

Floating-Base Deep Lagrangian Networks

Lucas Schulze¹, Juliano Decico Negri², Victor Barasuol³, Vivian Suzano Medeiros²,
Marcelo Becker², Jan Peters^{1,4,5,6}, Oleg Arenz¹

Abstract—Grey-box methods for system identification combine deep learning with physics-informed constraints, capturing complex dependencies while improving out-of-distribution generalization. Yet, despite the growing importance of floating-base systems such as humanoids and quadrupeds, current grey-box models ignore their specific physical constraints. For instance, the inertia matrix is not only positive definite but also exhibits branch-induced sparsity and input independence. Moreover, the 6×6 composite spatial inertia of the floating base inherits properties of single-rigid-body inertia matrices. As we show, this includes the triangle inequality on the eigenvalues of the composite rotational inertia. To address the lack of physical consistency in deep learning models of floating-base systems, we introduce a parameterization of inertia matrices that satisfies all these constraints. Inspired by Deep Lagrangian Networks (DeLaN), we train neural networks to predict physically plausible inertia matrices that minimize inverse dynamics error under Lagrangian mechanics. For evaluation, we collected and released a dataset on multiple quadrupeds and humanoids. In these experiments, our Floating-Base Deep Lagrangian Networks (FeLaN) achieve highly competitive performance on both simulated and real robots, while providing greater physical interpretability.

I. INTRODUCTION

Accurate dynamic models are essential for simulation, control, and state estimation in robotics. Classical system identification (SysID) constructs a white-box model based on physical principles and estimates its parameters from data. This approach requires prior knowledge of the system, but generalizes well as long as the model assumptions hold. In contrast, black-box methods approximate the dynamics directly from data without any prior knowledge. Although flexible, they often generalize poorly outside the training domain and lack interpretability. Grey-box methods combine data-driven models with physical structure. For example, Hamiltonian Neural Networks (HNN) [1] and Deep Lagrangian Networks (DeLaN) [2] embed Hamiltonian and Lagrangian mechanics into deep learning architectures, respectively. By enforcing physical properties such as conservation laws, these formulations improve interpretability, generalizability, and sample efficiency, and have demonstrated success on fixed-base [3] and single rigid-body [4], [5] robots. In addition, learning function approximations allows to learn a physically consistent model when the true system

states are partially or entirely unobserved [6], or combine the true states with additional representations for context adaptation [7].

However, naively applying these methods to floating-base systems, such as quadrupeds and humanoids, neglects several physical constraints which may limit the generalizability of the learned model. First of all, such robots contain multiple kinematic chains, leading to an inertia matrix with a sparse block structure where certain blocks are independent of specific joint inputs. Furthermore, the composite spatial inertia matrix [8], i.e., the part of the inertia matrix that maps floating-base accelerations to the total wrench applied at the base, should respect a triangular inequality [9]. Even for DeLaN, which directly predicts the Cholesky decomposition of the inertia matrix, imposing these additional constraints is challenging: sparsity and input-independencies are not preserved in the Cholesky matrix [10] and enforcing the triangular inequality is non-trivial. Hence, grey-box models for floating-base systems remain limited to simplified dynamics [11], [12].

In this work, we introduce a novel parameterization of the inertia matrix of floating-base system based on the reordered Cholesky factorization [10]. Our parameterization ensures positive-definiteness; branch-induced sparsity and input-independencies; the triangular inequality of the composite spatial inertia matrix; and the stationarity of the system total mass. Based on DeLaN, we propose the Floating-Base Deep Lagrangian Networks (FeLaN), a novel grey-box method for the identification of floating-base system dynamics. We validate our methods in several simulation environments and on multiple real quadrupeds and humanoids, namely on Unitree Go2 [13], Boston Dynamics Spot [14], Pal Robotics TALOS [15], and HyQReal2, the updated version of HyQReal1 [16].

To summarize, our main contributions are as follows:

- We extend the full physical consistency condition of single rigid body to the composite spatial inertia.
- We present a novel parametrization of the inertia matrix for floating-base systems that preserves branch-induced sparsity and ensures full physical consistency of the composite spatial inertia.
- Inspired by DeLaN, we introduce a deep learning architecture for physically consistent system identification of floating-base robots, combining Lagrangian mechanics with our proposed parametrization.
- We perform an extensive comparison of our approach against multiple baselines on both simulated and real robot data.

¹Department of Computer Science, Technical University of Darmstadt, Germany. ²Mobile Robotics Group, São Carlos School of Engineering, University of São Paulo (EESC-USP), Brazil. ³Dynamic Legged Systems Lab, Istituto Italiano di Tecnologia (IIT). ⁴Hessian.AI. ⁵German Research Center for AI (DFKI), Research Department: Systems AI for Robot Learning. ⁶Robotics Institute Germany (RIG).

Corresponding author: lucas.schulze@tu-darmstadt.de

The remainder of this paper is organized as follows. Section II reviews related work on system identification of floating-base systems. Section III briefly reviews Lagrangian mechanics and DeLaN. We then present our proposed parametrization of the whole-body inertia matrix in Sec. IV and introduce the deep learning architecture built on this parametrization in Sec. V. Section VI describes the compared baselines and the evaluated experiments. Finally, Section VII summarizes our findings and discusses future work.

II. RELATED WORK

Classical SysID techniques for rigid-body systems usually estimate the kinematic parameters (e.g., link lengths, joint axes) separately from the dynamic parameters (e.g., mass, center of mass, inertia), either by prior estimation or by assuming they are given [17]. This separation is motivated by the fact that the equations of motion are linear in the inertial parameters, which enables estimation via linear least squares assuming persistent excitation [18]. An alternative approach is to employ differentiable simulators to directly identify all system parameters through stochastic gradient descent [19], [20]. For floating-base systems, different sensor modalities were explored, such as joint torques [21], joint torques and force sensors [22], or force plates [23].

Conversely, black-box methods employ nonparametric models to learn either the full system dynamics or the residual with respect to a nominal model. For floating-base systems, used methods include Gaussian process regression [24], [25]; and neural networks [26], [27]. However, these methods usually model only the forward or inverse dynamics, not holding physical properties.

The integration of physics into deep learning demonstrated a useful inductive bias, especially in small data regimes [28]. For floating-base robots, [11] used DeLaN to estimate simplified formulations of a quadruped’s inertia matrix for model-based locomotion. However, only the positive definiteness of the inertia matrix is enforced, which is a necessary but not sufficient condition for full physical consistency. The composite spatial inertia of a floating-base system must also satisfy the triangle inequality for the principal moments of the rotational inertia, as in the single rigid body case presented in [9].

As the generalized coordinates of a floating-base system include the base pose in $SE(3)$, one could decide to represent the robot’s configuration by the pose of each body in $SE(3)$ instead of joint coordinates. For example, [29], [30], [31] follow this approach, where the inertia matrix simplifies to a block-diagonal concatenation of each body’s spatial inertia. While this formulation can ensure full physical consistency, it introduces redundant coordinates and requires knowledge of the system kinematics, as the body poses are unobserved states. In addition, the reported applications were restricted to systems with a few number of degrees of freedom, e.g., planar pendulums, quadrotors.

Noether’s theorem connects symmetries and conservation laws, motivating their use as an inductive bias [28]. For instance, [32] and [33] add symmetry constraints for ground

reaction force estimation, demonstrating significant improvements over black-box baselines. However, these methods depend on prior knowledge of the robot’s symmetry groups, which is not always available or generalizable to any robot.

Floating-base systems may have multiple kinematic branches, which are only indirectly coupled through the base, resulting in a partial functional decoupling. In an attempt to exploit this property, [34] learns an inverse dynamics function for each robot leg. However, they included contact state variables and a projection of the base position in the leg frame, which requires information regarding the robot kinematics. Furthermore, the partial functional decoupling is truly described by the branch-induced sparsity of the inertia matrix [8], which was first exploited by [10] to efficiently factorize a given inertia matrix.

Unlike structural constraints, Sparse Identification of Non-linear Dynamics (SINDy) [35] learns parsimonious models through optimization, leveraging the inherent sparsity of physical laws. For example, [12] combines SINDy with priors on quadruped jumping dynamics to estimate a low-dimensional model equivalent to a single rigid body. While this enforces sparsity, it fails to capture branch-induced sparsity and to guarantee the properties of the composite spatial inertia.

III. PRELIMINARIES

A. Notation and Definitions

Scalar and scalar-value functions are denoted by lowercase letters; vector and vector-valued are denoted with bold lowercase letters; and matrices with uppercase letters. The $n \times n$ identity matrix is denoted as $\mathbf{1}_n$. $\mathbf{S}(\mathbf{a})$ and $\mathbf{S}_a$ denote the skew-symmetric operator over the vector $\mathbf{a}$. $\mathbf{A} \succeq 0$ and $\mathbf{A} \succ 0$ represents that the symmetric matrix $\mathbf{A}$ is positive semidefinite and definite, respectively. The superscript $\mathbf{W}$ denotes a variable in the world frame, whereas all other variables are expressed in the base frame. The dependence of matrices and vectors on configuration variables is written explicitly at first mention. For brevity, these dependencies are omitted in subsequent expressions but remain implicitly valid.

B. Lagrangian Mechanics

Lagrangian mechanics describes the dynamics of mechanical systems by their energy. The Lagrangian $\mathcal{L}$ is defined as the difference between the kinetic K and potential energy P . For rigid body systems, the Lagrangian takes the form

$$\mathcal{L}(\boldsymbol{\nu}, \dot{\boldsymbol{\nu}}) = K(\boldsymbol{\nu}, \dot{\boldsymbol{\nu}}) - P(\boldsymbol{\nu}) = \frac{1}{2} \dot{\boldsymbol{\nu}}^T \mathbf{H}(\boldsymbol{\nu}) \dot{\boldsymbol{\nu}} - P(\boldsymbol{\nu}), \quad (1)$$

where $\boldsymbol{\nu}$ and $\dot{\boldsymbol{\nu}}$ are the generalized positions and velocities, respectively, and $\mathbf{H}(\boldsymbol{\nu})$ is the positive definite inertia matrix.

The equations of motion can be derived from the Euler-Lagrange equation with non-conservative forces $\boldsymbol{\tau}_\nu$

$$\frac{\partial^2 \mathcal{L}(\boldsymbol{\nu}, \dot{\boldsymbol{\nu}})}{\partial^2 \dot{\boldsymbol{\nu}}} \ddot{\boldsymbol{\nu}} + \frac{\partial \mathcal{L}(\boldsymbol{\nu}, \dot{\boldsymbol{\nu}})}{\partial \boldsymbol{\nu} \partial \dot{\boldsymbol{\nu}}} \dot{\boldsymbol{\nu}} - \frac{\partial \mathcal{L}(\boldsymbol{\nu}, \dot{\boldsymbol{\nu}})}{\partial \boldsymbol{\nu}} = \boldsymbol{\tau}_\nu, \quad (2)$$

where τ_ν includes all external forces acting on the system, e.g., actuated joint torques, contact forces. Equation (2) can be written in the following compact form

$$\mathbf{H}(\nu)\ddot{\nu} + \mathbf{c}(\nu, \dot{\nu}) + \mathbf{g}(\nu) = \tau_\nu, \quad (3)$$

where $\mathbf{c}(\nu, \dot{\nu})$ and $\mathbf{g}(\nu)$ are the generalized Coriolis and gravitational forces, respectively.

C. Deep Lagrangian Networks

Based on (1), DeLaN [36] approximates the joint space inertia matrix $\mathbf{H}$ and P using two networks to estimate the equations of motion (3) of a fixed-base robot. To guarantee physical consistency, i.e., $\mathbf{H} \succ 0$, DeLaN parameterizes the inertia matrix in terms of its Cholesky decomposition, i.e.

$$\mathbf{H}(\mathbf{q}) = \mathbf{C}(\mathbf{q})\mathbf{C}(\mathbf{q})^\top, \quad (4)$$

where $\mathbf{C}(\mathbf{q})$ is a lower triangular matrix with positive diagonal elements.

D. Fully Physically-Consistent Spatial Inertia

The inertia of a rigid body is defined by its mass $m_b \in \mathbb{R}$, the position of the center of mass in the body frame $\mathbf{r}_c \in \mathbb{R}^3$, the first mass moment $\mathbf{h}_b = m_b \mathbf{r}_c$, and the rotational inertia $\mathbf{I}_b \in \mathbb{R}^{3 \times 3}$ defined at the body frame. These parameters define the body spatial inertia [8]

$$\mathbf{M}_b = \begin{bmatrix} m_b \mathbf{1}_3 & \mathbf{S}(\mathbf{h}_b)^\top \\ \mathbf{S}(\mathbf{h}_b) & \mathbf{I}_b \end{bmatrix}. \quad (5)$$

As $\mathbf{I}_b$ results from the principal moment of inertia and the positivity of the mass density function, [9] demonstrated that the eigenvalues of $\mathbf{I}_b$, i.e., I_x, I_y, I_z , are not only positive, but also satisfies the triangle inequality

$$I_x + I_y \geq I_z, I_y + I_z \geq I_x, I_x + I_z \geq I_y, \quad (6)$$

which is equivalent to

$$\frac{1}{2} \text{Tr}(\mathbf{I}_b) \geq \lambda_{\max}(\mathbf{I}_b) \quad (7)$$

where $\lambda_{\max}(\mathbf{I}_b)$ is the largest eigenvalue of $\mathbf{I}_b$. Hence, full physical consistency is only defined for $m_b > 0$ and (6) fulfilled. From (7), [37] observed that the triangle inequality of a body's rotational inertia $\mathbf{I}_b$ is equivalent to requiring a positive semidefinite covariance Σ_b of the body's mass distribution, which are related through

$$\mathbf{I}_b = \text{Tr}(\Sigma_b) \mathbf{1}_3 - \Sigma_b. \quad (8)$$

IV. PHYSICALLY CONSISTENT INERTIA MATRIX PARAMETRIZATION

In this section, we first investigate the properties of the $\mathbf{H}$ of a floating-base system, and extend the concept of fully physical consistency to the composite spatial inertia. We then propose an unconstrained parametrization of $\mathbf{H}$ that address all the investigated physical constraints.

A. Inertia Matrix of a Floating-Base system

For floating-base systems, the generalized coordinates account for both the base pose $\mathbf{q}_B \in \text{SE}(3)$, and the joint position $\mathbf{q} \in \mathbb{R}^{n_q}$, leading to $\nu = [\mathbf{q}_B; \mathbf{q}]$. The top-left 6×6 block matrix of $\mathbf{H}$ corresponds to the composite spatial inertia $\mathbf{H}_B(\mathbf{q})$ of the whole floating-base system [8]. Equivalent to (5), the total mass $m \in \mathbb{R}$, the total first mass moment $\mathbf{h}(\mathbf{q}) \in \mathbb{R}^3$, and the composite rotational inertia $\mathbf{I}_B(\mathbf{q}) \in \mathbb{R}^{3 \times 3}$, define $\mathbf{H}_B(\mathbf{q})$ in the base frame as

$$\mathbf{H}_B(\mathbf{q}) = \begin{bmatrix} m \mathbf{1}_3 & \mathbf{S}(\mathbf{h}(\mathbf{q}))^\top \\ \mathbf{S}(\mathbf{h}(\mathbf{q})) & \mathbf{I}_B(\mathbf{q}) \end{bmatrix}. \quad (9)$$

Unlike the spatial inertia of a single rigid body, both $\mathbf{h}(\mathbf{q})$ and $\mathbf{I}_B(\mathbf{q})$ depend on the joint configuration $\mathbf{q}$ due to the transformation from body to base frame, whereas m remains constant and results in an isotropic linear inertia. Since $\mathbf{H}_B(\mathbf{q})$ has the same structure as the spatial inertia of a single rigid body (5), $\mathbf{H}_B(\mathbf{q}) \succ 0$ is a necessary but not sufficient condition for full physical consistency, as established by Lemma 1.

Lemma 1: A composite inertia matrix is fully physical consistent only if $m > 0$ and $\mathbf{I}_B(\mathbf{q})$ satisfies the triangle inequality.

Proof: Through the parallel axis theorem, the rotational inertia of the i th body with respect to the base frame is

$$\mathbf{I}_i(\mathbf{q}) = \mathbf{R}_i(\mathbf{q}) \mathbf{I}_{bi} \mathbf{R}_i(\mathbf{q}) + m_i \mathbf{S}(\mathbf{r}_i(\mathbf{q})) \mathbf{S}(\mathbf{r}_i(\mathbf{q})), \quad (10)$$

where $\mathbf{R}_i(\mathbf{q}) \in \text{SO}(3)$ is the rotation from body and base frame, $\mathbf{r}_i$ is the body position w.r.t. the base. The parallel axis theorem preserves the triangular inequality, as it corresponds to a change of reference frame [9]. From (7), the trace of the composite rotational inertia satisfies a minimum bound:

$$\text{Tr}(\mathbf{I}_B(\mathbf{q})) = \sum_{i=1}^{n_b} \text{Tr}(\mathbf{I}_i(\mathbf{q})) \geq \sum_{i=1}^{n_b} 2\lambda_{\max}(\mathbf{I}_i(\mathbf{q})). \quad (11)$$

Yet, $\mathbf{I}_i \succ 0$ implies in $\sum_{i=1}^{n_b} \lambda_{\max}(\mathbf{I}_i(\mathbf{q})) \geq \lambda_{\max}(\mathbf{I}_B(\mathbf{q}))$, that combined with (11), yields

$$\frac{1}{2} \text{Tr}(\mathbf{I}_B(\mathbf{q})) \geq \sum_{i=1}^{n_b} \lambda_{\max}(\mathbf{I}_{bi}(\mathbf{q})) \geq \lambda_{\max}(\mathbf{I}_B(\mathbf{q})), \quad (12)$$

proving the triangle inequality. ■

Note that by expressing $\mathbf{H}_B(\mathbf{q})$ in the base frame, the inertia matrix $\mathbf{H}(\mathbf{q})$ depends only on the joint positions $\mathbf{q}$, but not on the entire generalized coordinates ν , resulting in translational and rotational invariances regarding the base.

Due to the branched-induced sparsity, the joint inertia $\mathbf{H}_k$ of the k th kinematic branch is a function only of the joints of the k th branch, along with the linear and rotational inertial coupling with the robot's base.

B. Branch-Induced Sparsity

For fixed-base robots with a single kinematic branch, a positive definite inertia matrix can be estimated from the Cholesky factor (4) [2]. However, as established by

Lemma 1, $\mathbf{H} \succ 0$ is a necessary but not a sufficient condition for a physically consistent $\mathbf{H}$ of a floating-base system.

The main limitation is that the standard Cholesky factorization (4) always results in a dense factor $\mathbf{C}(\mathbf{q})$ [10], not holding the branch-induced sparsity and not necessarily the structure of $\mathbf{H}_B(\mathbf{q})$.

To exploit the branch-induced sparsity and efficiently factorize a given $\mathbf{H}$, [10] proposes to use a reordered Cholesky factorization

$$\mathbf{H}(\mathbf{q}) = \mathbf{L}(\mathbf{q})^T \mathbf{L}(\mathbf{q}), \quad (13)$$

where $\mathbf{L}(\mathbf{q})$ is a lower triangular matrix as $\mathbf{C}(\mathbf{q})$.

The reordered factorization in (13) is equivalent to a standard Cholesky factorization on a permutation of the original matrix. It preserves the same sparsity pattern from $\mathbf{H}$ in $\mathbf{L}$, while having the same numerical properties, e.g., positive definiteness. The only additional information required is the bodies' connectivity graph, i.e., the kinematic chains.

For a robot with n_k kinematic branches, note that $\mathbf{L}$ has the following structure:

$$\mathbf{L} = \begin{bmatrix} \mathbf{L}_L & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{L}_{RL} & \mathbf{L}_R & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{L}_{1L} & \mathbf{L}_{1R} & \mathbf{L}_1 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{L}_{2L} & \mathbf{L}_{2R} & \mathbf{0} & \mathbf{L}_2 & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{L}_{n_k L} & \mathbf{L}_{n_k R} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{L}_{n_k} \end{bmatrix} \quad (14)$$

where $\mathbf{L}_L(\mathbf{q})$, $\mathbf{L}_{RL}(\mathbf{q})$, $\mathbf{L}_R(\mathbf{q})$ are a function of all joints $\mathbf{q}$; $\mathbf{L}_{kL}(\mathbf{q}_k)$, $\mathbf{L}_{kR}(\mathbf{q}_k)$, $\mathbf{L}_k(\mathbf{q}_k)$ depend only of $\mathbf{q}_k$ of the k th kinematic branch. For brevity, we omit explicitly writing the dependence of matrices and vectors on the joint positions $\mathbf{q}$ in subsequent expressions; this dependence remains valid.

Therefore, any matrix $\mathbf{L}$ with the sparsity and functional characteristics in (14) implies the same properties in $\mathbf{H}$ using (13). Unless otherwise stated, $\mathbf{L}$ will hereafter denote the one defined in (13).

Based on the structure in (14), we parametrize its matrix blocks to fulfill all physical constraints of $\mathbf{H}_B(\mathbf{q})$. The following subsections detail how these constraints are ensured.

C. Positive Definiteness

As the reordered Cholesky (13) is equivalent to a factorization on a permuted matrix, $\mathbf{L}$ with positive diagonal implies $\mathbf{H} \succ 0$. More precisely, this can be achieved by ensuring all the diagonal blocks (e.g., $\mathbf{L}_L$, $\mathbf{L}_R$, $\mathbf{L}_1$, $\dots$, $\mathbf{L}_{n_k}$) to have positive diagonal elements.

D. Inertia Triangle Inequality

The relation (8) between a semidefinite covariance matrix and the rotational inertia is defined in any arbitrary frame [37]. Therefore, we expand its concept to the composite spatial inertia using Lemma 1, relating a configuration dependent covariance matrix $\Sigma_R(\mathbf{q}) \succ 0$ to a fully physically consistent $\mathbf{I}_B(\mathbf{q})$ by

$$\mathbf{I}_B = \text{Tr}(\Sigma_R) \mathbf{1}_3 - \Sigma_R, \quad (15)$$

where $\Sigma_R(\mathbf{q}) = \mathbf{L}_\Sigma(\mathbf{q})^T \mathbf{L}_\Sigma(\mathbf{q})$ with $\mathbf{L}_\Sigma(\mathbf{q}) \succ 0$.

To apply this parameterization to $\mathbf{I}_B(\mathbf{q})$, we expand (13) with (14) to obtain

$$\mathbf{L}_R^T \mathbf{L}_R + \mathbf{W}^T \mathbf{W} = \mathbf{I}_B, \quad (16)$$

where $\mathbf{W}(\mathbf{q}) = [\mathbf{L}_{1R}(\mathbf{q}_1), \dots, \mathbf{L}_{n_k R}(\mathbf{q}_{n_k})]$. Hence, we can express $\mathbf{L}_R$ in terms of Σ_R and $\mathbf{W}(\mathbf{q})$ as

$$\mathbf{L}_R^T \mathbf{L}_R = \mathbf{I}_B - \mathbf{W}^T \mathbf{W} \quad (17)$$

$$= \text{Tr}(\Sigma_R) \mathbf{1}_3 - \Sigma_R - \mathbf{W}^T \mathbf{W} = \mathbf{D}, \quad (18)$$

$$\mathbf{L}_R = \text{reorderedChol}(\mathbf{D}) \text{ with } \mathbf{D} \succ 0. \quad (19)$$

The factorization algorithm `reorderedChol` is proposed by [10] that factorizes an inertia matrix using the kinematic chain information to exploit the branch-induced sparsity.

E. First mass moment

Considering the linear and angular coupling due to the first mass moment given by $\mathbf{S}(\mathbf{h}(\mathbf{q}))$ in (9), expanding (13) defines the following constraint

$$\mathbf{L}_{RL}^T \mathbf{L}_R + \mathbf{K}^T \mathbf{W} = \mathbf{S}_h^T, \quad (20)$$

where $\mathbf{K}(\mathbf{q}) = [\mathbf{L}_{1L}(\mathbf{q}_1), \dots, \mathbf{L}_{n_k L}(\mathbf{q}_{n_k})]$. Therefore, the constraint (20) can be fulfilled by expressing $\mathbf{L}_{RL}(\mathbf{q})$ in terms of $\mathbf{L}_R$, $\mathbf{K}$, $\mathbf{W}$, and the skew-symmetric matrix $\mathbf{S}_h$,

$$\mathbf{L}_{RL} = ((\mathbf{S}_h^T - \mathbf{K}^T \mathbf{W}) \mathbf{L}_R^{-1})^T. \quad (21)$$

F. Mass

Similarly, the following constraint is defined for the linear inertia of the composite spatial inertia

$$\mathbf{L}_L^T \mathbf{L}_L + \mathbf{U}^T \mathbf{U} = m \mathbf{1}_3, \quad (22)$$

where $\mathbf{U} = [\mathbf{L}_{RL}, \mathbf{K}]$. Thus, we express $\mathbf{L}_L$ in terms of the total mass $m > 0$ and $\mathbf{U}$ as

$$\mathbf{L}_L^T \mathbf{L}_L = m \mathbf{1}_3 - \mathbf{U}^T \mathbf{U} = \mathbf{T}, \quad (23)$$

$$\mathbf{L}_L = \text{reorderedChol}(\mathbf{T}) \text{ with } \mathbf{T} \succ 0. \quad (24)$$

G. Branched Kinematic Chains

Branched kinematic chains, i.e., chains with sub-chains branching from one of their bodies, introduce additional branch-induced sparsity. This type of branching is commonly observed in humanoids, where the arms and head branch from the upper torso, and in quadrupeds with spine joints. Specifically, the factor $\mathbf{L}_{kj}$ associated to the sub-chain j of a chain k depends only on $\mathbf{q}_{kj}$. Consequently, the branch-induced sparsity can be fully exploited through appropriate computation of the elements of $\mathbf{L}$, while the remainder of the parametrization and its associated physical consistency properties are preserved.

H. Parametrization Overview

Given m , $\mathbf{h}$, $\mathbf{L}_\Sigma$ and $\mathbf{L}_{kL}$, $\mathbf{L}_{kR}$, $\mathbf{L}_k$ for each k kinematic branch, we compute an $\mathbf{L}(\mathbf{q})$ which leads to a physically consistent $\mathbf{H}$ through (20), (22), and (24). By directly parametrizing the composite spatial inertia and the joint-space inertia, we require fewer parameters than estimating all 16 kinodynamic parameters for each robot body. Table I shows the number of parameters estimated for each evaluated robot.

TABLE I: Number of parameters per robot.

	Go2 / Spot / HyQReal2 ($n_q = 12$)	Spot with Arm ($n_q = 17$)	Talos ($n_q = 22$)
$\mathbf{H}$	324	529	784
Standard $\mathbf{C}$	171	276	406
Reordered $\mathbf{L}$	117	162	210
Proposed	106	151	199
Body Parameters	208	288	368

V. FLOATING-BASE DELAN

In this section, we present how we combine deep learning with the parametrization proposed in Sec. IV.

A. Inertia Matrix

Similar to DeLaN [36], we estimate the elements of $\mathbf{L}(\mathbf{q})$ with deep neural networks. Positive definiteness is satisfied by enforcing positive diagonal elements, achieved by applying an activation function, e.g., ReLU, softplus, combined with an offset ϵ_L .

To enforce input-independencies for each kinematic branch, we define n_k networks $\mathbf{NN}_k$ to estimate $\mathbf{L}_{kL}(\mathbf{q}_k)$, $\mathbf{L}_{kR}(\mathbf{q}_k)$ and $\mathbf{L}_k(\mathbf{q}_k)$ using only the joints $\mathbf{q}_k$ of the k th branch as input. Meanwhile, $\mathbf{L}_R(\mathbf{q})$ and $\mathbf{L}_\Sigma(\mathbf{q})$ are estimated by a single network $\mathbf{NN}_R$ that takes all joint positions as input, while m is computed by θ_m^2 which is an unconstrained trainable parameter.

Note that (17) requires $\mathbf{D} \succ 0$, which is not necessarily true only from the network's outputs. Therefore, to ensure a proper factorization, we define $\hat{\mathbf{D}}$ using a shifted term β as

$$\beta = \epsilon_D + \text{softplus}(-\mu_D), \quad (25)$$

$$\hat{\mathbf{D}} = \mathbf{D} + \beta \mathbf{1}_3, \quad (26)$$

where μ_D is the smallest eigenvalue of $\mathbf{D}$, and ϵ_D is a small offset. Analogously, we ensure $\mathbf{T} \succ 0$ in (24) using

$$\hat{m} = \text{softplus}(m - \lambda_U) + \epsilon_m + \lambda_U, \quad (27)$$

$$\mathbf{T} = \hat{m} \mathbf{1}_3 - \mathbf{U}^T \mathbf{U} \quad (28)$$

where λ_U is the largest eigenvalue of the product $\mathbf{U}^T \mathbf{U}$ and ϵ_m is a small offset. Both $\mathbf{U}^T \mathbf{U}$ and $\mathbf{W}^T \mathbf{W}$ can be interpreted as the contribution of all the kinematic branches to the composite spatial inertia. Therefore, ϵ_m and ϵ_D can be interpreted as minimal bounds regarding the contribution of the robot's base to the spatial inertia. Furthermore, $\hat{m}$ and $\hat{\mathbf{D}}$ do not affect any of the physical consistency constraints. Finally, Algorithm 1 summarizes the inertia matrix computation from the networks' outputs and (21), (23), (27), (28).

B. Potential Energy

Consider the potential energy of a system with n_b bodies,

$$P(\mathbf{q}) = \sum_{i=1}^{n_b} m_i \mathbf{g}^T \mathbf{r}_i^{\mathbf{W}} = \sum_{i=1}^{n_b} m_i \mathbf{g}^T (\mathbf{r}_U^{\mathbf{W}} + \mathbf{R} \mathbf{r}_i(\mathbf{q})) \quad (29)$$

$$= \mathbf{g}^T \left(\sum_{i=1}^{n_b} m_i \mathbf{r}_U^{\mathbf{W}} + \mathbf{R} \sum_{i=1}^{n_b} m_i \mathbf{r}_i(\mathbf{q}) \right), \quad (30)$$

Algorithm 1 Inertia Matrix Computation

Require: joint position $\mathbf{q}$, parameters $[\theta_m, \phi_R, \dots, \phi_{n_k}]$

- 1: $\mathbf{h}, \mathbf{L}_\Sigma = \mathbf{NN}_R(\mathbf{q}, \phi_R)$, $m = \theta_m^2$
- 2: **for** every $k \in \{1, \dots, n_k\}$ **do**
- 3: $\mathbf{L}_{kL}, \mathbf{L}_{kR}, \mathbf{L}_k = \mathbf{NN}_k(\mathbf{q}_k, \phi_k)$
- 4: **end for**
- 5: $\Sigma_R = \mathbf{L}_\Sigma^T \mathbf{L}_\Sigma$
- 6: $\mathbf{K} = [\mathbf{L}_{1L}, \dots, \mathbf{L}_{n_kL}]$, $\mathbf{W} = [\mathbf{L}_{1R}, \dots, \mathbf{L}_{n_kR}]$
- 7: $\mathbf{D} = \text{Tr}(\Sigma_R) \mathbf{1}_3 - \Sigma_R - \mathbf{W}^T \mathbf{W}$
- 8: $\beta = \epsilon_D + \text{softplus}(-\mu_D)$
- 9: $\hat{\mathbf{D}} = \mathbf{D} + \beta \mathbf{1}_3$
- 10: $\mathbf{L}_R = \text{reorderedChol}(\hat{\mathbf{D}})$
- 11: $\mathbf{L}_{RL} = ((\mathbf{S}_h - \mathbf{K}^T \mathbf{W}) \mathbf{L}_R^{-1})^T$
- 12: $\mathbf{U} = [\mathbf{L}_{RL}, \mathbf{K}]$
- 13: $\hat{m} = \text{softplus}(m - \lambda_U) + \epsilon_m + \lambda_U$
- 14: $\mathbf{T} = \hat{m} \mathbf{1}_3 - \mathbf{U}^T \mathbf{U}$
- 15: $\mathbf{L}_L = \text{reorderedChol}(\mathbf{T})$
- 16: Assemble $\mathbf{L}$ in (14) with $\mathbf{L}_L, \mathbf{U}, \mathbf{L}_R, \mathbf{W}, \mathbf{L}_1, \dots, \mathbf{L}_{n_k}$
- 17: $\mathbf{H} = \mathbf{L}^T \mathbf{L}$
- 18: **return** $\hat{m}, \mathbf{h}, \mathbf{H}$

where $\mathbf{g} \in \mathbb{R}^3$ is the gravity vector, $\mathbf{r}_U^{\mathbf{W}} \in \mathbb{R}^3$ is the robot's base position in the world frame, $\mathbf{R}$ is the rotation matrix from base to world frame. Note that the first sum yields the total mass m , and the last one the first mass moment $\mathbf{h}$

$$P(\mathbf{q}) = \mathbf{g}^T (m \mathbf{r}_U^{\mathbf{W}} + \mathbf{R} \mathbf{h}(\mathbf{q})). \quad (31)$$

Therefore, we compute the potential energy directly from the estimated terms of $\mathbf{H}$, without another network.

C. Generalized Coordinates

For floating-base systems, the orientation is usually described by Euler angles Θ or quaternions, while the angular velocities are used for the generalized velocities. Consequently, the generalized velocities are not the time derivative of the generalized position. To enable automatic differentiation, we convert the angular velocities to Euler angles time derivatives with $\dot{\Theta} = \mathbf{W}_\eta(\Theta) \omega$. This transformation also requires converting the angular wrenches to generalized torques. As the potential energy is defined in the world frame, $\mathbf{H}$ in (13) also has to be transformed from base to world frame. Therefore, we apply the following transformations to the total external torque and inertia matrix:

$$\tau_\nu = \begin{bmatrix} \mathbf{1}_3 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_\eta(\Theta)^T & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1}_{n_q} \end{bmatrix} \tau, \quad \mathbf{H}^{\mathbf{W}} = \mathbf{T}_H^T \mathbf{H} \mathbf{T}_H, \quad (32)$$

with

$$\mathbf{T}_H = \begin{bmatrix} \mathbf{R}(\Theta)^T & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_\eta(\Theta)^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1}_{n_q} \end{bmatrix}. \quad (33)$$

D. Loss Function

The Lagrangian (1) cannot be learned directly with standard supervised learning, as the system's energy is not observed. Following the approach of [36], the energies are

learned indirectly by minimizing the error from the estimated torque $f^{-1}(\boldsymbol{\nu}, \dot{\boldsymbol{\nu}}, \ddot{\boldsymbol{\nu}}, \boldsymbol{\phi})$ given by Algorithm 2. More precisely, by optimizing the following problem

$$\begin{aligned} \boldsymbol{\phi}^* = \arg \min_{\boldsymbol{\phi}} & \left\| \boldsymbol{\tau}_{\nu} - f^{-1}(\boldsymbol{\nu}, \dot{\boldsymbol{\nu}}, \ddot{\boldsymbol{\nu}}, \boldsymbol{\phi}) \right\|_{\mathbf{W}_{\tau}}^2 \\ & + w_{\mathbf{U}}(\hat{\mathbf{m}} - \mathbf{m})^2 + w_{\mathbf{D}}\text{softplus}(-\mu_{\mathbf{D}})^2, \end{aligned} \quad (34)$$

where $\mathbf{W}_{\tau}$ is the diagonal covariance of $\boldsymbol{\tau}_{\nu}$ in the training set; $w_{\mathbf{U}}$ and $w_{\mathbf{D}}$ are weights for auxiliaries terms from (25) and (26). Note that physical consistency is achieved as a hard constraint due to the structured parametrization, the auxiliary terms only encourage the intermediate values $\mathbf{m}$ and $\mathbf{D}$ to be positive and positive-definite, respectively, which empirically accelerated training during our experiments. We also apply the input layer $\mathbf{T}_{\mathbf{q}}(\mathbf{q}) = [\cos(\mathbf{q}), \sin(\mathbf{q})]$ to each network, as proposed in [36].

Algorithm 2 Torque Computation

Require: $\boldsymbol{\nu}, \dot{\boldsymbol{\nu}}, \ddot{\boldsymbol{\nu}}, \boldsymbol{\phi}$
1: $[\mathbf{r}_{\mathbf{U}}^{\mathbf{W}}, \boldsymbol{\Theta}, \mathbf{q}] = \boldsymbol{\nu}$
2: $\mathbf{m}, \mathbf{h}(\mathbf{q}), \mathbf{H}_t(\mathbf{q}) = \text{Algorithm 1}(\mathbf{q}, \boldsymbol{\phi})$
3: $P = (31)$ with $\mathbf{m}, \mathbf{h}(\mathbf{q})$
4: $\mathbf{H}^{\mathbf{W}} = \mathbf{T}_{\mathbf{H}}^{\mathbf{T}} \mathbf{H} \mathbf{T}_{\mathbf{H}}$
5: $\mathcal{L} = \frac{1}{2} \dot{\boldsymbol{\nu}}^{\mathbf{T}} \mathbf{H}^{\mathbf{W}} \dot{\boldsymbol{\nu}} - P$
6: **return** $\boldsymbol{\tau}_{\nu} = (2)$ with $\mathcal{L}$

VI. EXPERIMENTS

We evaluate our proposed architecture for system identification across floating-base robots of different sizes and degrees of freedom on simulation and real-world data.

A. Baselines

To evaluate the trade-off of physics assumptions and estimation accuracy, we compare our method against different baselines for estimating the inverse dynamics model. We employ a feed-forward neural network (FFNN) with inputs $\boldsymbol{\Theta}, \mathbf{q}, \dot{\boldsymbol{\nu}}, \ddot{\boldsymbol{\nu}}$. Additionally, we use DeLaN [36] with one network using $\mathbf{q}$ as input to estimate $\mathbf{C}(\mathbf{q})$ of $\mathbf{H}(\mathbf{q})$ in (32), and a second network that uses the entire $\boldsymbol{\nu}$ to estimate $P(\boldsymbol{\nu})$. To evaluate the proposed potential energy parametrization in (31), we implement a modified version of DeLaN, namely DeLaN-PP, which uses a single network for $\mathbf{H}(\mathbf{q})$ and computes P from its estimated terms using (31). We also consider a variant of our proposed network FeLaN, named FeLaN-BS, that incorporates only branch-induced sparsity and positive definiteness. In this variant, $\mathbf{L}_{\mathbf{L}}, \mathbf{L}_{\mathbf{RL}},$ and $\mathbf{L}_{\mathbf{R}}$ are estimated by a neural network. We add the input layer $\mathbf{T}_{\mathbf{q}}(\mathbf{q})$ to every entry, with the exception of the FFNN and the linear coordinates in DeLaN, which do not receive this input. We use 2 hidden layers of 32 neurons for FFNN, and 2 layers of 16 neurons for the other networks.

Regarding white-box methods, we use Mujoco XLA [38] as a differentiable recursive Newton-Euler (DNEA) function parametrized by 16 kinodynamic body parameters. In this

case, physical consistency is connected to the inertial parameters. Therefore, we evaluate parameterization with different degrees of physical consistency:

- DNEA NS [20]: unconstrained inertial parameters
- DNEA PD [20]: positive definiteness via $\mathbf{I}_b = \mathbf{C}_b \mathbf{C}_b^{\mathbf{T}}$
- DNEA Tri [20]: (6) with a geometric parametrization
- DNEA Cov [37]: parametrize $\boldsymbol{\Sigma}_b = \mathbf{C}_b \mathbf{C}_b^{\mathbf{T}}$
- DNEA Log [39]: log-Cholesky factorization of the pseudo-inertia matrix

All methods were implemented and trained in JAX [40] using stochastic gradient descent. Training details are provided in Appendix A.

B. Simulated Robots

For an initial evaluation, with robots mainly governed by rigid-body dynamics without actuator dynamics, we collect data at 100 Hz for Go2 and Talos in MuJoCo XLA [38]. For Talos, we consider only the first four arm joints, assuming the remaining joints as fixed. To ensure sufficient system excitation and data diversity, we command uniformly sampled desired velocities and base heights to be tracked by the model predictive controller proposed in [41] for both robots. For Talos' arms, we provide sine-wave reference trajectories with uniformly sampled amplitudes and frequencies. Each dataset yields a total of 50k samples. As the ground-truth model is available, we compute $\boldsymbol{\tau}_{\nu}$ using (3). Table II presents the normalized mean squared error (NMSE) of the estimated torques computed with $\mathbf{W}_{\tau}$ for all methods. To compare performance across robots, we define the relative NMSE as $r\text{NMSE}_i = (\text{NMSE}_i - \text{NMSE}_{\min}) / \text{NMSE}_{\max}$ for every method in the test set (also shown in Tab. II).

For Go2, the physically consistent DNEAs achieved the best performance, since the estimated parameters capture the full dynamics. Among the grey-box methods, FeLaN-BS showed similar performance, while the remaining grey-box methods were satisfactory. For Talos, FeLaN-BS and FeLaN achieved errors close to DNEAs, even outperforming DNEA Cov and DNEA Log. Although the amplitude errors remain small, the slightly worse performance of DNEA on Talos may be due to contacts adding larger spikes in the data when compared to Go2, which make the learning more challenging. For both robots, the FFNN performed worst. Nevertheless, FeLaN-BS achieved the best $r\text{NMSE}$.

Figure 1 shows each component in (3) of the linear force z and two Talos joint torques. All grey-box methods approximate the total torque well (see Fig. 1a), but only FeLaN preserves a stationary gravity component for z due to proper mass parametrization.

C. Real Robots

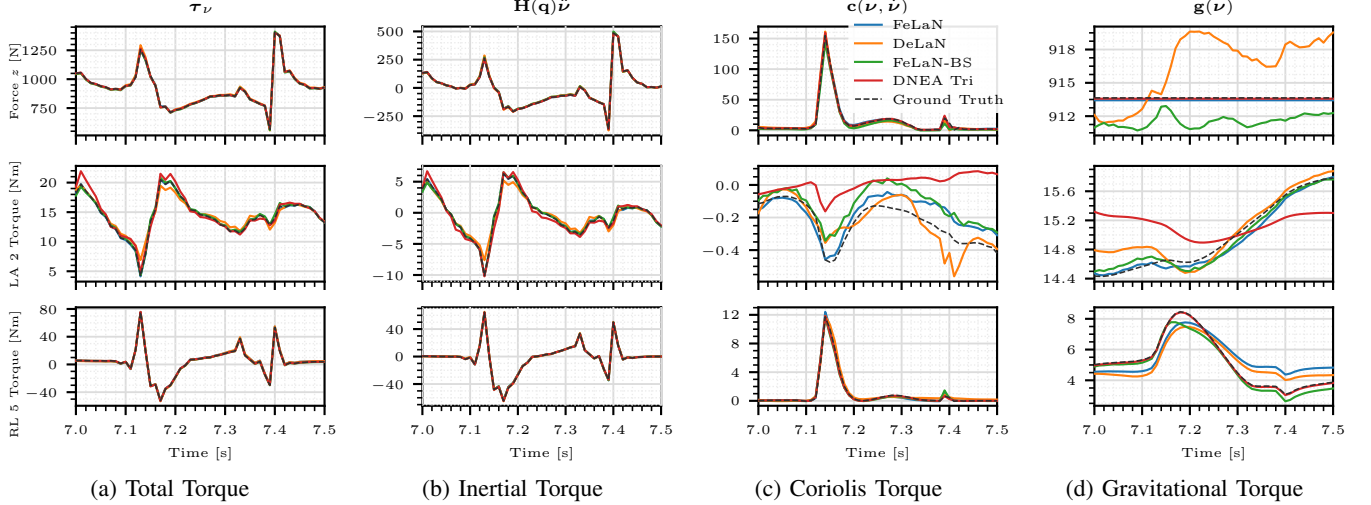
To assess all methods also on real data, we ran tests with 4 different robots: HyQReal2, Spot, Spot+Arm, and Talos.

As a ground truth model is not available, we need to estimate $\boldsymbol{\tau}_{\nu}$, which in the case of a legged robot is

$$\boldsymbol{\tau}_{\nu} = \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\tau}_{\mathbf{q}} \end{bmatrix} + \sum_{i=1}^{n_c} \mathbf{J}_{\mathbf{c}i}^{\mathbf{T}} \mathbf{f}_{\mathbf{c}i} \quad (35)$$

TABLE II: Inverse dynamics normalized mean squared error for simulated Go2 and Talos.

Method	Go2		Talos		r NMSE
	Train	Test	Train	Test	
FFNN	$9.5\text{e-}1 \pm 4.3\text{e-}2$	$1.4\text{e}0 \pm 9.3\text{e-}2$	$5.3\text{e}0 \pm 2.8\text{e-}1$	$5.0\text{e}0 \pm 2.1\text{e-}1$	9.9e-1
DeLaN	$2.6\text{e-}2 \pm 1.0\text{e-}2$	$3.1\text{e-}2 \pm 1.3\text{e-}2$	$3.1\text{e-}1 \pm 1.1\text{e-}1$	$2.8\text{e-}1 \pm 8.0\text{e-}2$	3.4e-2
DeLaN-PP	$2.5\text{e-}2 \pm 6.9\text{e-}3$	$3.3\text{e-}2 \pm 2.0\text{e-}2$	$9.0\text{e-}1 \pm 1.3\text{e}0$	$8.2\text{e-}1 \pm 1.1\text{e}0$	8.8e-2
FeLaN-BS	$1.7\text{e-}3 \pm 5.4\text{e-}4$	$3.0\text{e-}3 \pm 5.1\text{e-}3$	$5.9\text{e-}2 \pm 1.3\text{e-}2$	$5.4\text{e-}2 \pm 1.2\text{e-}2$	8.7e-4
FeLaN	$4.3\text{e-}2 \pm 1.9\text{e-}3$	$5.9\text{e-}2 \pm 3.5\text{e-}3$	$3.1\text{e-}1 \pm 1.2\text{e}0$	$2.7\text{e-}1 \pm 1.1\text{e}0$	4.4e-2
DNEA NS	$1.3\text{e-}2 \pm 1.9\text{e-}2$	$1.2\text{e-}2 \pm 1.8\text{e-}2$	$2.3\text{e-}1 \pm 6.3\text{e-}1$	$2.0\text{e-}1 \pm 5.5\text{e-}1$	1.8e-2
DNEA PD	$6.8\text{e-}4 \pm 6.2\text{e-}5$	$6.3\text{e-}4 \pm 1.3\text{e-}4$	$2.8\text{e-}1 \pm 7.0\text{e-}1$	$2.2\text{e-}1 \pm 5.6\text{e-}1$	1.6e-2
DNEA Tri	$6.8\text{e-}4 \pm 6.1\text{e-}5$	$6.3\text{e-}4 \pm 1.1\text{e-}4$	$3.9\text{e-}1 \pm 6.5\text{e-}1$	$2.9\text{e-}1 \pm 4.6\text{e-}1$	2.4e-2
DNEA Cov	$6.8\text{e-}4 \pm 4.7\text{e-}5$	$6.2\text{e-}4 \pm 1.2\text{e-}4$	$3.7\text{e-}1 \pm 1.1\text{e}0$	$3.1\text{e-}1 \pm 9.0\text{e-}1$	2.6e-2
DNEA Log	$6.7\text{e-}4 \pm 5.0\text{e-}5$	$6.1\text{e-}4 \pm 8.4\text{e-}5$	$5.6\text{e-}1 \pm 1.0\text{e}0$	$4.7\text{e-}1 \pm 8.4\text{e-}1$	4.1e-2


 Fig. 1: Estimated force z and torques for left arm (LA) second joint, and right leg (RL) fifth joint for simulated Talos.

where $\tau_q \in \mathbb{R}^{n_q}$ are the joint torques; $\mathbf{J}_{ci}$ and $\mathbf{f}_{ci}$ are the i th contact Jacobian and force. All evaluated robots are equipped with torque sensors. For contact forces, Talos uses an ankle sensor, HyQReal2 relies on a force estimator based on its dynamical model, and Spot's forces were measured with external 6D force plates. For HyQReal2 and Talos, trajectories similar to Go2 in simulation were executed, producing 75k and 125k samples, respectively. To keep Spot on the force plates, small velocity commands were applied while trotting in place, resulting in 66k samples. For the arm, sine-wave trajectories with uniformly sampled parameters were tracked by a position controller, yielding 45k samples.

Although the estimation of τ_ν requires information about the robot's kinematics, all the methods still learn agnostic to any kinematic parameter.

Table III reports the NMSE of the different methods across robots. Unlike in simulation, real robot dynamics are affected by actuator and contact dynamics, friction, noise, state estimation errors, and other unmodeled effects. Consequently, DNEA methods struggle to accurately estimate the inverse dynamics torques due to limited model complexity, being outperformed by the FFNN. In contrast, DeLaN-PP, FeLaN-BS, and FeLaN achieved the lowest r NMSE, highlighting the effectiveness of the proposed potential energy parametrization. The results highlight the advantage of FeLaN, which

achieved similar performance with a more interpretable and physical consistent model, preserving properties such as stationary mass and branch-induced sparsity.

VII. CONCLUSION

In this work, we introduced a novel physically consistent parametrization of the floating-base inertia matrix, based on a reordered Cholesky factorization, which achieves branch-induced sparsity and ensures full physical consistency of the composite spatial inertia. Building on this parametrization, we proposed FeLaN, a grey-box method for physically consistent system identification of floating-base dynamics. FeLaN learns the whole-body dynamics model using only prior knowledge on the kinematic chains, without requiring any knowledge of dynamics or other kinematic parameters. We validated FeLaN on both simulated and real robot data, showing performance comparable to state-of-the-art methods while offering greater interpretability.

As future work, we plan to explore using FeLaN for locomotion and loco-manipulation tasks, either as a learned dynamical model or alongside adaptive control techniques.

APPENDIX

A. Training

The datasets are split into 90% for training and 10% for testing. Table IV lists the used hyperparameters.

TABLE III: Inverse dynamics normalized mean squared error for the real robot data.

Method	HyQReal2		Spot		Spot with Arm		Talos		r NMSE
	Train	Test	Train	Test	Train	Test	Train	Test	
FFNN	10.2 $\pm$ 16.1	11.5 $\pm$ 18.0	5.4 $\pm$ 0.2	4.0 $\pm$ 0.3	7.8 $\pm$ 0.6	9.5 $\pm$ 0.3	28.0 $\pm$ 0.1	27.0 $\pm$ 0.1	0.210
DeLaN	88.3 $\pm$ 5.8	278.0 $\pm$ 167.0	6.5 $\pm$ 0.2	4.0 $\pm$ 0.2	20.2 $\pm$ 12.0	10.9 $\pm$ 0.9	4.2 $\pm$ 0.1	4.5 $\pm$ 0.4	0.268
DeLaN-PP	8.7 $\pm$ 0.9	10.1 $\pm$ 1.1	7.4 $\pm$ 0.1	4.9 $\pm$ 0.3	13.1 $\pm$ 0.2	11.1 $\pm$ 0.2	9.0 $\pm$ 0.5	8.6 $\pm$ 0.4	0.102
FeLaN-BS	10.7 $\pm$ 0.8	12.6 $\pm$ 0.7	7.8 $\pm$ 0.1	4.9 $\pm$ 0.2	12.4 $\pm$ 0.3	10.9 $\pm$ 1.1	9.5 $\pm$ 0.2	8.9 $\pm$ 0.2	0.106
FeLaN	10.5 $\pm$ 0.2	14.8 $\pm$ 0.8	8.4 $\pm$ 0.9	5.1 $\pm$ 0.3	14.0 $\pm$ 0.5	11.4 $\pm$ 0.6	9.6 $\pm$ 0.4	9.1 $\pm$ 0.4	0.126
DNEA NS	16.0 $\pm$ 0.3	20.3 $\pm$ 0.3	11.1 $\pm$ 0.1	6.0 $\pm$ 0.1	17.6 $\pm$ 0.2	13.1 $\pm$ 0.2	18.5 $\pm$ 0.8	16.9 $\pm$ 0.7	0.271
DNEA PD	16.2 $\pm$ 0.2	20.4 $\pm$ 0.4	11.2 $\pm$ 0.1	6.0 $\pm$ 0.1	17.7 $\pm$ 0.1	13.2 $\pm$ 0.2	19.6 $\pm$ 1.2	18.0 $\pm$ 1.0	0.287
DNEA Tri	16.2 $\pm$ 0.2	20.7 $\pm$ 0.4	11.2 $\pm$ 0.1	6.0 $\pm$ 0.1	17.8 $\pm$ 0.1	13.3 $\pm$ 0.2	19.9 $\pm$ 0.7	18.2 $\pm$ 0.6	0.290
DNEA Cov	16.3 $\pm$ 0.2	20.7 $\pm$ 0.3	11.3 $\pm$ 0.1	6.1 $\pm$ 0.1	17.8 $\pm$ 0.1	13.3 $\pm$ 0.2	20.0 $\pm$ 1.1	18.4 $\pm$ 1.0	0.294
DNEA Log	16.4 $\pm$ 0.2	20.8 $\pm$ 0.3	11.3 $\pm$ 0.1	6.0 $\pm$ 0.1	17.8 $\pm$ 0.1	13.3 $\pm$ 0.2	20.0 $\pm$ 0.9	18.4 $\pm$ 0.7	0.293

TABLE IV: Training hyperparameters.

Parameter	Value	Parameter	Value
Activation	Tanh	ϵ_L	0.01
Batch Size	1024	ϵ_m	0.1
Learning Rate	5×10^{-4}	ϵ_D	0.01
Weight decay	10^{-5}	Gradient Clip Norm	1000

REFERENCES

- [1] S. Greydanus *et al.*, “Hamiltonian neural networks,” in *Advances in Neural Information Processing Systems*, 2019.
- [2] M. Lutter *et al.*, “Deep lagrangian networks: Using physics as model prior for deep learning,” in *International Conference on Learning Representations*, 2019.
- [3] M. Lutter, K. Listmann, and J. Peters, “Deep lagrangian networks for end-to-end learning of energy-based control for under-actuated systems,” *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.
- [4] T. Duong *et al.*, “Port-hamiltonian neural ode networks on lie groups for robot dynamics learning and control,” *IEEE Transactions on Robotics*, 2024.
- [5] A. Altawaitan *et al.*, “Hamiltonian dynamics learning from point cloud observations for nonholonomic mobile robot control,” *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.
- [6] P. Toth *et al.*, “Hamiltonian generative networks,” in *International Conference on Learning Representations*, 2020.
- [7] L. Schulze *et al.*, “Context-aware deep lagrangian networks for model predictive control,” 2025.
- [8] R. Featherstone, *Rigid Body Dynamics Algorithms*. Springer US, 2008.
- [9] S. Traversaro *et al.*, “Identification of fully physical consistent inertial parameters using optimization on manifolds,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.
- [10] R. Featherstone, “Efficient factorization of the joint-space inertia matrix for branched kinematic trees,” *The International Journal of Robotics Research*, June 2005.
- [11] P. Kotecha *et al.*, “Investigating lagrangian neural networks for infinite horizon planning in quadrupedal locomotion,” 2025.
- [12] G. Buriani *et al.*, “Symbolic learning of interpretable reduced-order models for jumping quadruped robots,” 2025.
- [13] Unitree Robotics. (2023) Go2 quadruped robot. Accessed: 2025-09-15. [Online]. Available: <https://www.unitree.com/go2>
- [14] Boston Dynamics. (2024) Spot: The agile mobile robot. Accessed: 2025-09-15. [Online]. Available: <https://www.bostondynamics.com/products/spot/>
- [15] O. Stasse *et al.*, “Talos: A new humanoid research platform targeted for industrial applications,” in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, 2017.
- [16] C. Semini *et al.*, “Brief introduction to the quadruped robot hyqreal,” in *International Conference on Robotics and Intelligent Machines (IRIM)*. IRIM, 2019.
- [17] T. Lee *et al.*, “Robot model identification and learning: A modern perspective,” *Annual Review of Control, Robotics, and Autonomous Systems*, 2024.
- [18] T. Lee, B. D. Lee, and F. C. Park, “Optimal excitation trajectories for mechanical systems identification,” *Automatica*, 2021.
- [19] M. Lutter, J. Silberbauer, J. Watson, and J. Peters, “Differentiable physics models for real-world offline model-based reinforcement learning,” *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021.
- [20] G. Sutanto *et al.*, “Encoding physical constraints in differentiable newton-euler algorithm,” ser. *Proceedings of Machine Learning Research*, 2020.
- [21] S. Khorshidi *et al.*, “Physically-consistent parameter identification of robots in contact,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2025.
- [22] Y. Ogawa *et al.*, “Dynamic parameters identification of a humanoid robot using joint torque sensors and/or contact forces,” in *2014 IEEE-RAS International Conference on Humanoid Robots*, 2014.
- [23] K. Ayusawa *et al.*, “Identifiability and identification of inertial parameters using the underactuated base-link dynamics for legged multibody systems,” *The International Journal of Robotics Research*, 2014.
- [24] R. Grandia *et al.*, “Contact invariant model learning for legged robot locomotion,” *IEEE Robotics and Automation Letters*, 2018.
- [25] E. Arcari *et al.*, “Bayesian multi-task learning mpc for robotic mobile manipulation,” *IEEE Robotics and Automation Letters*, 2023.
- [26] E. Heiden *et al.*, “Neuralsim: Augmenting differentiable simulators with neural networks,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.
- [27] A. Nagabandi *et al.*, “Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.
- [28] J. Watson *et al.*, “Machine learning with physics knowledge for prediction: A survey,” *Trans. on Machine Learning Research*, 2025.
- [29] M. Finzi *et al.*, “Simplifying hamiltonian and lagrangian neural networks via explicit constraints,” *Advances in neural information processing systems*, 2020.
- [30] T. Duong and N. Atanasov, “Hamiltonian-based neural ODE networks on the SE(3) manifold for dynamics learning and control,” in *Robotics: Science and Systems (RSS)*, 2021.
- [31] V. Duruisseaux *et al.*, “Lie group forced variational integrator networks for learning and control of robot systems,” in *Learning for Dynamics and Control Conference*, 2023.
- [32] D. Ordoñez-Apaez *et al.*, “Morphological symmetries in robotics,” *The International Journal of Robotics Research*, 2025.
- [33] F. Xie *et al.*, “Morphological-symmetry-equivariant heterogeneous graph neural network for robotic dynamics learning,” in *Proceedings of the 7th Annual Learning for Dynamics & Control Conference*, ser. *Proceedings of Machine Learning Research*, 2025.
- [34] J. eun Lee *et al.*, “Sample efficient dynamics learning for symmetrical legged robots: Leveraging physics invariance and geometric symmetries,” 2022.
- [35] S. L. Brunton *et al.*, “Discovering governing equations from data by sparse identification of nonlinear dynamical systems,” *Proceedings of the National Academy of Sciences*, 2016.
- [36] M. Lutter and J. Peters, “Combining physics and deep learning to learn continuous-time dynamics models,” *The International Journal of Robotics Research*, 2023.

- [37] P. M. Wensing *et al.*, “Linear matrix inequalities for physically consistent inertial parameter identification: A statistical perspective on the mass distribution,” *IEEE Robotics and Automation Letters*, 2018.
- [38] E. Todorov *et al.*, “Mujoco: A physics engine for model-based control,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.
- [39] C. Rucker and P. M. Wensing, “Smooth parameterization of rigid-body inertia,” *IEEE Robotics and Automation Letters*, 2022.
- [40] J. Bradbury *et al.*, “JAX: composable transformations of Python+NumPy programs,” 2018. [Online]. Available: <http://github.com/google/jax>
- [41] L. Amatiucci *et al.*, “Primal-dual ilqr for gpu-accelerated learning and control in legged robots,” 2025.