IFAC Journal of Systems and Control 35 (2026) 100360

Contents lists available at ScienceDirect

IFAC Journal of Systems and Control

journal homepage: www.elsevier.com/locate/ifacsc

Full length article

Symbolic learning of interpretable reduced-order models for jumping quadruped robots[☆], [☆☆]

Gioele Buriani [a,1], Jingyue Liu [a,1], Maximilian Stölzle [a,b], Cosimo Della Santina [a,c], Jiatao Ding [d,*]

[a] Department of Cognitive Robotics, Delft University of Technology, Building 34, Mekelweg 2, 2628 CD Delft, Netherlands
[b] Computer Science and Artificial Intelligence Laboratory (CSAIL), Massachusetts Institute of Technology, Cambridge, MA 02139, USA
[c] Institute of Robotics and Mechatronics, German Aerospace Center (DLR), 82234 Wessling, Germany
[d] Department of Industrial Engineering, University of Trento, Italy

ARTICLE INFO

Article history:
Received 4 August 2025
Received in revised form 11 December 2025
Accepted 31 December 2025
Available online 8 January 2026

Keywords:
Quadruped jumping
Symbolic regression
Autoencoders
Interpretability
Reduced-order model

ABSTRACT

Reduced-order models are central to motion planning and control of quadruped robots, yet existing templates are often hand-crafted for a specific locomotion modality. This motivates the need for automatic methods that extract task-specific, interpretable low-dimensional dynamics directly from data. We propose a methodology that combines a linear autoencoder with symbolic regression to derive such models. The linear autoencoder provides a consistent latent embedding for configurations, velocities, accelerations, and inputs, enabling the sparse identification of nonlinear dynamics (SINDy) to operate in a compact, physics-aligned space. A multi-phase, hybrid-aware training scheme ensures coherent latent coordinates across contact transitions. We focus our validation on quadruped jumping—a representative, challenging, yet contained scenario in which a principled template model is especially valuable. The resulting symbolic dynamics outperform the state-of-the-art handcrafted actuated spring-loaded inverted pendulum (aSLIP) baseline in simulation and hardware across multiple robots and jumping modalities.

1. Introduction

Legged robots are now pervasive systems, deployed in applications ranging from planetary exploration (Arm et al., 2023) to search-and-rescue operations (Bai et al., 2018). Yet, despite this growing presence, their dynamics remain remarkably difficult to model—especially when seeking a description expressive enough to capture essential behaviors and lightweight enough for real-time use.

Classic Lagrangian and Newton–Euler formulations provide full-order models of legged systems (Bellicoso et al., 2017; Di Carlo et al., 2018; Ding, Posthoorn, et al., 2024; Yan et al., 2021), but these models remain too computationally heavy for online deployment (Li et al., 2023). This has led to reduced-order templates such as spring-loaded inverted pendulum (SLIP) (Blickhan, 1989) and single rigid body dynamics (SRBD) (Hong et al., 2020), which capture dominant behaviors at far lower cost. Yet these templates lose accuracy in multi-contact transitions, rapid reconfiguration, and complex terrain (Carpentier & Mansard, 2018). In practice, one must choose between high-dimensional models that are difficult to deploy and compact, task-specific heuristics. A principled way to automatically extract reduced-order models—accurate, compact, and interpretable—would significantly advance the analysis and control of hybrid locomotion dynamics (Ding et al., 2025). However, such a method is still missing.

In this paper, we turn to interpretable data-driven techniques to address this challenge. Data-driven learning has shown strong capabilities in modeling physical systems, with numerous works leveraging neural networks to capture complex dynamics (Calandra et al., 2015; Hashemi et al., 2023; Ogunmolu et al., 2016; Rajendra & Brahmajirao, 2020). Dimensionality-reduction methods such as autoencoders extract low-dimensional latent features (Gonzalez & Balajewicz, 2018; Hinton & Salakhutdinov, 2006; Refinetti & Goldt, 2022), after which multi-layer perceptrons (MLPs) or recurrent neural networks (RNNs) can model reduced-order dynamics—often gaining accuracy at the expense of interpretability. Physics-informed approaches, including physics-informed neural networks (PINNs) (Raissi et al., 2019), deep Lagrangian networks (DeLaN) (Liu et al., 2024; Lutter et al.,

[☆] This article is part of a Special issue entitled: ‘IFAC WC 2026 - TC1.1’ published in IFAC Journal of Systems and Control.

[☆☆] This work is supported by the European Union’s Horizon Europe Program Project EMERGE - Grant Agreement No. 101070918.

[*] Corresponding author.

E-mail addresses: gioele.buriani@perciv.ai (G. Buriani), J.Liu-14@tudelft.nl (J. Liu), mstolzle@mit.edu (M. Stölzle), C.DellaSantina@tudelft.nl (C. Della Santina), jiatao.ding@unitn.it (J. Ding).

[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.ifacsc.2025.100360

2019), and coupled oscillator networks (Stölzle & Santina, 2024), improve interpretability, data efficiency, and out-of-distribution performance (Stölzle, 2025). Still, none of these methods handle the hybrid, time-dependent structure inherent to locomotion. This limitation is particularly acute for floating-base dynamics, where contacts induce abrupt changes in inertial and potential forces—effects that current learning frameworks are not designed to capture.

Given this gap, no existing work provides a systematic method for learning reduced-order models of legged systems. We address this by combining linear autoencoders with symbolic regression, specifically the sparse identification of nonlinear dynamics (SINDy) (Brunton et al., 2016; Pandey et al., 2024), extending prior autoencoder-based formulations (Champion et al., 2019) to quadruped dynamics, where such techniques remain unexplored.

The linear autoencoder is essential: its constant Jacobian—equal to its weight matrix—provides a unified linear mapping of configurations, velocities, accelerations, and inputs, enabling consistent and interpretable identification of reduced-order dynamics. Direct identification in the observation space is impractical due to scale disparities and dimensionality, while nonlinear autoencoders introduce state-dependent latent geometries and distort physical structure through separate nonlinear mappings of positions and velocities.

To obtain a representation valid across hybrid phases, we adopt a multi-phase training scheme: train the encoder in the contact phase (launch + landing), refine the decoder across all phases, and finally identify SINDy parameters with the autoencoder frozen. We validate the approach on jumping motions, a demanding form of dynamic locomotion that enables navigation in complex environments (Atanassov et al., 2024; Ding et al., 2023; Nguyen et al., 2019; Panichi et al., 2025b) yet remains challenging due to high accelerations and impact-driven dynamics.

In summary, this work contributes to legged robotics and physics-informed machine learning through:

- A physics-aligned linear latent embedding that provides consistent linear mappings of positions, velocities, accelerations, and inputs, enabling symbolic learning of reduced-order and interpretable jumping dynamics.
- A sequential multi-phase training strategy that preserves coherent latent coordinates and symbolic models across the hybrid phases of a jump.
- The first demonstration of learning reduced-order dynamics for legged locomotion, with evaluation across different quadruped robots and jumping modalities.

2. Preliminary

2.1. Robot description

Traditional modeling approaches represent the quadruped robot as a movable base (i.e., the body) with kinematic chains (i.e., the legs) attached, as shown in Fig. 1.

As illustrated in Fig. 1, the Unitree Go1, for example, exhibits 12 degrees of freedom (DOFs) for the limbs alone, with 6 underactuated DoFs for the floating base's position and orientation relative to an inertial frame (Righetti et al., 2011). The configuration space of quadruped robots is typically parameterized as (Mistry et al., 2010)

$$\mathbf{q} = \begin{bmatrix} \mathbf{q}_\mathrm{j}^\top & \mathbf{q}_\mathrm{b}^\top \end{bmatrix}^\top \in \mathbb{R}^{m+6}, \tag{1}$$

where $\mathbf{q}_\mathrm{j} \in \mathbb{R}^m$ contains the coordinates of the actuated joints, with m representing the number of joints, and $\mathbf{q}_\mathrm{b} \in SE(3)$ captures the coordinates of the unactuated floating base, including Cartesian positions $(x, y, z) \in \mathbb{R}^3$ and Euler angles $(\phi, \theta, \psi) \in SO(3)$. The constrained dynamics follow

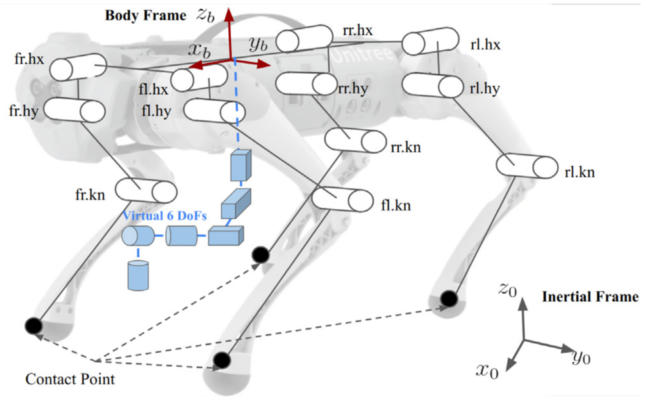


Fig. 1. Quadruped robot with labeled joints and body frame.

$$\mathbf{M}(\mathbf{q})\,\ddot{\mathbf{q}} + \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{S}^\top \boldsymbol{\tau} + \mathbf{J}_\mathrm{C}^\top(\mathbf{q})\,\boldsymbol{\lambda}, \tag{2}$$

where $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{(m+6)\times(m+6)}$ is the mass matrix, $\mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^{m+6}$ captures centripetal, Coriolis, and gravitational forces, $\mathbf{S} = \begin{bmatrix} \boldsymbol{I}_{m\times m} & \mathbf{0}_{m\times 6} \end{bmatrix}$ is the selection matrix for actuated joints, $\boldsymbol{\tau} \in \mathbb{R}^m$ denotes actuation torques, $\mathbf{J}_\mathrm{C} \in \mathbb{R}^{k\times(m+6)}$ is the Jacobian matrix for k linearly independent constraints, and $\boldsymbol{\lambda} \in \mathbb{R}^k$ represents the constraint forces.

Consider a quadruped robot where ground reaction forces (GRFs) constitute the dominant constraint forces acting on the body. For each leg (modeled as an independent kinematic chain), the GRF $\mathbf{F}_k \in \mathbb{R}^3$ at the kth foot can be computed from the joint torques $\boldsymbol{\tau}_k$ through the leg Jacobian transpose

$$\mathbf{F}_k = \mathbf{J}_k^{-\top} \boldsymbol{\tau}_k, \tag{3}$$

where $\mathbf{J}_k \in \mathbb{R}^{3\times n_j}$ is the Jacobian of the kth leg ($n_j = m/4$ being the number of joints per leg).

The net effect of GRFs on the robot's center of mass (CoM) dynamics can be described by the time derivatives of the linear and angular momentum of the CoM

$$\dot{\boldsymbol{\mathcal{P}}} = \sum_{k=1}^{4} \mathbf{F}_k, \quad \dot{\boldsymbol{L}} = \sum_{k=1}^{4} ((\mathbf{p}_k - \mathbf{b}) \times \mathbf{F}_k), \tag{4}$$

where $\boldsymbol{\mathcal{P}} \in \mathbb{R}^3$ and $\boldsymbol{L} \in \mathbb{R}^3$ are the linear and angular momentum, respectively. Here, $\mathbf{p}_k \in \mathbb{R}^3$ denotes the world-frame position of the kth leg, and $\mathbf{b} \in \mathbb{R}^3$ is the CoM position in world coordinates.

The generalized wrench $\boldsymbol{\mathcal{F}} \in \mathbb{R}^6$ that acts on the body frame is

$$\boldsymbol{\mathcal{F}} = \begin{bmatrix} \dot{\boldsymbol{\mathcal{P}}}^\top & \dot{\boldsymbol{L}}^\top \end{bmatrix}^\top \in \mathbb{R}^6, \tag{5}$$

which leads to the complete system input formulation

$$\mathbf{u} = \begin{bmatrix} \boldsymbol{\tau}^\top & \boldsymbol{\mathcal{F}}^\top \end{bmatrix}^\top \in \mathbb{R}^{m+6}. \tag{6}$$

2.2. Encoder–decoder representation learning

An autoencoder, i.e., encoder–decoder architecture, enables one to obtain a compact but informative representation of high-dimensional state variables (Hinton & Salakhutdinov, 2006). The encoder $E(\cdot)$ maps the original variable $\boldsymbol{x} \in \mathbb{R}^n$ into a low-dimensional latent vector $\boldsymbol{z} = E(\boldsymbol{x}) \in \mathbb{R}^r$ with $r \ll n$, effectively capturing the essential features that characterize the system's dynamics. The decoder $D(\cdot)$ reconstructs the high-dimensional signal from the latent representation, i.e., $\hat{\boldsymbol{x}} = D(\boldsymbol{z})$, ensuring that

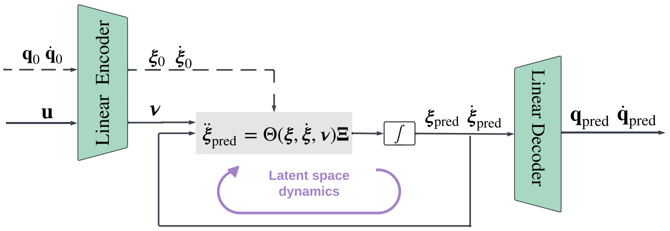

$\mathbf{q}_0\ \dot{\mathbf{q}}_0$
$\mathbf{u}$
Linear Encoder
$\xi_0\ \dot{\xi}_0$
$\boldsymbol{\nu}$
$\ddot{\xi}_{\text{pred}} = \Theta(\xi, \dot{\xi}, \nu)\Xi$
$\int$
$\xi_{\text{pred}}\ \dot{\xi}_{\text{pred}}$
Linear Decoder
$\mathbf{q}_{\text{pred}}\ \dot{\mathbf{q}}_{\text{pred}}$
Latent space dynamics

Fig. 2. Methodology overview: Symbolic latent-space dynamics $\ddot{\boldsymbol{\xi}} = \Theta(\boldsymbol{\xi}, \dot{\boldsymbol{\xi}}, \boldsymbol{\nu})\,\Xi$ are learned to predict the evolution of the full-state system from $(\mathbf{q}, \dot{\mathbf{q}})$.

the latent space preserves the underlying structure of the original data. This framework not only reduces computational complexity but also provides a smooth and continuous latent manifold that is amenable to downstream tasks such as policy learning.

2.3. Sindy

As a typical symbolic learning, also often referred to as symbolic regression, approach, SINDy (Brunton et al., 2016) enables us to identify symbolic representations of the dynamics by constructing a symbolic library of candidate functions $\Theta(\boldsymbol{\xi}, \dot{\boldsymbol{\xi}}, \boldsymbol{\nu})$, including polynomial and trigonometric functions of latent variables ($\boldsymbol{\xi}$), their derivatives ($\boldsymbol{\xi}^{(n)}$), and the transformed control inputs ($\boldsymbol{\nu}$). This library represents a comprehensive set of terms that might appear in the system's governing equations. The objective is to represent the dynamics of the latent variables as a sparse combination of these candidate functions. Mathematically, this is expressed as a sparse regression problem, with the latent dynamics expressed as

$$\ddot{\boldsymbol{\xi}}_{\text{pred}} = \Theta(\boldsymbol{\xi}, \dot{\boldsymbol{\xi}}, \boldsymbol{\nu})\,\Xi, \tag{7}$$

where $\Xi \in \mathbb{R}^{p\times l}$ is a sparse matrix of coefficients to be determined, with p being the number of the candidate functions in $\Theta(\boldsymbol{\xi}, \dot{\boldsymbol{\xi}}, \boldsymbol{\nu})$ and l being the dimension of the latent space. The optimization problem is typically solved using sequential thresholding methods to ensure a sparse solution (Brunton et al., 2016; Champion et al., 2019). Promotes sparsity by penalizing the ℓ_1-norm of Ξ, which encourages most entries in Ξ to be zero. This sparsity-promoting optimization ensures that only a minimal set of candidate functions is selected, resulting in an interpretable model.

3. Methodology

3.1. Proposed learning architecture

We combine linear autoencoders with SINDy to model jumping dynamics in the latent space, achieving dimensionality reduction while preserving the essential dynamic characteristics, enabling efficient representation of complex jumping motions. Fig. 2 provides an overview of the encoded latent space and the established dynamics loop.

Fig. 3 details the core components, including the encoder–decoder structure, latent space dynamics prediction, and real-space dynamics reconstruction. That is, we encode the initial state into a latent space, predict the dynamics in this latent space, and decode them back to the full-order system space. In particular, we ensure accurate reconstruction of system states and dynamical modeling through specialized loss components.

To maintain direct interpretability, we adopt the linear autoencoder to reduce the dimensionality of the state space while preserving a linear relationship with the original space. The autoencoder consists of an encoder $(\mathbf{W}_{\text{e}}, \mathbf{B}_{\text{e}})$ and a decoder $(\mathbf{W}_{\text{d}}, \mathbf{B}_{\text{d}})$. The encoder's weights and bias are $\mathbf{W}_{\text{e}} \in \mathbb{R}^{l\times(m+6)}$ and $\mathbf{B}_{\text{e}} \in \mathbb{R}^{l}$, while the decoder's weights and bias are $\mathbf{W}_{\text{d}} \in \mathbb{R}^{(m+6)\times l}$ and $\mathbf{B}_{\text{d}} \in \mathbb{R}^{m+6}$, where l is the dimension of the desired latent space.

With this encoder, we map the robot's configuration $\mathbf{q}$ and its derivatives $\mathbf{q}^{(n)}$ into a lower-dimensional latent space using the linear transformation

$$\boldsymbol{\xi}^{(n)} = \mathbf{W}_{\text{e}}\,\mathbf{q}^{(n)} + \delta_{n0}\,\mathbf{B}_{\text{e}}, \quad n = 0, 1, 2, \tag{8}$$

where $\boldsymbol{\xi} \in \mathbb{R}^{l}$ represents the latent representation, $\boldsymbol{\xi}^{(n)}$ denotes the nth derivative of $\boldsymbol{\xi}$, and δ_{n0} is the Kronecker delta, equal to 1 when $n = 0$ and 0 otherwise. The system's actuation is mapped into latent space as

$$\boldsymbol{\nu} = \mathbf{W}_{\text{e}}^{\dagger\top}\mathbf{u}, \ \ \boldsymbol{\nu} \in \mathbb{R}^{l}. \tag{9}$$

After establishing the latent space mapping, we employ the SINDy algorithm to identify symbolic representations of the latent dynamics (see Fig. 3(b)). Particularly, we follow Eq. (7) to predict the accelerations. Subsequently, the predicted accelerations $\ddot{\boldsymbol{\xi}}_{\text{pred}}$ are transformed back to the original state space by the linear decoder

$$\ddot{\mathbf{q}}_{\text{pred}} = \mathbf{W}_{\text{d}}\,\ddot{\boldsymbol{\xi}}_{\text{pred}}. \tag{10}$$

For training the encoder, decoder, and reduced-order symbolic dynamics, we consider the following loss components:

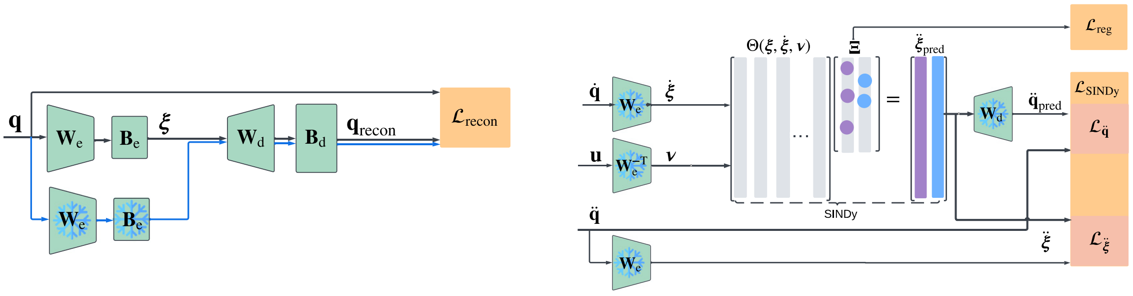
- $\mathcal{L}_{\text{recon}} = \|\mathbf{q} - \mathbf{q}_{\text{recon}}\|_2^2$ ensures accurate reconstruction of the original state. The reconstructed configuration $\mathbf{q}_{\text{recon}}$ is given by
$$\mathbf{q}_{\text{recon}} = \mathbf{W}_{\text{d}}\,(\mathbf{W}_{\text{e}}\mathbf{q} + \mathbf{B}_{\text{e}}) + \mathbf{B}_{\text{d}}. \tag{11}$$
- $\mathcal{L}_{\text{SINDy}} = \|\ddot{\boldsymbol{\xi}} - \ddot{\boldsymbol{\xi}}_{\text{pred}}\|_2^2 + \left\|\ddot{\mathbf{q}} - \ddot{\mathbf{q}}_{\text{pred}}\right\|_2^2$ captures the prediction loss, ensuring that the learned dynamics in the latent space and in the reconstructed space are accurate.
- $\mathcal{L}_{\text{reg}} = \|\Xi\|_1$ promotes sparsity in SINDy coefficients.

3.2. Multi-phase training strategy

Jumping motion poses unique modeling challenges due to its hybrid dynamical nature. This study considers two types of jumping: pronking and froggy jumps.

In the pronking jump, the robot launches and lands with full limb contact simultaneously. This motion consists of two dominant phases: (1) a contact phase, where all legs remain grounded during takeoff and landing, and (2) a flight phase, where the robot is entirely airborne. Conversely, the froggy jump consists of an initial phase where all legs are grounded, followed by a transition phase where only the rear legs remain in contact with the ground, and then progressing to a flight phase.

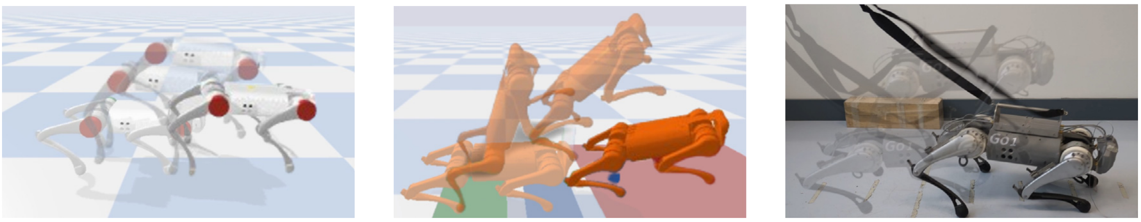
To effectively model the hybrid dynamics, we propose a structured training methodology (Fig. 3) that combines transfer learning principles (Torrey & Shavlik, 2010) with phase-specific optimization. First, we train the encoder and decoder statically on reconstructing the configurations of the full contact phase by minimizing the $\mathcal{L}_{\text{recon}}$ loss. Then, the encoder weights are frozen, and the decoder is trained on all motion phases, including the flight phase, while still minimizing $\mathcal{L}_{\text{recon}}$. Finally, with the encoder and decoder weights fixed, we train the symbolic dynamics for each motion phase (contact, partial contact, and flight) by minimizing the $\mathcal{L}_{\text{SINDy}}$ and $\mathcal{L}_{\text{reg}}$ loss terms. In this way, the final model is physically coherent and can accurately reconstruct the robot's motion across different phases.



(a) Training of encoder and decoder in a static setting

(b) Training of symbolic latent dynamics in a dynamic setting

Fig. 3. Flowchart of the pipeline. **(a)** The linear autoencoder is first trained on the contact phase dataset, minimizing the reconstruction loss, $\mathcal{L}_{\text{rec}}$, depicted by the black line. Subsequently, by fixing the encoder, the decoder is fine-tuned on a dataset that includes all motion phases (see the blue line). **(b)** In the third step, we train, for each motion phase separately, a symbolic dynamical model $\ddot{\boldsymbol{\xi}}_{\text{pred}}$ in latent space. Here, we employ the SINDy approach to regress a sparse matrix Ξ that selects and scales basis functions from the library $\Theta(\boldsymbol{\xi}, \dot{\boldsymbol{\xi}}, \boldsymbol{\nu})$. The loss components $\mathcal{L}_{\text{SINDy}}$ and $\mathcal{L}_{\text{reg}}$ ensure accurate dynamics prediction and sparsity of the coefficients, respectively. The SINDy loss $\mathcal{L}_{\text{SINDy}}$ is calculated based on the predicted reduced-order model accelerations, $\ddot{\boldsymbol{\xi}}_{\text{pred}}$, and the corresponding decoded configuration-space accelerations, $\ddot{\mathbf{q}}_{\text{pred}}$.



(a) Simulated Go1

(b) Simulated A1

(c) Real Go1

Fig. 4. Sequential images of quadruped jumps: **(a)** A simulated Go1 robot performs a pronking jump. **(b)** A simulated A1 executes a froggy jump. **(c)** A real Go1 performs a pronking jump.

3.3. Data collection and preprocessing

Datasets were collected from both simulated and real-world environments. Simulated data were generated using the PyBullet engine (Coumans & Bai, 2016) with Unitree Go1 and A1 robots. Real-world data was gathered from a Unitree Go1 robot.

Simulation data for Go1 captured forward pronking motions (see Fig. 4(a)), achieved with model predictive control (MPC) (Ding, Atanassov, et al., 2024). Data from A1 for froggy jumping (see Fig. 4(b)) were collected using a Cartesian PD controller (Nguyen & Nguyen, 2022). The dataset includes joint angles and velocities ($\mathbf{q}_{\text{j}}, \dot{\mathbf{q}}_{\text{j}} \in \mathbb{R}^{12}$), body position and velocity ($\mathbf{q}_{\text{b}}, \dot{\mathbf{q}}_{\text{b}} \in \mathbb{R}^{6}$), input torques ($\boldsymbol{\tau} \in \mathbb{R}^{12}$), and the contact state ($\mathbf{c} \in \mathbb{R}^{4}$) for each timestep.

The simulated Go1 data consists of 156 jumps (120 for training, 30 for validation, and 6 for testing), with each jump having 800 points: 300 for launching, 300 for landing, and 200 for flight. The A1 dataset contains 150 jumps (100 for training, 40 for validation, and 10 for testing), with each jump comprising 1200 points: 400 for launch, 400 for landing, and 400 for flight. To enhance robustness, a variant of the datasets with added noise was considered, as described in the work from Panichi et al. (2025).

Real-world data was collected on a Go1 robot, as depicted in Fig. 4(c). The quadruped was operated using the same control algorithm as in the simulation, with data collection performed similarly using the official Go1 SDK. This dataset comprises 15 jumps, distributed as 12 for training, 2 for validation, and 1 for testing, maintaining the same data point distribution as the Go1 simulated data.

The collected data are preprocessed to compute necessary derivatives and forces. The state's acceleration $\ddot{\mathbf{q}}$ is calculated by numerically differentiating the velocity data $\dot{\mathbf{q}}$. The force on the CoM, $\mathcal{F} \in \mathbb{R}^{6}$, is calculated based on Eqs. (4) and (5), forming the final system input, $\mathbf{u}$.

3.4. Evaluation procedure

To validate the model Θ, the initial state ($\mathbf{q}_0, \dot{\mathbf{q}}_0$) and inputs $\mathbf{u}$ are transformed by the trained encoder and the input transformation equation to the latent initial state ($\boldsymbol{\xi}_0, \dot{\boldsymbol{\xi}}_0$) and $\boldsymbol{\nu}$, respectively. The model then predicts latent accelerations $\ddot{\boldsymbol{\xi}}_{\text{pred}}$, which are integrated using the *lsoda* method (Hindmarsh & Petzold, 2005) to obtain predicted latent space state $\boldsymbol{\xi}_{\text{pred}}$ and $\dot{\boldsymbol{\xi}}_{\text{pred}}$ with a simulation frequency of 500 Hz. This process is repeated for each time step. The predicted latent configuration $\boldsymbol{\xi}_{\text{pred}}$ is then decoded into a original configuration space by $\mathbf{q}_{\text{pred}} = \mathbf{W}_{\text{d}}\, \boldsymbol{\xi}_{\text{pred}} + \mathbf{B}_{\text{d}}$ for direct comparison with actual jump data.

An alternative evaluation approach involves simulating the learned system for shorter intervals (e.g., 0.1 s — 50 steps) with periodic resets to the original latent state ($\boldsymbol{\xi}, \dot{\boldsymbol{\xi}}$). This strategy decreases error accumulation over time, showcasing the precision of the model for medium-horizon forecasts, which could be used in planning and control frameworks such as MPC.

4. Results

This section evaluates the approach. We explored several latent-space dimensionalities to assess the trade-off between model compactness and accuracy. Two representative configurations are reported in Table 1.

Table 1
Hyperparameters of the models. The number of candidate functions is computed as $\sum_{k=0}^{p}(2l+k-1)+2l+l$ for latent dimension l and polynomial order p.

Parameter	Symbol/Value	Go1	A1
Latent dimension	l	2	4
Polynomial order	p	1	1
# Candidates	$\lvert\Theta\rvert$	15	29
Sparsity threshold	λ	0.01	
Learning rate	η	1×10^{-3}	
Batch size	B	100	
Epochs	N_{epoch}	501	

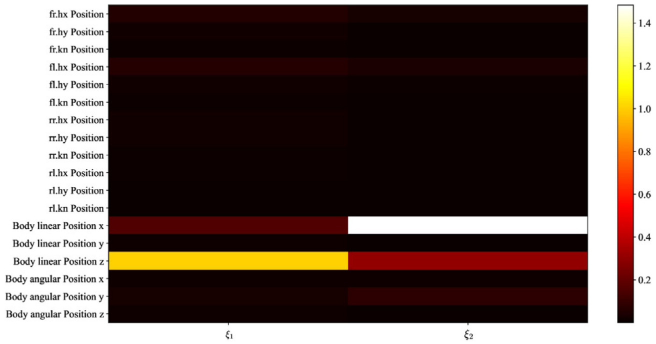


Fig. 5. Heatmap of the encoder weight matrix $\mathbf{W}_{\mathrm{e}}^{\mathrm{T}}$ (Go1 simulation, two latent dimensions). Brighter colors indicate larger weights; rows are 18 configuration dimensions and columns are the two latent dimensions.

4.1. Go1 simulation results

4.1.1. Learning Go1 pronking

We initially explored a two-dimensional latent model for the pronking. Fig. 5 visualizes the transposed weight matrix $\mathbf{W}_{\mathrm{e}}^{\mathrm{T}}$ of the trained encoder, indicating that the latent dimensions primarily correspond to the CoM positions in the x and z coordinates.

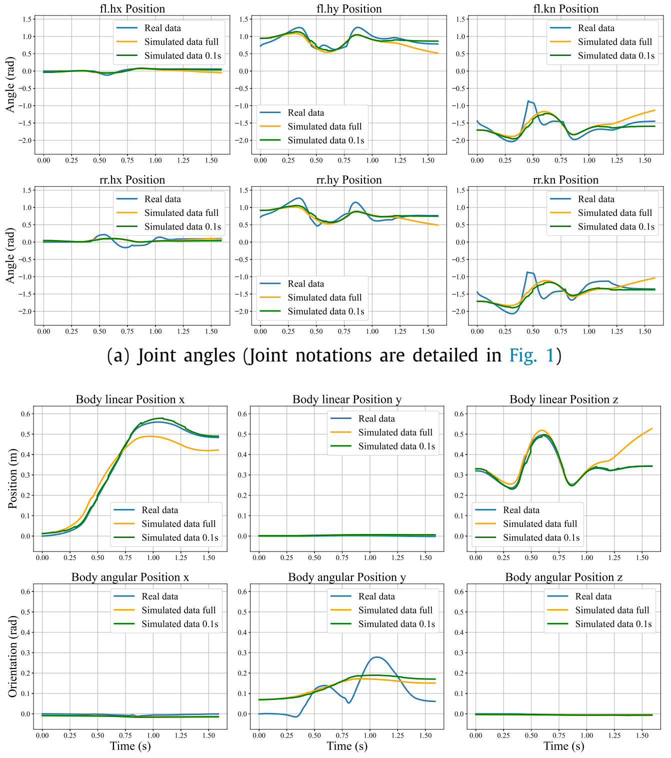
The learned equations of motion in the latent space for the contact and flight phases, respectively, are

$$\begin{cases}\ddot{\xi}_1{=}0.36-0.16\,\dot{\xi}_1-0.91\,\dot{\xi}_2-0.75\,\sin(\dot{\xi}_1)+0.11\,\nu_1-0.05\,\nu_2,\\ \ddot{\xi}_2{=}-0.16-0.23\,\dot{\xi}_1-0.75\,\dot{\xi}_2-0.96\,\sin(\dot{\xi}_2)+0.14\,\nu_2,\end{cases}\tag{12}$$

and

$$\begin{cases}\ddot{\xi}_1=0.29-1.22\,\dot{\xi}_1-1.00\sin(\dot{\xi}_1)+51.05\,\nu_1,\\ \ddot{\xi}_2=0.42\,\dot{\xi}_1+0.52\sin(\dot{\xi}_1).\end{cases}\tag{13}$$

The latent coordinates do not correspond to explicit physical variables, but the learned symbolic equations display structures that are consistent with key principles of quadruped jumping. The constant terms behave like gravity-related biases, the linear velocity terms act as effective damping, and the input-dependent components reflect how actuation influences the motion. The trigonometric functions of latent velocities serve as smooth nonlinear shaping terms that encode phase-dependent effects arising from posture changes, contact transitions, and limb coordination. These nonlinear components naturally emerge when compressing high-dimensional multibody dynamics into a low-dimensional latent representation.



(a) Joint angles (Joint notations are detailed in Fig. 1)

(b) CoM positions and orientations

Fig. 6. Evaluation of the trained model (with two latent dimensions) for simulated Go1 pronking. Considering the symmetry between right and left side, we only report the joints angle profiles in front left (fl) and rear right (rr) leg.

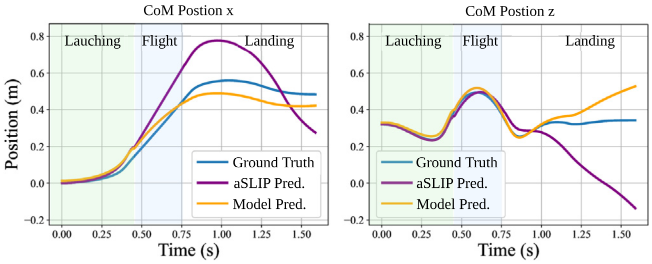


Fig. 7. Comparison between the learned model and the aSLIP model regarding the CoM positions prediction, where the Go 1 performs a pronking jumping motion.

The prediction results for Go1 pronking are shown in Fig. 6. Table 2 reports the mean errors and the standard deviation (std) along each direction, demonstrating the high prediction accuracy. To further demonstrate the effectiveness, we regard the actuated SLIP (aSLIP) model (see Appendix) as the baseline. Fig. 7 compares the body's linear position along the x and z axes, demonstrating that the learned model outperforms the baseline reduced-order model. Statistical results demonstrate that the average prediction error is reduced from 5.7e−2 m to 1.6e−2 m along the x direction and reduced from 9.5e−2 m to 4.0e−2 m along z direction.

4.1.2. Latent space dimensionality tradeoff analysis

Fig. 8 illustrates the effect of latent dimensionality on model design. Fig. 9 further evaluates the trade-off between accuracy (via final reconstruction error $\mathcal{E}_{\text{dec}}$) and complexity (via active coefficients $\lvert\Xi\rvert$) across 18 latent dimensions.

To quantitatively evaluate this trade-off, we employ a combined loss $\mathcal{L}_{\text{mod}}$ (Bozdogan, 1987), calculated as

$$\mathcal{L}_{\text{mod}}=2\,\mathcal{E}_{\text{dec}}+2\,\gamma\;\log(\lvert\Xi\rvert),\tag{14}$$

with $\gamma=0.001$.

Table 2
Prediction performance of the trained model on simulated and real datasets. Values denote mean absolute error ± standard deviation for joint angles and CoM pose components.

Dataset	Pre.	Joint angles						CoM position			CoM orientation		
		fl hx	fl hy	fl hz	rr hx	rr hy	rr hz	x	y	z	roll	pitch	yaw
SIM Go1	Full	1.3e−2 ± 1.5e−2	−8.0e−2 ± 0.11	7.5e−2 ± 0.18	2.3e−2 ± 7.6e−2	−5.4e−2 ± 0.13	3.7e−2 ± 0.22	−2.2e−2 ± 5.0e−2	4.4e−3 ± 1.3e−3	4.1e−2 ± 5.0e−2	−9.8e−3 ± 1.8e−3	3.0e−2 ± 6.4e−2	−1.5e−4 ± 8.1e−4
	0.1s	5.0e−3 ± 7.6e−3	−4.1e−3 ± 9.4e−2	−2.1e−2 ± 0.17	4.4e−3 ± 8.5e−2	−5.4e−3 ± 0.11	−2.5e−2 ± 0.21	8.9e−3 ± 4.6e−3	4.3e−3 ± 1.3e−3	5.4e−4 ± 9.4e−3	−9.3e−3 ± 2.6e−3	3.8e−2 ± 6.1e−2	−1.5e−4 ± 8.1e−4
SIM A1	Full	1.0e−10 ± 1.1e−10	0.19 ± 0.16	−0.30 ± 0.34	1.0e−10 ± 1.1e−10	0.26 ± 0.20	−0.32 ± 0.36	−3.5e−2 ± 5.1e−2	−8.0e−3 ± 1.5e−3	−6.1e−2 ± 5.8e−2	2.4e−3 ± 4.7e−3	0.10 ± 8.3e−2	−8.7e−4 ± 3.6e−3
	0.1s	1.0e−10 ± 2.6e−10	2.8e−3 ± 3.9e−2	−3.0e−2 ± 0.16	1.0e−10 ± 2.6e−10	1.4e−2 ± 5.2e−2	−3.9e−2 ± 0.23	−7.9e−3 ± 1.2e−3	−8.0e−3 ± 1.5e−3	−2.9e−3 ± 2.0e−3	2.4e−3 ± 4.7e−3	9.8e−3 ± 5.6e−2	−8.7e−4 ± 3.6e−3
Real Go1	Full	−3.7e−2 ± 2.6e−2	0.22 ± 0.15	−0.35 ± 0.39	−4.2e−2 ± 4.7e−2	0.34 ± 0.18	−0.65 ± 0.49	7.5e−2 ± 7.0e−2	−9.6e−4 ± 9.6e−3	−8.1e−2 ± 7.7e−2	3.3e−2 ± 2.2e−2	−0.14 ± 9.2e−2	0.1 ± 7.8e−2
	0.1s	−4.6e−3 ± 2.3e−2	1.7e−2 ± 8.3e−2	3.8e−2 ± 0.19	−7.5e−2 ± 5.1e−2	0.12 ± 5.6e−2	−0.21 ± 0.17	1.2e−3 ± 3.1e−3	−7.1e−3 ± 5.4e−3	1.2e−3 ± 3.4e−3	2.9e−2 ± 1.9e−2	−0.10 ± 5.9e−2	0.12 ± 8.9e−2

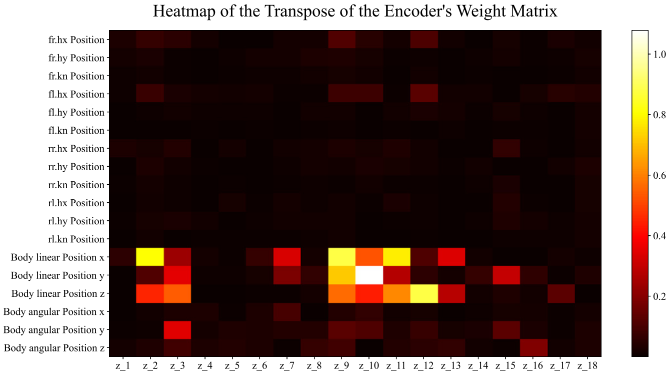


Fig. 8. Heatmap of the encoder weight matrix $\mathbf{W}_e^T$ (Go1 simulation, 18 latent dimensions).

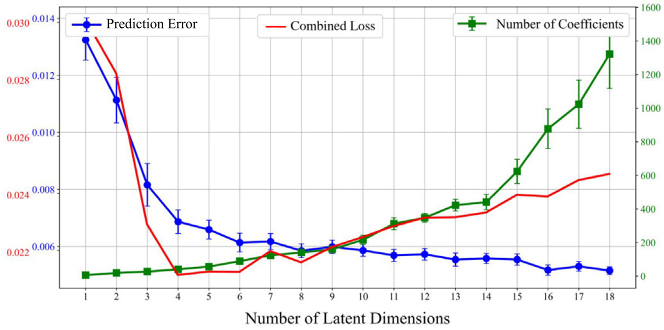


Fig. 9. Accuracy *vs.* complexity with the increase of latent space dimensions.

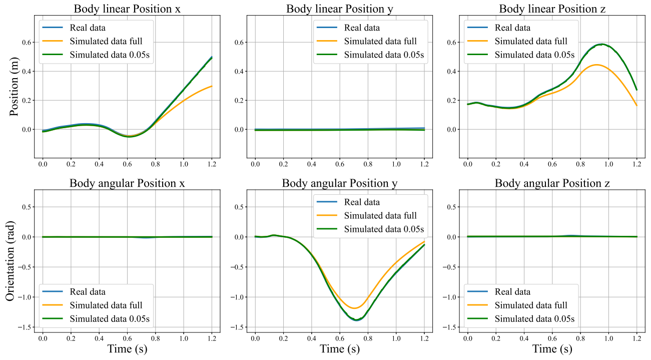


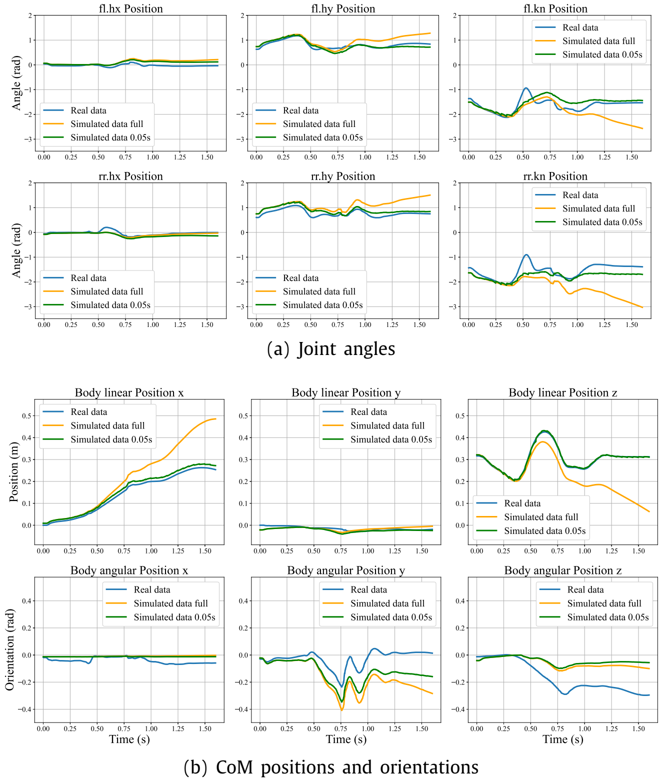
Fig. 10. Learning A1 froggy jumping (simulation data) with four latent dimensions.

The combined loss (red line in Fig. 9) suggests optimal performance with 4–6 latent dimensions. This aligns with the visibly significant components in Fig. 8, where further increasing dimensionality yields marginal gains at the cost of complexity.

4.2. A1 simulation results

We also conducted experiments with a simulated A1 froggy jump to evaluate the algorithm's effectiveness across different platforms and modalities. The results are illustrated in Fig. 10 and Table 2.

A notable difference is observed in the body angular position y (i.e., pitch angle) between the Unitree A1 and Go1. The froggy jump causes a more pronounced angular rotation along the y-axis in the A1's case. Despite the increased significance of this angular dimension for the froggy jump, the four-dimensional model successfully captures this dynamic and most joint positions. High precision is obtained when reset the latent state every 0.1 s.



(a) Joint angles

(b) CoM positions and orientations

Fig. 11. Learning Go1 pronking jumping (real data) with four latent dimensions.

4.3. Go1 real world results

Given the scarcity of real-world compared to simulated data, we first train the model on simulation data with added noise and then subsequently fine-tune it on the few-sample real-world dataset. Fig. 11 illustrates the model's ability to replicate a jump on the hardware, and Table 2 reports the prediction error.

As can be seen from Fig. 11 and Table 2, the proposed model performs well on actual hardware data, despite reduced accuracy during flight and initial landing phases compared to simulations. The divergence between real and predicted trajectories mainly results from unmodeled actuator and contact effects that accumulate during real execution. Applying domain randomization in future work could further alleviate this discrepancy by enhancing the model's robustness to such unmodeled variations. The symbolic model still captures the key motion patterns, with residual errors from friction, delay, and noise remaining bounded.

5. Discussion and conclusion

This work demonstrates that a low-dimensional, physics-aligned latent space can faithfully capture the essential dynamics of quadruped jumping. By combining a linear autoencoder embedding with sparse regression, the proposed framework yields compact and interpretable symbolic models that approximate full-order dynamics with meaningful accuracy. The learned equations recover gravity-like loading, effective damping, and actuation influence, while revealing coupling effects absent in idealized template models yet necessary to represent the richer behaviors of the full quadruped system. Beyond these empirical results, this study highlights how a linear latent embedding—together with a coherent multi-phase training strategy—enables SINDy to discover reduced-order template dynamics for hybrid legged motion, marking a conceptual departure from prior nonlinear SINDy-AE frameworks.

The approach remains sensitive to unmodeled actuator and contact effects, which accumulate during long-horizon rollouts and partly explain the observed sim-to-real discrepancies. Future work will extend the methodology to more diverse locomotion behaviors, incorporate hybrid contact transitions more explicitly, and scale the framework to higher-dimensional morphologies. Another promising direction is to integrate the learned symbolic models into adaptive or phase-dependent controllers for real-time robot movement tasks.

CRediT authorship contribution statement

Gioele Buriani: Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Conceptualization. **Jingyue Liu:** Writing – review & editing, Writing – original draft, Visualization, Supervision, Project administration, Conceptualization. **Maximilian Stölzle:** Writing – review & editing, Supervision. **Cosimo Della Santina:** Writing – review & editing, Supervision, Funding acquisition, Conceptualization. **Jiatao Ding:** Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Jingyue Liu, Maximilian Stölzle, and Cosimo Della Santina report financial support was provided by European Union's Horizon Europe Program Project EMERGE under Grant Agreement No. 101070918. The other authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix. Baseline: actuated slip model

The actuated SLIP (aSLIP) (Ding, Atanassov, et al., 2024) model extends the basic SLIP model by incorporating actuators, allows for enhanced control over legged jumping, as depicted in Fig. A.1. The CoM acceleration $\ddot{\mathbf{b}}$ is

$$\begin{aligned} &\text{In flight:} && \ddot{\mathbf{b}} = \boldsymbol{g} \\ &\text{In Contact:} && \ddot{\mathbf{b}} = \frac{k_s|\boldsymbol{l}_0-\boldsymbol{l}|\hat{\boldsymbol{l}}}{m} + \boldsymbol{g} + \boldsymbol{u} \end{aligned} \tag{A.1}$$

where $k_s \in \mathbb{R}^+$ is the spring constant, $m \in \mathbb{R}^+$ is the body mass, $\boldsymbol{l} = \mathbf{b} - \mathbf{p}_k$ is the leg vector, $\hat{\boldsymbol{l}}$ is the unit vector along the leg retraction direction, $\boldsymbol{l}_0$ is the rest length, and $\boldsymbol{g} = [0, 0, -g]^\top$ with g being the vertical gravitational constant. $\boldsymbol{u} \in \mathbb{R}^3$ are the driving forces generated by joint motors, which can be seen as the accumulated GRFs on four legs, as calculated using (3).

Data availability

Data will be made available on request.

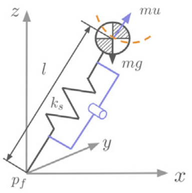


Fig. A.1. The aSLIP model. The actuation (denoted as $m\boldsymbol{u}$) is explicitly captured to enhance controllability.

References

Arm, P., Waibel, G., Preisig, J., Tuna, T., Zhou, R., Bickel, V., Ligeza, G., Miki, T., Kehl, F., Kolvenbach, H., et al. (2023). Scientific exploration of challenging planetary analog environments with a team of legged robots. *Science Robotics*, *8*(80), Article eade9548.

Atanassov, V., Ding, J., Kober, J., Havoutis, I., & Della Santina, C. (2024). Curriculum-based reinforcement learning for quadrupedal jumping: A reference-free design. *IEEE Robotics & Automation Magazine*.

Bai, L., Guan, J., Chen, X., Hou, J., & Duan, W. (2018). An optional passive/active transformable wheel-legged mobility concept for search and rescue robots. *Robotics and Autonomous Systems*, *107*, 145–155.

Bellicoso, C. D., Jenelten, F., Fankhauser, P., Gehring, C., Hwangbo, J., & Hutter, M. (2017). Dynamic locomotion and whole-body control for quadrupedal robots. In *2017 IEEE/RSJ international conference on intelligent robots and systems* (pp. 3359–3365). IEEE.

Blickhan, R. (1989). The spring-mass model for running and hopping. *Journal of Biomechanics*, *22*(11–12), 1217–1227.

Bozdogan, H. (1987). Model selection and Akaike's Information Criterion (aic): The general theory and its analytical extensions. *Psychometrika*, *52*(3), 345–370.

Brunton, S. L., Proctor, J. L., & Kutz, J. N. (2016). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, *113*(15), 3932–3937.

Calandra, R., Ivaldi, S., Deisenroth, M. P., Rueckert, E., & Peters, J. (2015). Learning inverse dynamics models with contacts. In *2015 IEEE international conference on robotics and automation* (pp. 3186–3191). IEEE.

Carpentier, J., & Mansard, N. (2018). Multicontact locomotion of legged robots. *IEEE Transactions on Robotics*, *34*(6), 1441–1460.

Champion, K., Lusch, B., Kutz, J. N., & Brunton, S. L. (2019). Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, *116*(45), 22445–22451.

Coumans, E., & Bai, Y. (2016). Pybullet, a python module for physics simulation for games, robotics and machine learning.

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

Ding, J., Atanassov, V., Panichi, E., Kober, J., & Della Santina, C. (2024). Robust quadrupedal jumping with impact-aware landing: Exploiting parallel elasticity. *IEEE Transactions on Robotics*, *40*, 3212–3231.

Ding, J., van Löben Sels, M. A., Angelini, F., Kober, J., & Della Santina, C. (2023). Robust jumping with an articulated soft quadruped via trajectory optimization and iterative learning. *IEEE Robotics and Automation Letters*, *9*(1), 255–262.

Ding, J., Posthoorn, P., Atanassov, V., Boekel, F., Kober, J., & Della Santina, C. (2024). Quadrupedal locomotion with parallel compliance: E-go design, modeling, and control. *IEEE/ASME Transactions on Mechatronics*, *29*(4), 2839–2848.

Ding, Jiatao, Yang, Peiyu, Boekel, Fabio, Kober, Jens, Pan, Wei, Saveriano, Matteo, & Della Santina, Cosimo (2025). Versatile, robust, and explosive locomotion with rigid and articulated compliant quadrupeds. *arXiv preprint arXiv:2504.12854*.

Gonzalez, F. J., & Balajewicz, M. (2018). Deep convolutional recurrent autoencoders for learning low-dimensional feature dynamics of fluid systems. arXiv preprint arXiv:1808.01346.

Hashemi, A., Orzechowski, G., Mikkola, A., & McPhee, J. (2023). Multibody dynamics and control using machine learning. *Multibody System Dynamics*, *58*(3), 397–431.

Hindmarsh, A. C., & Petzold, L. R. (2005). Lsodar, ordinary differential equation solver for stiff or non-stiff system with root-finding.

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*(5786), 504–507.

Hong, S., Kim, J. H., & Park, H. W. (2020). Real-time constrained nonlinear model predictive control on so (3) for dynamic legged locomotion. In *2020 IEEE/RSJ international conference on intelligent robots and systems* (pp. 3982–3989). IEEE.

Li, J., Gao, H., Wan, Y., Yu, H., & Zhou, C. (2023). A real-time planning and control framework for robust and dynamic quadrupedal locomotion. *Journal of Bionic Engineering*, *20*(4), 1449–1466.

Liu, J., Borja, P., & Della Santina, C. (2024). Physics-informed neural networks to model and control robots: A theoretical and experimental investigation. *Advanced Intelligent Systems*, *6*(5), Article 2300385.

Lutter, M., Ritter, C., & Peters, J. (2019). Deep lagrangian networks: Using physics as model prior for deep learning. arXiv preprint arXiv:1907.04490.

Mistry, M., Buchli, J., & Schaal, S. (2010). Inverse dynamics control of floating base systems using orthogonal decomposition. In *2010 IEEE international conference on robotics and automation* (pp. 3406–3412). IEEE.

Nguyen, C., & Nguyen, Q. (2022). Contact-timing and trajectory optimization for 3D jumping on quadruped robots. In *2022 IEEE/RSJ international conference on intelligent robots and systems* (pp. 11994–11999). IEEE.

Nguyen, Q., Powell, M. J., Katz, B., Carlo, J. Di., & Kim, S. (2019). Optimized jumping on the MIT cheetah 3 robot. In *2019 international conference on robotics and automation* (pp. 7448–7454). IEEE.

Ogunmolu, O., Gu, X., Jiang, S., & Gans, N. (2016). Nonlinear systems identification using deep dynamic neural networks. arXiv preprint arXiv:1610.01439.

Pandey, P., Khodaparast, H. H., Friswell, M., Chatterjee, T., Jamia, N., & Deighan, T. (2024). Data-driven system identification of unknown systems utilising sparse identification of nonlinear dynamics (Sindy). *Journal of Physics: Conference Series*, *2647*, Article 062006.

Panichi, E., Ding, J., Atanassov, V., Yang, P., Kober, J., Pan, W., & Della Santina, C. (2025). On-the-fly jumping with soft landing: Leveraging trajectory optimization and behavior cloning. *IEEE/ASME Transactions on Mechatronics*.

Panichi, Edoardo, Ding, Jiatao, Atanassov, Vassil, Yang, Peiyu, Kober, Jens, Pan, Wei, & Santina, Cosimo Della (2025). On-the-fly jumping with soft landing: leveraging trajectory optimization and behavior cloning. *IEEE/ASME Transactions on Mechatronics*, *30*(4), 3142–3151. http://dx.doi.org/10.1109/TMECH.2025.3572176.

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, *378*, 686–707.

Rajendra, P., & Brahmajirao, V. (2020). Modeling of dynamical systems through deep learning. *Biophysical Reviews*, *12*(6), 1311–1320.

Refinetti, M., & Goldt, S. (2022). The dynamics of representation learning in shallow, non-linear autoencoders. In *International conference on machine learning* (pp. 18499–18519). PMLR.

Righetti, L., Buchli, J., Mistry, M., & Schaal, S. (2011). Inverse dynamics control of floating-base robots with external constraints: A unified view. In *2011 IEEE international conference on robotics and automation* (pp. 1085–1090). IEEE.

Stölzle, Maximilian (2025). Safe yet precise soft robots: incorporating physics into learned models for control. ISBN: 978-94-6384-836-7, http://dx.doi.org/10.4233/uuid:24c1f667-8fd6-431a-bb78-11d22f8cb3da.

Stölzle, M., & Santina, C. D. (2024). Input-to-state stable coupled oscillator networks for closed-form model-based control in latent space. In *Proceedings of the 38th international conference on neural information processing systems* (pp. 82010–82059).

Torrey, L., & Shavlik, J. (2010). Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques* (pp. 242–264). IGI Global Scientific Publishing.

Yan, W., Pan, Y., Che, J., Yu, J., & Han, Z. (2021). Whole-body kinematic and dynamic modeling for quadruped robot under different gaits and mechanism topologies. *PeerJ Computer Science*, *7*, Article e821.