2 & \tilde{u}_3 & \cdots & \tilde{u}_{N-1} & \tilde{u}_N \end{bmatrix}$$

The desired matrix minimizing the mean squared prediction error is

$$\mathbf{K} = \mathbf{Z}^{+}\mathbf{Z}^{T}(\mathbf{Z}^{T}\mathbf{Z})^{-1}$$

The DMDc algorithm [8] computes the above inverse from the singular value decomposition of the data matrix. The resulting linear system model then has the from

$$\mathbf{z}_{k+1} \quad = \quad \mathbf{K}_{11}\mathbf{z}_k + \mathbf{K}_{12}\tilde{u}_k$$

This algorithm has previously been used to obtain reduced order representations of Koopman operators [14].

The DMDc algorithm outlined above presumes state accessibility. In many cases, however, this is not a realistic assumption and we will only have access to the inputs and outputs. Provided the system is observable, then Taken's embedding theorem [7] allows us to use a vector formed from past sampled outputs as the system state,

$$\mathbf{z}_k = \begin{bmatrix} y(t_{k-N_e}) & y(t_{k-N_e+1}) & \cdots & y(t_{k-1}) & y(t_k) \end{bmatrix}^T$$

provided the embedding dimension, N_e, is sufficiently large. We refer to this as the *delay-embedding* of the output. Fig. 3 shows that the continuous-time output of the MF-system, $\tilde{y}(t)$, is passed though a delay embedding block to create the discrete-time state vector $\mathbf{z}_k$. This state is then used in a feedback controller to create the control input $\tilde{u}_k$. This control is also discrete-time and Fig. 3 shows a zero order hold (ZOH) is used to convert this discrete-time signal into the continuous-time input used by Σ_{mf}.

We now present initial results showing how well our DMDc learned MF-abstraction worked on the hopper. For the hopper, the command $u(t)$ is the forward speed, and the output is the difference between the hopper's actual forward speed and the speed predicted by the MM-abstraction. To generate the data used by the DMDc algorithm, we simulated the MF-system over the terrain shown in Fig. 2. The control input was chosen to periodically disturb the nominal command, u_0 every T hops,

$$u(t) = u_0 - \sum_{k=0}^{\infty} a_k e^{-\lambda_k(t-kT)} u_s(t-kT)$$

where u_s is a unit step function and (a_k, λ_k) are randomly chosen parameters characterizing the size and duration of the perturbation. The resulting training data was then used in the DMDc algorithm to identify a linear MF-abstraction for $\mathbf{\Sigma}_{\text{mf}}$.

The testing performance of a trained MF-abstraction is shown in Fig. 4. The top plot shows the MF-system's output, $\tilde{y}(t)$, and the MF-abstraction's prediction of that output. The bottom plot shows the testing error. The MF-abstraction's testing error is nearly zero except at those points where there is a discontinuous jump in the input. These results, therefore, suggest that the MF-abstraction accurately predicts the natural dynamics of the plant. The following section uses this abstraction to design an adaptive passivity-based controller (see Fig. 3) for the hopper.

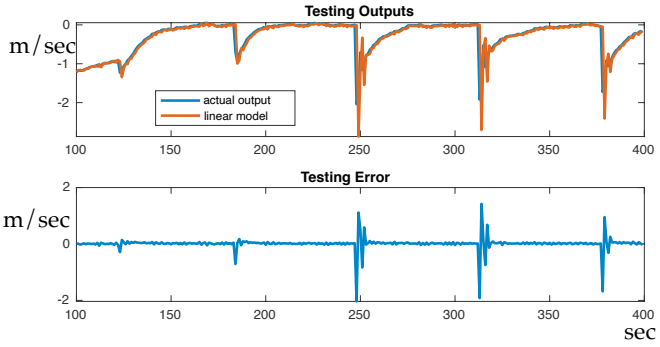

Fig. 4. Testing Data and Performance of the DMDc linear model of the Hopper's MF-system

V. PASSIVE ADAPTIVE HOPPING

Because the MF-abstraction is linear, there are several linear control techniques one can use to design the controller. Prior work focused on linear quadratic regulators (LQR) [9] and model predictive controllers [10], all of which seek an optimal control law. This section uses a passivity-based approach to regulating the hopper's model-following system. The objective of this control is to force the plant's forward velocity to track the MM-abstraction's predicted forwarded velocity. We demonstrate that the performance of this controlled system is robust to unmodeled passive dynamics.

We consider a modification of Raibert's hopper shown in Fig. 5. The template for this hopper is a triple inverted pendulum whose first link is the leg, second link is the head, and third link is a "load" attached to the head through a spring as shown in Fig. 5. This "load" has sufficient mass so that when it swings about its pivot it moves the leg/head. In particular, if the spring constant is small then the hopper is a bobble head and when the spring is stiff then the hopper has a stiff head.

We used the DMDc algorithm from section IV to identify a discrete-time linear MF-abstraction for the hopper. We obtain a passive input-output system by defining a virtual output, q_k, shown in Fig. 3, that takes values

$$q_k = \mathbf{C}\mathbf{z}_k + \mathbf{D}\tilde{u}_k$$

This input/output system's state-space realization, $\left[\begin{array}{c|c} \mathbf{A} & \mathbf{B} \\ \hline \mathbf{C} & \mathbf{D} \end{array}\right]$, will be

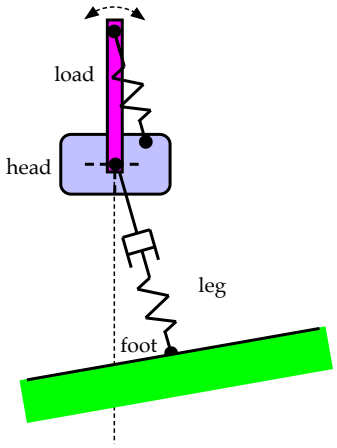

Fig. 5. Bobble Headed Hopper

strictly passive if there exists a symmetric positive definite matrix, $\mathbf{P}$, such that [15]

$$\begin{bmatrix} \mathbf{P}-\mathbf{A}^T\mathbf{P}\mathbf{A} & \mathbf{C}^T-\mathbf{A}^T\mathbf{P}\mathbf{B} \\ \mathbf{C}-\mathbf{B}^T\mathbf{P}\mathbf{A} & \mathbf{D}+\mathbf{D}^T-\mathbf{B}^T\mathbf{P}\mathbf{B} \end{bmatrix} > 0 \qquad (6)$$

We need to find the $\mathbf{C}$ and $\mathbf{D}$ matrices that satisfy the linear matrix inequality (6). Provided the MF-system is zero-state observable, then control inputs of the form $\tilde{u}_k = -K_g q_k$ would asymptotically stabilize the hopper for any $K_g > 0$.

The matrices $\mathbf{C}$ and $\mathbf{D}$ are found as follows. We first determine a symmetric positive definite matrix, $\mathbf{P}$, that satisfies the discrete-time Lyapunov equation. We then let $\mathbf{C} = \mathbf{B}^T\mathbf{P}\mathbf{A}$ and choose $\mathbf{D}$ sufficiently large so that $\mathbf{D}+\mathbf{D}^T-\mathbf{B}^T\mathbf{P}\mathbf{B} > 0$ These choices force the left hand side of equation (6) to be a block diagonal matrix whose blocks are all positive definite.

We used the recursive learning algorithms described in sections III and IV to identify MM and MF abstractions for a bobble head hopper whose spring constant was stiff enough to prevent bobbling. A passive control law was then designed for the MF-abstraction. We then tested our system by having the hopper traverse the uneven terrain shown in the top plot of Fig. 6. For the first 45 seconds of this test scenario the hopper had a stiff head, after 45 seconds the spring constant was changed so the head could bobble. The second plot in Fig. 6 shows that the feedback control keeps the hopper moving forward even though oscillatory bursts occur every time the hopper transitions from hopping downhill to hopping uphill.

An important feature of passivity-based control is that it often reduces to a high-gain feedback strategy whose performance can be improved by simply increasing the gain, K_g. We tested this aspect of the controller by adaptively changing K_g so the hopper learns to more effectively regulate its MF-abstraction even though we trained with a stiff headed hopper. In particular, we used the signed version of the MIT rule to update K_g using the update law $K_{k+1} = K_k + \gamma \mathrm{sgn}(\tilde{y}_k)\tilde{y}_k^2$ where K_{k+1} is the gain updated using the model following error $\tilde{y}_k = \psi_k - y_k$ at time k. The bottom plot in Fig. 6 shows the forward velocity for the adaptively controlled hopper. Comparing this to the middle plot where no adaptation was done, we can readily see that each oscillatory burst changes the gain in a way that reduces the severity of subsequent bursts.

VI. FUTURE DIRECTIONS

This paper reports on recent work demonstrating an approach for learning how to hop across uneven terrain. The approach assumes there is a base controller that ensures the plant's orbits are compact and that the inputs driving the plant are generated by a dynamical system Σ_g, whose dynamics are approximately known. From this information,

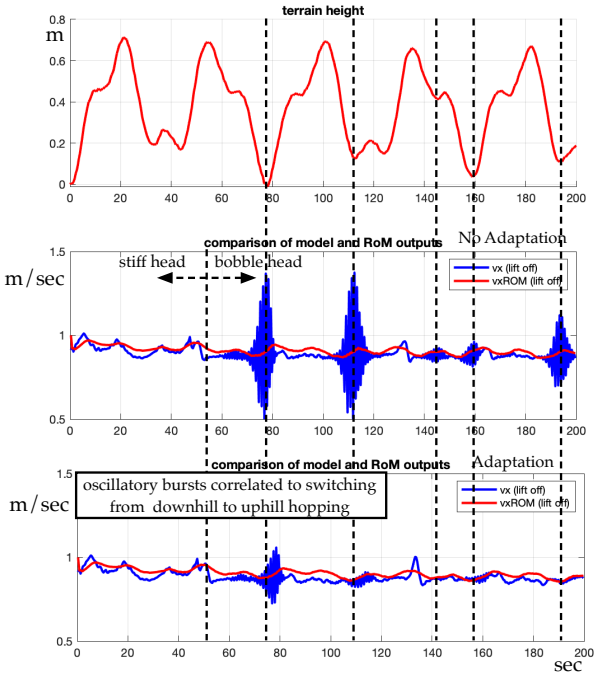

Fig. 6. Adaptive Improvement of Bobble Head through MIT rule

we demonstrate a two-step process in which input-output data from be step-length controlled hopper is used to 1) train a moment-matching abstraction of the plant's steady-state behavior and once this abstraction is known we use 2) the DMDc algorithm to learn a linear state-based model of the hopper's natural dynamics. These abstractions were used to design a high-gain controller that adapted to passive dynamics neglected during training.

The benefit of the proposed approach is that it avoids the offline training techniques found in popular machine learning methods such as deep reinforcement learning. This occurs because the MM-abstraction and MF-abstraction can be learned through regression methods that lend themselves to online recursive implementation. The approach may be seen as an example of structured risk management (SRM) [16] in which the model structure can be explained in terms of the plant's *basic decomposition* into steady-state and natural behaviors.

Future work is examining the extent to which this framework can be applied to identifying and managing other complex dynamical systems. We are currently working to extend this approach to a robotic quadruped. Related work with another group is applying this approach to wireless network congestion control algorithms to changes in link reliability.

REFERENCES

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski

et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[2] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International Conference on Machine Learning Proceedings*, ser. Proceedings of Machine Learning Research, vol. 37, 2015, pp. 1889–1897.

[3] J. Hodgins and M. Raibert, "Adjusting step length for rough terrain locomotion," *IEEE Transactions on Robotics Automation*, vol. 7, no. 3, pp. 289–298, 1991.

[4] R. W. Easton, *Geometric methods for discrete dynamical systems*. Oxford University Press on Demand, 1998, no. 50.

[5] A. Astolfi, "Model reduction by moment matching for linear and nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 55, no. 10, pp. 2321–2336, 2010.

[6] G. Scarciotti and A. Astolfi, "Data-driven model reduction by moment matching for linear and nonlinear systems," *Automatica*, vol. 79, pp. 340–351, 2017.

[7] F. Takens, "Detecting strange attractors in turbulence," in *Dynamical systems and turbulence, Warwick 1980*. Springer, 1981, pp. 366–381.

[8] J. L. Proctor, S. L. Brunton, and J. N. Kutz, "Dynamic mode decomposition with control," *SIAM Journal on Applied Dynamical Systems*, vol. 15, no. 1, pp. 142–161, 2016.

[9] E. Kaiser, J. N. Kutz, and S. L. Brunton, "Data-driven discovery of koopman eigenfunctions for control," *Machine Learning: Science and Technology*, vol. 2, no. 3, p. 035023, 2021.

[10] M. Korda and I. Mezić, "Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control," *Automatica*, vol. 93, pp. 149–160, 2018.

[11] A. Isidori and C. I. Byrnes, "Steady-state behaviors in nonlinear systems with an application to robust disturbance rejection," *Annual Reviews in Control*, vol. 32, no. 1, pp. 1–16, 2008.

[12] C. C. Conley, *Isolated invariant sets and the Morse index*. American Mathematical Soc., 1978.

[13] M. H. Raibert, *Legged robots that balance*. MIT press, 1986.

[14] S. Brunton, B. Brunton, J. Proctor, and J. Kutz, "Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control," *PLoS ONE*, vol. 11, no. 2-e0150171, 2016.

[15] S.-P. Wu, S. Boyd, and L. Vandenberghe, "Fir filter design via semidefinite programming and spectral factorization," in *Proceedings of 35th IEEE Conference on Decision and Control*, vol. 1. IEEE, 1996, pp. 271–276.

[16] V. Vapnik, *Statistical Learning Theory*. John Wileu & Sons, 1998.