Hamiltonian-based Neural ODE Networks on the SE(3) Manifold For Dynamics Learning and Control

Thai Duong and Nikolay Atanasov
Department of Electrical and Computer Engineering
University of California, San Diego
La Jolla, CA 92093 USA
Email: {tduong, natanasov}@ucsd.edu

***Abstract*—Accurate models of robot dynamics are critical for safe and stable control and generalization to novel operational conditions. Hand-designed models, however, may be insufficiently accurate, even after careful parameter tuning. This motivates the use of machine learning techniques to approximate the robot dynamics over a training set of state-control trajectories. The dynamics of many robots, including ground, aerial, and underwater vehicles, are described in terms of their $SE(3)$ pose and generalized velocity, and satisfy conservation of energy principles. This paper proposes a Hamiltonian formulation over the $SE(3)$ manifold of the structure of a neural ordinary differential equation (ODE) network to approximate the dynamics of a rigid body. In contrast to a black-box ODE network, our formulation guarantees total energy conservation by construction. We develop energy shaping and damping injection control for the learned, potentially under-actuated $SE(3)$ Hamiltonian dynamics to enable a unified approach for stabilization and trajectory tracking with various platforms, including pendulum, rigid-body, and quadrotor systems.**

Supplementary Material

Software and videos supplementing this paper:
https://thaipduong.github.io/SE3HamDL/

I. Introduction

Motion planning and optimal control algorithms depend on the availability of accurate system dynamics models. Models obtained from first principles and calibrated over a small set of parameters via system identification [19] are widely used for unmanned ground vehicles (UGVs), unmanned aerial vehicles (UAVs), and unmanned underwater vehicles (UUVs). Such models often over-simplify or even incorrectly describe the underlying structure of the dynamical system, leading to bias and modeling errors that cannot be corrected by optimization over few parameters. Data-driven techniques [26, 9, 42, 29, 6] have emerged as a powerful approach to approximate the function describing the system dynamics with an over-parameterized machine learning model, trained over a dataset of system state and control trajectories. Neural networks are especially expressive function approximation models, capable of identifying and generalizing dynamics interaction patterns from the training data. Training neural network models, however, typically requires large amounts of data and computation time, which is impractical in robotics applications. Recent works [22, 14, 8, 13, 5, 32] have considered a hybrid approach to this problem, where prior knowledge of the physics, governing the system dynamics, is used to assist the learning process. The dynamics of physical systems obey kinematic constraints and energy conservation laws. While these laws are known to be universally true, a black-box machine learning model might struggle to infer them for the training data, causing poor generalization. Instead, prior knowledge may be encoded into the learning model, e.g., using a prior distribution [9], a graph-network forward kinematic model [34], or a network architecture reflecting the structure of Lagrangian [22] or Hamiltonian [13] mechanical systems. Moreover, many physical robot platforms are composed of rigid-body interconnections and their state evolution respects the position and orientation kinematics over the $SE(3)$ manifold [23]. The goal of this paper is to incorporate both the $SE(3)$ kinematics and energy conservation constraints in the structure of the dynamics learning model. We only focus on learning the dynamics of a single rigid body but this is already sufficient to model many UGV, UAV, and UUV robots. We also aim to design a unified control approach that attempts to stabilize the learned model without depending on any prior knowledge of its parameters or level of under-actuation.

Lagrangian and Hamiltonian mechanics [20, 15] provide physical system descriptions that can be integrated into the structure of a neural network [13, 3, 5, 11, 43, 41]. Prior work, however, has only considered vector-valued states, when designing Lagrangian- or Hamiltonian-structured neural networks. This limits the applicability of these techniques as most interesting robot systems have states on the $SE(3)$ manifold. Hamiltonian equations of motion are available for orientation states but existing formulations rely predominantly on 3 dimensional vector parametrizations, such as Euler angles [24, 35], which suffer from singularities. We design a neural ordinary differential equation (ODE) network [4], whose structure captures Hamiltonian dynamics over the $SE(3)$ manifold [17]. Our model guarantees by construction that its long-term trajectory predictions satisfy $SE(3)$ constraints and total energy conservation. Inspired by [43, 44], we model kinetic energy and potential energy by separate neural networks, each governed by a set of Hamiltonian equations on $SE(3)$. The Hamiltonian formulation can be generalized to a Port-Hamiltonian one [38], enabling us to design an energy-based controller for trajectory tracking. In summary, this paper makes the following **contributions**.

- We design a neural ODE model that respects the structure of Hamiltonian dynamics over the $SE(3)$ manifold to enable data-driven learning of rigid-body system dynamics.
- We develop a unified controller for (Port-)Hamiltonian $SE(3)$ dynamics that achieves trajectory tracking if permissible by the system's degree of underactuation.
- We demonstrate our dynamics learning and control approach for fully-actuated pendulum and rigid-body systems and an under-actuated quadrotor system.

II. Related Work

Physics-guided dynamics learning, in which prior knowledge about a physical system is integrated into the design of a machine learning model, has received significant attention recently [41]. Models designed with structure respecting kinematic constraints [34], symmetry [33, 39], Lagrangian mechanics [32, 22, 14, 8, 21] or Hamiltonian mechanics [13, 3, 5, 11, 43, 41] guarantee that the laws of physics are satisfied by construction, regardless of the training data. Sanchez-Gonzalez et al. [34] design graph neural networks to represent the kinematic structure of complex dynamical systems and demonstrate forward model learning and online planning via gradient-based trajectory optimization. Ruthotto et al. [33] propose a partial differential equation (PDE) interpretation of convolutional neural networks and derive new parabolic and hyperbolic ResNet architectures guided by PDE theory. Wang et al. [39] design symmetry equivariant neural network models, encoding rotation, scaling, and uniform motion, to learn physical dynamics that are robust to symmetry group distributional shifts. Lagrangian-based methods [32, 22, 14, 8, 21] design neural network models for physical systems, based on the Euler-Lagrange differential equations of motion [20, 15], in terms of generalized coordinates $\mathfrak{q}$, their velocity $\dot{\mathfrak{q}}$ and a Lagrangian function $\mathcal{L}(\mathfrak{q}, \dot{\mathfrak{q}})$, defined as the difference between the kinetic and potential energies. The energy terms are modeled by neural networks, either separately [22, 21] or together [8]. Hamiltonian-based methods [13, 3, 5, 11, 43, 41] use a Hamiltonian formulation [20, 15] of the system dynamics, instead, in terms of generalized coordinates $\mathfrak{q}$, generalized momenta $\mathfrak{p}$, and a Hamiltonian function, $\mathcal{H}(\mathfrak{q}, \mathfrak{p})$, representing the total energy of the system. Greydanus et al. [13] model the Hamiltonian as a neural network and update its parameters by minimizing the discrepancy between its symplectic gradients and the time derivatives of the states $(\mathfrak{q}, \mathfrak{p})$. This approach, however, requires that the state time derivatives are available in the training data set. Chen et al. [5] and Zhong et al. [43] relax this assumption by using differentiable leapfrog integrators [18] and differentiable ODE solvers [4], respectively. The need for time derivatives of the states is eliminated by back-propagating a loss function measuring state discrepancy through the ODE solvers via the adjoint method. Toth et al. [37] show that, instead of from state trajectories, the Hamiltonian function can be learned from high-dimensional image observations. Finzi et al. [11] show that using Cartesian coordinates with explicit constraints improves both the accuracy and data efficiency for the Lagrangian- and Hamiltonian-based approaches. In a recent closely related work, Zhong et al. [44] showed that dissipating elements, such as friction or air drag, can be incorporated in a Hamiltonian-based neural ODE network by reformulating the system dynamics in Port-Hamiltonian form [38].

In addition to learning the dynamics of a physical system from training data, this paper considers designing control methods for stabilization and trajectory tracking, relying only on the Hamiltonian dynamics structure rather than a particular system realization. The Hamiltonian formulation and its Port-Hamiltonian generalization [38] are built around the notion of system energy and, hence, are naturally related to control techniques for stabilization aiming to minimize the total energy. Since the minimum point of the Hamiltonian might not correspond to a desired regulation point, the control design needs to inject additional energy to ensure that the minimum of the total energy is at the desired equilibrium. For fully-actuated (Port-)Hamiltonian systems, it is sufficient to shape the potential energy only using an energy-shaping and damping-injection (ES-DI) controller [38]. For under-actuated systems, both the kinetic and potential energies needs to be shaped, e.g., via interconnection and damping assignment passivity-based control (IDA-PBC) [38, 27, 1, 7]. Wang and Goldsmith [40] extend the IDA-PBC controller to trajectory tracking problem. Closely related to our controller design, Souza et. al. [36] apply this technique to design controller for an underactuated quadrotor but use Euler angles as the orientation representation.

While existing Hamiltonian-based learning methods are designed for generalized coordinates in $\mathbb{R}^n$, we develop a neural ODE network for learning Hamiltonian dynamics over the $SE(3)$ manifold [17]. We connect Hamiltonian-dynamics learning with the idea of IDA-PBC control to allow stabilization of any rigid-body robot without relying on its model parameters a priori. We design the trajectory tracking controller for under-actuated systems based on the IDA-PBC approach and show how to construct desired pose and momentum trajectories given only a desired position trajectory. We demonstrate the tight integration of dynamics learning and control to achieve closed-loop trajectory tracking with fully- and under-actuated quadrotor robots.

III. Problem Statement

Consider a robot with state $\mathbf{x}$ consisting of its position $\mathbf{p} \in \mathbb{R}^3$, orientation $\mathbf{R} \in SO(3)$, body-frame linear velocity $\mathbf{v} \in \mathbb{R}^3$, and body-frame angular velocity $\boldsymbol{\omega} \in \mathbb{R}^3$. Let $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$ characterize the robot dynamics with control input $\mathbf{u}$. For example, the control input of an Ackermann-drive UGV may include its linear acceleration and steering angle rate, and that of a quadrotor UAV may include the total thrust and moment generated by the propellers.

We assume that the function $\mathbf{f}$ specifying the robot dynamics is unknown and aim to approximate it using a dataset $\mathcal{D}$ of state and control trajectories. Specifically, let $\mathcal{D} = \{t^{(i)}_{0:N}, \mathbf{x}^{(i)}_{0:N}, \mathbf{u}^{(i)}\}_{i=1}^{D}$ consist of D state sequences $\mathbf{x}^{(i)}_{0:N}$, obtained by applying a constant control input $\mathbf{u}^{(i)}$ to the

system with initial condition $\mathbf{x}_0^{(i)}$ at time $t_0^{(i)}$ and sampling its state $\mathbf{x}^{(i)}(t_n^{(i)}) =: \mathbf{x}_n^{(i)}$ at times $t_0^{(i)} < t_1^{(i)} < \ldots < t_N^{(i)}$. We aim to find a function $\bar{\mathbf{f}}_{\boldsymbol{\theta}}$ with parameters $\boldsymbol{\theta}$ that approximates the true dynamics $\mathbf{f}$ well on the dataset $\mathcal{D}$. To optimize the parameters $\boldsymbol{\theta}$, we roll out the approximate dynamics $\bar{\mathbf{f}}_{\boldsymbol{\theta}}$ with initial state $\mathbf{x}_0^{(i)}$ and constant control $\mathbf{u}^{(i)}$ and minimize the discrepancy between the computed state sequence $\bar{\mathbf{x}}_{1:N}^{(i)}$ and the true state sequence $\mathbf{x}_{1:N}^{(i)}$ in $\mathcal{D}$. The dynamics learning problem is summarized below.

Problem 1. Given a dataset $\mathcal{D} = \{t_{0:N}^{(i)}, \mathbf{x}_{0:N}^{(i)}, \mathbf{u}^{(i)}\}_{i=1}^{D}$ and a function $\bar{\mathbf{f}}_{\boldsymbol{\theta}}$, find the parameters $\boldsymbol{\theta}$ that minimize:

$$\begin{aligned} \min_{\boldsymbol{\theta}} \; & \sum_{i=1}^{D} \sum_{n=1}^{N} \ell(\mathbf{x}_j^{(i)}, \bar{\mathbf{x}}_j^{(i)}) \\ \text{s.t. } & \dot{\bar{\mathbf{x}}}^{(i)}(t) = \bar{\mathbf{f}}_{\boldsymbol{\theta}}(\bar{\mathbf{x}}^{(i)}(t), \mathbf{u}^{(i)}), \;\; \bar{\mathbf{x}}^{(i)}(t_0) = \mathbf{x}_0^{(i)}, \\ & \bar{\mathbf{x}}_n^{(i)} = \bar{\mathbf{x}}^{(i)}(t_n), \;\; \forall n = 1, \ldots, N, \;\; \forall i = 1, \ldots, D, \end{aligned} \tag{1}$$

where ℓ is a distance metric on the state space.

Further, we aim to design a feedback controller that attempts to track a desired state trajectory $\mathbf{x}^*(t)$, $t \geq t_0$, for any learned realization $\bar{\mathbf{f}}_{\boldsymbol{\theta}}$ of the robot dynamics.

Problem 2. Given an initial condition $\mathbf{x}_0$ at time t_0, desired state trajectory $\mathbf{x}^*(t)$, $t \geq t_0$, and learned dynamics $\bar{\mathbf{f}}_{\boldsymbol{\theta}}$, design a feedback control law $\mathbf{u} = \boldsymbol{\pi}(\mathbf{x}, \boldsymbol{\theta}, \mathbf{x}^*(t))$ such that $\limsup_{t\to\infty} \ell(\mathbf{x}(t), \mathbf{x}^*(t))$ is bounded.

In our setting, the robot kinematics need to evolve over the $SE(3)$ manifold and the dynamics $\mathbf{f}(\mathbf{x}, \mathbf{u})$ respect the law of energy conservation, when there is no control input, $\mathbf{u} = \mathbf{0}$. We embed these constraints in the structure of the parametric function $\bar{\mathbf{f}}_{\boldsymbol{\theta}}$. We review the $SE(3)$ kinematics and Hamiltonian dynamics equations next.

IV. Preliminaries

A. SE(3) Kinematics

Consider a fixed world inertial frame of reference $\{W\}$ and a rigid body with a body-fixed frame $\{B\}$ attached to its center of mass. The pose of $\{B\}$ in $\{W\}$ is determined by the position $\mathbf{p} = [x, y, z]^\top \in \mathbb{R}^3$ of the center of mass and the orientation of the coordinate axes of $\{B\}$:

$$\mathbf{R} = \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 \end{bmatrix}^\top \in SO(3), \tag{2}$$

where $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3 \in \mathbb{R}^3$ are the rows of the rotation matrix $\mathbf{R}$, which is an element of the special orthogonal group:

$$SO(3) = \left\{ \mathbf{R} \in \mathbb{R}^{3\times 3} : \mathbf{R}^\top \mathbf{R} = \mathbf{I}, \det(\mathbf{R}) = 1 \right\}. \tag{3}$$

The rigid-body position and orientation can be combined in a single pose matrix $\mathbf{T} \in SE(3)$, which is an element of the special Euclidean group:

$$SE(3) = \left\{ \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0}^\top & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4} : \mathbf{R} \in \text{SO(3)}, \mathbf{p} \in \mathbb{R}^3 \right\}. \tag{4}$$

The kinematic equations of motion of the rigid body are determined by the linear velocity $\mathbf{v} \in \mathbb{R}^3$ and angular velocity $\boldsymbol{\omega} \in \mathbb{R}^3$ of frame $\{B\}$ with respect to frame $\{W\}$, expressed in body-frame coordinates. The generalized velocity $\boldsymbol{\zeta} = [\mathbf{v}^\top, \boldsymbol{\omega}^\top]^\top \in \mathbb{R}^6$ determines the rate of change of the rigid-body pose according to the $SE(3)$ kinematics:

$$\dot{\mathbf{T}} = \mathbf{T}\hat{\boldsymbol{\zeta}} =: \mathbf{T} \begin{bmatrix} \hat{\boldsymbol{\omega}} & \mathbf{v} \\ \mathbf{0}^\top & 0 \end{bmatrix}, \tag{5}$$

where we overload $\hat{\cdot}$ to denote the mapping from a vector $\boldsymbol{\zeta} \in \mathbb{R}^6$ to a 4×4 twist matrix $\hat{\boldsymbol{\zeta}}$ in the Lie algebra $\mathfrak{se}(3)$ of $SE(3)$ and from a vector $\boldsymbol{\omega} \in \mathbb{R}^3$ to a 3×3 skew-symmetric matrix $\hat{\boldsymbol{\omega}}$ in the Lie algebra $\mathfrak{so}(3)$ of $SO(3)$:

$$\hat{\boldsymbol{\omega}} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}. \tag{6}$$

Please refer to [2] for an excellent introduction to the use of matrix Lie groups in robot state estimation problems.

B. Hamiltonian and Port-Hamiltonian Dynamics

There are three predominant formulations of classical mechanics [15] for describing the motion of macroscopic objects: Newtonian, Lagrangian, and Hamiltonian. Newtonian mechanics models the dynamics of mobile objects using forces and Cartesian coordinates according to the Newton's laws of motion. Lagrangian and Hamiltonian mechanics use generalized coordinates and energy in their formulations, which simplifies the equations of motion and reveals conserved quantities and their symmetries. Lagrangian mechanics considers generalized coordinates $\mathfrak{q} \in \mathbb{R}^n$ and velocity $\dot{\mathfrak{q}} \in \mathbb{R}^n$, and defines a Lagrangian function $\mathcal{L}(\mathfrak{q}, \dot{\mathfrak{q}})$ as the difference between kinetic energy $\frac{1}{2}\dot{\mathfrak{q}}^\top \mathbf{M}(\mathfrak{q})\dot{\mathfrak{q}}$ and the potential energy $V(\mathfrak{q})$:

$$\mathcal{L}(\mathfrak{q}, \dot{\mathfrak{q}}) = \frac{1}{2}\dot{\mathfrak{q}}^\top \mathbf{M}(\mathfrak{q})\dot{\mathfrak{q}} - V(\mathfrak{q}), \tag{7}$$

where the symmetric positive definite matrix $\mathbf{M}(\mathfrak{q}) \in \mathbb{S}_{\succ 0}^{n\times n}$ represents the generalized mass of the system. Starting from the Lagrangian formulation, Hamiltonian mechanics expresses the system dynamics in terms of the generalized coordinates $\mathfrak{q} \in \mathbb{R}^n$ and generalized momenta $\mathfrak{p} \in \mathbb{R}^n$, defined as:

$$\mathfrak{p} = \frac{\partial \mathcal{L}(\mathfrak{q}, \dot{\mathfrak{q}})}{\partial \dot{\mathfrak{q}}} = \mathbf{M}(\mathfrak{q})\dot{\mathfrak{q}}. \tag{8}$$

Instead of the Lagrangian function, a Hamiltonian function $\mathcal{H}(\mathfrak{q}, \mathfrak{p})$, representing the total energy of the dynamical systems, is obtained by a Legendre transformation of $\mathcal{L}(\mathfrak{q}, \dot{\mathfrak{q}})$:

$$\mathcal{H}(\mathfrak{q}, \mathfrak{p}) = \mathfrak{p}^\top \dot{\mathfrak{q}} - \mathcal{L}(\mathfrak{q}, \dot{\mathfrak{q}}) = \frac{1}{2}\mathfrak{p}^\top \mathbf{M}(\mathfrak{q})^{-1}\mathfrak{p} + V(\mathfrak{q}). \tag{9}$$

The Hamiltonian characterizes the system dynamics according to the equations:

$$\dot{\mathfrak{q}} = \frac{\partial \mathcal{H}}{\partial \mathfrak{p}}, \quad \dot{\mathfrak{p}} = -\frac{\partial \mathcal{H}}{\partial \mathfrak{q}} + \mathbf{g}(\mathfrak{q})\mathbf{u}, \tag{10}$$

where $\mathbf{u} \in \mathbb{R}^n$ is an external affine generalized control input with a coefficient matrix $\mathbf{g}(\mathfrak{q}) \in \mathbb{R}^{n\times n}$ that only affects the generalized momenta $\mathfrak{p}$. Without any external control, i.e., $\mathbf{u} = \mathbf{0}$, the total energy of the system is conserved, $\frac{d}{dt}\mathcal{H}(\mathfrak{q}, \mathfrak{p}) = 0$.

The notion of energy in dynamical systems is shared across multiple domains, including mechanical, electrical, and thermal. A Port-Hamiltonian generalization [38] of Hamiltonian mechanics is used to model systems with energy-storing elements (e.g., kinetic and potential energy), energy-dissipating elements (e.g., friction or resistors), and external energy sources (e.g., control inputs), connected via energy ports. An input-state-output Port-Hamiltonian system is expressed in the following form:

$$\begin{bmatrix}\dot{\mathfrak{q}}\\ \dot{\mathfrak{p}}\end{bmatrix} = (\mathcal{J}(\mathfrak{q},\mathfrak{p}) - \mathcal{R}(\mathfrak{q},\mathfrak{p}))\begin{bmatrix}\frac{\partial \mathcal{H}}{\partial \mathfrak{q}}\\ \frac{\partial \mathcal{H}}{\partial \mathfrak{p}}\end{bmatrix} + \mathcal{G}(\mathfrak{q},\mathfrak{p})\mathbf{u}, \tag{11}$$

where $\mathcal{J}(\mathfrak{q},\mathfrak{p})$ is a skew-symmetric inter-connection matrix, representing the energy-storing elements, $\mathcal{R}(\mathfrak{q},\mathfrak{p})$ is a positive semi-definite dissipation matrix, representing the energy-dissipating elements, and $\mathcal{G}(\mathfrak{q},\mathfrak{p})$ is an input matrix such that $\mathcal{G}(\mathfrak{q},\mathfrak{p})\mathbf{u}$ represents the external energy sources. Intuitively, without the energy-dissipating elements and external energy sources, the skew-symmetry of $\mathcal{J}(\mathfrak{q},\mathfrak{p})$ guarantees the energy conservation of the system. The Hamiltonian dynamics (10) are a special case of the Port-Hamiltonian dynamics (11) with:

$$\mathcal{J}(\mathfrak{q},\mathfrak{p}) = \begin{bmatrix}\mathbf{0} & \mathbf{I}\\ -\mathbf{I} & \mathbf{0}\end{bmatrix},\ \mathcal{R}(\mathfrak{q},\mathfrak{p}) = \mathbf{0},\ \mathcal{G}(\mathfrak{q},\mathfrak{p}) = \begin{bmatrix}\mathbf{0}\\ \mathbf{g}(\mathfrak{q})\end{bmatrix}. \tag{12}$$

C. Symplectic Neural ODE Networks

The previous section described a system with Hamiltonian dynamics of the form $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathbf{u})$ in (10) with $\mathbf{x} = [\mathfrak{q}^\top \quad \mathfrak{p}^\top]^\top$. Now, we consider approximating the function $\mathbf{f}(\mathbf{x},\mathbf{u})$ when its elements, generalized mass $\mathbf{M}(\mathfrak{q})$, potential energy $V(\mathfrak{q})$, input matrix $\mathbf{g}(\mathfrak{q})$, are unknown. Chen et al. [4] proposed a neural ODE formulation to approximate the closed-loop dynamics $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\boldsymbol{\pi}(\mathbf{x}))$ for some unknown control policy $\mathbf{u} = \boldsymbol{\pi}(\mathbf{x})$ by a neural network $\bar{\mathbf{f}}_{\boldsymbol{\theta}}(\mathbf{x})$. The parameters of the network $\bar{\mathbf{f}}_{\boldsymbol{\theta}}(\mathbf{x})$ are trained using a data set $\mathcal{D} = \{t^{(i)}_{0:N}, \mathbf{x}^{(i)}_{0:N}\}_i$ of state trajectory samples $\mathbf{x}^{(i)}_n = \mathbf{x}^{(i)}(t^{(i)}_n)$ via forward and backward passes through a differentiable ODE solver. Given an initial state $\mathbf{x}^{(i)}_0$ at time $t^{(i)}_0$, a forward pass returns predicted states at times $t^{(i)}_1, \ldots, t^{(i)}_N$:

$$\{\bar{\mathbf{x}}^{(i)}_1, \ldots, \bar{\mathbf{x}}^{(i)}_N\} = \text{ODESolver}(\mathbf{x}^{(i)}_0, \bar{\mathbf{f}}_{\boldsymbol{\theta}}, t^{(i)}_1, \ldots, t^{(i)}_N). \tag{13}$$

The gradient of a loss function, $\sum_{i=1}^{D}\sum_{j=1}^{N}\ell(\mathbf{x}^{(i)}_j, \bar{\mathbf{x}}^{(i)}_j)$, is back-propagated by solving another ODE with adjoint states. The parameters $\boldsymbol{\theta}$ are updated by gradient descent to minimize the loss.

For physical systems, a symplectic neural ODE formulation, proposed by Zhong et al. [43], extends the neural ODE by integrating the structure of the Hamiltonian dynamics in (9) and (10) into the neural network model $\bar{\mathbf{f}}_{\boldsymbol{\theta}}(\mathbf{x})$. Three neural networks are used to approximate the three unknown functions: generalized mass inverse $\mathbf{M}_{\boldsymbol{\theta}_1}(\mathbf{q})^{-1}$, potential energy $V_{\boldsymbol{\theta}_2}(\mathbf{q})$, and input matrix $\mathbf{g}_{\boldsymbol{\theta}_3}(\mathbf{q})$, where $\boldsymbol{\theta}_1$, $\boldsymbol{\theta}_2$, $\boldsymbol{\theta}_3$ are the network parameters. A constant control input $\mathbf{u}$ is also considered, leading to the approximated dynamics:

$$\begin{bmatrix}\dot{\mathbf{x}}\\ \dot{\mathbf{u}}\end{bmatrix} = \begin{bmatrix}\bar{\mathbf{f}}_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{u})\\ \mathbf{0}\end{bmatrix} \tag{14}$$

where $\bar{\mathbf{f}}_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{u})$ has the form of (10).

V. Hamiltonian Dynamics on the $SE(3)$ Manifold

The symplectic neural ODE formulation [43] is designed for generalized coordinates $\mathfrak{q}$ and generalized momenta $\mathfrak{p}$, both in $\mathbb{R}^n$. For dynamical systems with states evolving on a manifold, such as $SE(3)$, it is necessary to enforce the manifold constraints (Sec. IV-A) on the Hamiltonian dynamics. In this section, we describe the Hamiltonian equations of motion on the $SE(3)$ manifold and reformulate the dynamics in the input-state-output Port-Hamiltonian form.

Let $\mathfrak{q} = [\mathbf{p}^\top \quad \mathbf{r}_1^\top \quad \mathbf{r}_2^\top \quad \mathbf{r}_3^\top]^\top \in \mathbb{R}^{12}$ be the generalized coordinates, and $\boldsymbol{\zeta} = [\mathbf{v}^\top \quad \boldsymbol{\omega}^\top]^\top \in \mathbb{R}^6$ be the generalized velocity. Note that $\mathfrak{q}$ and $\boldsymbol{\zeta}$ have different dimensions and satisfy the $SE(3)$ kinematics constraints in (5), re-expressed in vectorized form as:

$$\dot{\mathfrak{q}} = \mathfrak{q}^\times\boldsymbol{\zeta}, \qquad \mathfrak{q}^\times = \begin{bmatrix}\mathbf{R}^\top & \mathbf{0} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \hat{\mathbf{r}}_1^\top & \hat{\mathbf{r}}_2^\top & \hat{\mathbf{r}}_3^\top\end{bmatrix}^\top. \tag{15}$$

The Lagrangian function on $SE(3)$ is expressed in terms of $\mathfrak{q}$ and $\boldsymbol{\zeta}$, instead of $\mathfrak{q}$ and $\dot{\mathfrak{q}}$:

$$\mathcal{L}(\mathfrak{q},\boldsymbol{\zeta}) = \frac{1}{2}\boldsymbol{\zeta}^\top\mathbf{M}(\mathfrak{q})\boldsymbol{\zeta} - V(\mathfrak{q}). \tag{16}$$

The generalized mass matrix has a block-diagonal form when the body frame is attached to the center of mass [17]:

$$\mathbf{M}(\mathfrak{q}) = \begin{bmatrix}\mathbf{M}_1(\mathfrak{q}) & \mathbf{0}\\ \mathbf{0} & \mathbf{M}_2(\mathfrak{q})\end{bmatrix} \in \mathbb{S}^{6\times6}_{\succ 0}, \tag{17}$$

where $\mathbf{M}_1(\mathfrak{q}), \mathbf{M}_2(\mathfrak{q}) \in \mathbb{S}^{3\times3}_{\succ 0}$. The generalized momenta are defined, as before, via the partial derivative of the Lagrangian with respect to the velocity:

$$\mathfrak{p} = \begin{bmatrix}\mathfrak{p}_\mathbf{v}\\ \mathfrak{p}_{\boldsymbol{\omega}}\end{bmatrix} = \frac{\partial\mathcal{L}(\mathfrak{q},\boldsymbol{\zeta})}{\partial\boldsymbol{\zeta}} = \mathbf{M}(\mathfrak{q})\boldsymbol{\zeta} \in \mathbb{R}^6. \tag{18}$$

While the Hamiltonian function, $\mathcal{H}(\mathfrak{q},\mathfrak{p})$, is the same as (9), the system dynamics do not satisfy (10) due to the $SE(3)$ kinematics constraints in (15). However, the dynamics can be specified in a Port-Hamiltonian form [17, 12, 31] as in (11) with inter-connection matrix:

$$\mathcal{J}(\mathfrak{q},\mathfrak{p}) = \begin{bmatrix}\mathbf{0} & \mathfrak{q}^\times\\ -\mathfrak{q}^{\times\top} & \mathfrak{p}^\times\end{bmatrix}, \qquad \mathfrak{p}^\times = \begin{bmatrix}\mathbf{0} & \hat{\mathfrak{p}}_\mathbf{v}\\ \hat{\mathfrak{p}}_\mathbf{v} & \hat{\mathfrak{p}}_{\boldsymbol{\omega}}\end{bmatrix}, \tag{19}$$

and dissipation and input matrices:

$$\mathcal{R}(\mathfrak{q},\mathfrak{p}) = \mathbf{0}, \qquad \mathcal{G}(\mathfrak{q},\mathfrak{p}) = \begin{bmatrix}\mathbf{0}^\top & \mathbf{g}(\mathfrak{q})^\top\end{bmatrix}^\top. \tag{20}$$

Note that the dissipation matrix may not be necessarily zero and can, in fact, be used to model the effects of friction or drag forces [44]. However, for the clarity of the model learning approach and unified control design, we leave this for future work. We consider a system with unknown generalized mass $\mathbf{M}(\mathfrak{q})$, potential energy $V(\mathfrak{q})$, input matrix $\mathbf{g}(\mathfrak{q})$, and design a structured neural ODE network to learn these terms from state-control trajectories.

VI. Hamiltonian $SE(3)$ Dynamics Learning

This section describes a neural ODE network design incorporating $SE(3)$ kinematics and energy conservation constraints. We discuss data generation, embedding the constraints into the model architecture, and the training process.

A. Training Data Generation

We collect a data set $\mathcal{D} = \{t^{(i)}_{0:N}, \mathbf{x}^{(i)}_{0:N}, \mathbf{u}^{(i)}\}_{i=1}^{D}$ consisting of state sequences $\mathbf{x}^{(i)}_{0:N}$, where $\mathbf{x}^{(i)}_n = [\mathfrak{q}^{(i)\top}_n \quad \boldsymbol{\zeta}^{(i)\top}_n]^\top$, $\mathfrak{q}^{(i)}_n = [\mathbf{p}^{(i)\top}_n \quad \mathbf{r}^{(i)\top}_{1n} \quad \mathbf{r}^{(i)\top}_{2n} \quad \mathbf{r}^{(i)\top}_{3n}]^\top \in \mathbb{R}^{12}$, $\boldsymbol{\zeta}^{(i)}_n = [\mathbf{v}^{(i)\top}_n \quad \boldsymbol{\omega}^{(i)\top}_n]^\top \in \mathbb{R}^6$ for $n = 0, \dots, N$. Such data are generated by applying a constant control input $\mathbf{u}^{(i)}$ to the system and sampling the state $\mathbf{x}^{(i)}_n = \mathbf{x}^{(i)}(t^{(i)}_n)$ at times $t^{(i)}_n$ for $n = 0, \dots, N$. The generalized coordinates $\mathfrak{q}$ and velocity $\boldsymbol{\zeta}$ may be obtained from an odometry algorithm, such as [10, 25], or from a motion capture system. In physics-based simulation the data can be generated by applying random control inputs $\mathbf{u}^{(i)}$. In real-world applications, where safety is a concern, data may be collected by a human operator manually controlling the robot.

B. Model Design

To learn the dynamics $\mathbf{f}(\mathbf{x}, \mathbf{u})$ from the data set $\mathcal{D}$, we design a neural ODE network (Sec. IV-C), approximating the dynamics via a parametric function $\bar{\mathbf{f}}_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{u})$. To impose the $SE(3)$ Hamiltonian equations described in Sec. V on the structure of the neural network $\bar{\mathbf{f}}_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{u})$, we expand (11) with the interconnection matrix $\mathcal{J}$ in (19):

$$\dot{\mathfrak{p}} = \mathbf{R}\frac{\partial \mathcal{H}(\mathfrak{q}, \mathfrak{p})}{\partial \mathfrak{p}_{\mathbf{v}}}, \tag{21}$$

$$\dot{\mathbf{r}}_i = \mathbf{r}_i \times \frac{\partial \mathcal{H}(\mathfrak{q}, \mathfrak{p})}{\partial \mathfrak{p}_{\boldsymbol{\omega}}}, \quad i = 1, 2, 3 \tag{22}$$

$$\dot{\mathfrak{p}}_{\mathbf{v}} = \mathfrak{p}_{\mathbf{v}} \times \frac{\partial \mathcal{H}(\mathfrak{q}, \mathfrak{p})}{\partial \mathfrak{p}_{\boldsymbol{\omega}}} - \mathbf{R}^\top \frac{\partial \mathcal{H}(\mathfrak{q}, \mathfrak{p})}{\partial \mathbf{p}} + \mathbf{g}_{\mathbf{v}}(\mathfrak{q})\mathbf{u}, \tag{23}$$

$$\dot{\mathfrak{p}}_{\boldsymbol{\omega}} = \mathfrak{p}_{\boldsymbol{\omega}} \times \frac{\partial \mathcal{H}(\mathfrak{q}, \mathfrak{p})}{\partial \mathfrak{p}_{\boldsymbol{\omega}}} + \mathfrak{p}_{\mathbf{v}} \times \frac{\partial \mathcal{H}(\mathfrak{q}, \mathfrak{p})}{\partial \mathfrak{p}_{\mathbf{v}}} + + \sum_{i=1}^{3} \mathbf{r}_i \times \frac{\partial \mathcal{H}(\mathfrak{q}, \mathfrak{p})}{\partial \mathbf{r}_i} + \mathbf{g}_{\boldsymbol{\omega}}(\mathfrak{q})\mathbf{u}, \tag{24}$$

where the input matrix $\mathbf{g}(\mathfrak{q}) = \begin{bmatrix}\mathbf{g}_{\mathbf{v}}(\mathfrak{q})^\top & \mathbf{g}_{\boldsymbol{\omega}}(\mathfrak{q})^\top\end{bmatrix}^\top$ is decomposed into components corresponding to $\mathfrak{p}_{\mathbf{v}}$ and $\mathfrak{p}_{\boldsymbol{\omega}}$ and the Hamiltonian function $\mathcal{H}(\mathfrak{q}, \mathfrak{p})$ is defined in (9). Since the generalized momenta $\mathfrak{p}$ are not directly available in the data set $\mathcal{D}$ (Sec. VI-A), we use the time derivative of the generalized velocity, obtained from (18):

$$\dot{\boldsymbol{\zeta}} = \left(\frac{d}{dt}\mathbf{M}^{-1}(\mathfrak{q})\right)\mathfrak{p} + \mathbf{M}^{-1}(\mathfrak{q})\dot{\mathfrak{p}}. \tag{25}$$

The approximated dynamics function $\bar{\mathbf{f}}_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{u})$ is described by (21), (22), and (25) with an internal state $\mathfrak{p}$ satisfying the ODEs in (23) and (24).

To integrate the Hamiltonian equations into the structure of $\bar{\mathbf{f}}_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{u})$, we use four neural networks with parameters $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3, \boldsymbol{\theta}_4)$ to approximate the blocks $\mathbf{M}_1^{-1}(\mathfrak{q})$, $\mathbf{M}_2^{-1}(\mathfrak{q})$ of the inverse generalized mass in (17), the potential energy $V(\mathfrak{q})$, and the input matrix $\mathbf{g}(\mathfrak{q})$, respectively. The blocks $\mathbf{M}_1^{-1}(\mathfrak{q})$, $\mathbf{M}_2^{-1}(\mathfrak{q})$ are forced to be positive definite using Cholesky decomposition:

$$\begin{aligned}\mathbf{M}_1^{-1}(\mathfrak{q}) &= \mathbf{L}_1(\mathfrak{q})\mathbf{L}_1^\top(\mathfrak{q}) + \varepsilon\mathbf{I},\\ \mathbf{M}_2^{-1}(\mathfrak{q}) &= \mathbf{L}_2(\mathfrak{q})\mathbf{L}_2^\top(\mathfrak{q}) + \varepsilon\mathbf{I},\end{aligned} \tag{26}$$

where $\mathbf{L}_1(\mathfrak{q})$, $\mathbf{L}_2(\mathfrak{q})$ are lower-triangular matrices implemented as two neural networks with parameters $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$, respectively, and $\varepsilon > 0$. The time derivative $\frac{d}{dt}\mathbf{M}(\mathfrak{q})^{-1}$ and the partial derivatives $\frac{\partial \mathcal{H}}{\partial \mathfrak{p}}$ and $\frac{\partial \mathcal{H}}{\partial \mathfrak{q}}$ are calculated using automatic differentiation, e.g. by Pytorch.

In many applications, prior information is available about the generalized mass matrices $\mathbf{M}_1^{-1}(\mathfrak{q})$, $\mathbf{M}_2^{-1}(\mathfrak{q})$ and can be used to assist the training process. If a potentially inaccurate estimate of the masses is available, the neural networks modeling $\mathbf{L}_1(\mathfrak{q})$, $\mathbf{L}_2(\mathfrak{q})$ can be pre-trained to fit this estimate. For example, if we guess that the generalized mass matrix inverse is an identity matrix $\mathbf{I}$, we can sample M random inputs $\mathfrak{q}$ and pre-train the network to output $\mathbf{I}$. The pre-trained networks are used as initialization for the full training process which may correct these parameters using the dataset $\mathcal{D}$.

C. Training Process

Let $\bar{\mathbf{x}}^{(i)}(t) \in \mathbb{R}^{18}$ denote the state trajectory predicted with control input $\mathbf{u}^{(i)}$ by the approximate dynamics $\mathbf{f}_{\boldsymbol{\theta}}$ initialized at $\bar{\mathbf{x}}^{(i)}(t^{(i)}_0) = \mathbf{x}^{(i)}_0$. For sequence i, forward passes through the ODE solver in (13) return the predicted states $\bar{\mathbf{x}}^{(i)}_{0:N}$ at times $t^{(i)}_{0:N}$, where $\bar{\mathbf{x}}^{(i)}_n = [\bar{\mathfrak{q}}^{(i)\top}_n \quad \bar{\boldsymbol{\zeta}}^{(i)\top}_n]^\top$, $\bar{\mathfrak{q}}^{(i)\top}_n = [\bar{\mathbf{p}}^{(i)\top}_n \quad \bar{\mathbf{r}}^{(i)\top}_{1n} \quad \bar{\mathbf{r}}^{(i)\top}_{2n} \quad \bar{\mathbf{r}}^{(i)\top}_{3n}]^\top$, $\bar{\boldsymbol{\zeta}}^{(i)\top}_n = [\bar{\mathbf{v}}^{(i)\top}_n \quad \bar{\boldsymbol{\omega}}^{(i)\top}_n]^\top$, for $n = 1, \dots, N$. The predicted rotation matrix $\bar{\mathbf{R}}^{(i)}_n = [\bar{\mathbf{r}}^{(i)}_{1n} \quad \bar{\mathbf{r}}^{(i)}_{2n} \quad \bar{\mathbf{r}}^{(i)}_{3n}]^\top$ and the ground-truth one $\mathbf{R}^{(i)}_n = [\bar{\mathbf{r}}^{(i)}_{1n} \quad \bar{\mathbf{r}}^{(i)}_{2n} \quad \bar{\mathbf{r}}^{(i)}_{3n}]^\top$ are used to calculate an orientation loss:

$$\mathcal{L}_{\mathbf{R}}(\boldsymbol{\theta}) = \sum_{i=1}^{D}\sum_{n=1}^{N}\left\|\left(\log(\bar{\mathbf{R}}^{(i)}_n \mathbf{R}^{(i)\top}_n)\right)^\vee\right\|_2^2, \tag{27}$$

where $\log : SE(3) \mapsto \mathfrak{so}(3)$ converts a rotation matrix to a skew-symmetric matrix and $(\cdot)^\vee : \mathfrak{so}(3) \mapsto \mathbb{R}^3$ is the inverse of the hat map in (6), which extracts the components of a vector $\mathbf{w} \in \mathbb{R}^3$ from $\hat{\mathbf{w}} \in \mathfrak{so}(3)$. We use the squared Euclidean norm to calculate losses for the position and generalized velocity terms:

$$\begin{aligned}\mathcal{L}_{\mathbf{p}}(\boldsymbol{\theta}) &= \sum_{i=1}^{D}\sum_{n=1}^{N}\|\mathbf{p}^{(i)}_n - \bar{\mathbf{p}}^{(i)}_n\|_2^2,\\ \mathcal{L}_{\boldsymbol{\zeta}}(\boldsymbol{\theta}) &= \sum_{i=1}^{D}\sum_{n=1}^{N}\|\boldsymbol{\zeta}^{(i)}_n - \bar{\boldsymbol{\zeta}}^{(i)}_n\|_2^2.\end{aligned} \tag{28}$$

The total loss $\mathcal{L}(\boldsymbol{\theta})$ is defined as:

$$\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}_{\mathbf{R}}(\boldsymbol{\theta}) + \mathcal{L}_{\mathbf{p}}(\boldsymbol{\theta}) + \mathcal{L}_{\boldsymbol{\zeta}}(\boldsymbol{\theta}). \tag{29}$$

The gradient of the total loss function $\mathcal{L}(\boldsymbol{\theta})$ is back-propagated by solving an ODE with adjoint states [4]. Specifically, let $\mathbf{a} = \frac{\partial \mathcal{L}}{\partial \bar{\mathbf{x}}}$ be the adjoint state and $\mathbf{s} = (\bar{\mathbf{x}}, \mathbf{a}, \frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}})$ be

the augmented state. The augmented state dynamics are [4]:

$$\dot{\mathbf{s}} = \bar{\mathbf{f}}_{\mathbf{s}} = (\bar{\mathbf{f}}_{\boldsymbol{\theta}}, -\mathbf{a}^\top \frac{\partial \bar{\mathbf{f}}_{\boldsymbol{\theta}}}{\partial \bar{\mathbf{x}}}, -\mathbf{a}^\top \frac{\partial \bar{\mathbf{f}}_{\boldsymbol{\theta}}}{\partial \boldsymbol{\theta}}). \tag{30}$$

The predicted state $\bar{\mathbf{x}}$, the adjoint state $\mathbf{a}$, and the derivatives $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}}$ can be obtained by a single call to a reverse-time ODE solver starting from $\mathbf{s}_N = \mathbf{s}(t_N)$:

$$\mathbf{s}_0 = \left(\bar{\mathbf{x}}_0, \mathbf{a}_0, \frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}}\right) = \text{ODESolver}(\mathbf{s}_N, \bar{\mathbf{f}}_{\mathbf{s}}, t_N), \tag{31}$$

where at each time $t_k, k = 1, \ldots, N$, the adjoint state $\mathbf{a}_k$ at time t_k is reset to $\frac{\partial \mathcal{L}}{\partial \bar{\mathbf{x}}_k}$. The resulting derivative $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}}$ is used to update the parameters $\boldsymbol{\theta}$ using gradient descent.

VII. Energy-based Control Design

The function $\bar{\mathbf{f}}_{\boldsymbol{\theta}}$ learned in Sec. VI satisfies the input-state-output Port-Hamiltonian dynamics on the $SE(3)$ manifold in Sec. V by design. This section extends the interconnection and damping assignment passivity-based control (IDA-PBC) [38, 40, 36] to the $SE(3)$ manifold to achieve trajectory tracking (Problem 2) based on the learned $SE(3)$ Port-Hamiltonian dynamics.

First, consider a desired regulation point $\mathbf{x}^* = (\mathfrak{q}^*, \mathfrak{p}^*)$ that the system should be stabilized to. The Hamiltonian function $\mathcal{H}(\mathfrak{q}, \mathfrak{p})$, representing the total energy of the system, generally does not have a minimum at $\mathbf{x}^*$. An IDA-PBC controller is designed to inject additional energy $\mathcal{H}_a(\mathfrak{q}, \mathfrak{p})$ such that the desired total energy $\mathcal{H}_d(\mathfrak{q}, \mathfrak{p})$ achieves its minimum at $\mathbf{x}^*$:

$$\mathcal{H}_d(\mathfrak{q}, \mathfrak{p}) = \mathcal{H}(\mathfrak{q}, \mathfrak{p}) + \mathcal{H}_a(\mathfrak{q}, \mathfrak{p}). \tag{32}$$

To drive the system towards the desired state $\mathbf{x}^*$, the Port-Hamiltonian dynamics in (11) should be shaped into a desired form [38, 27, 40]:

$$\begin{bmatrix} \dot{\mathfrak{q}} \\ \dot{\mathfrak{p}} \end{bmatrix} = (\mathcal{J}_d(\mathfrak{q}, \mathfrak{p}) - \mathcal{R}_d(\mathfrak{q}, \mathfrak{p})) \begin{bmatrix} \frac{\partial \mathcal{H}_d}{\partial \mathfrak{q}} \\ \frac{\partial \mathcal{H}_d}{\partial \mathfrak{p}} \end{bmatrix}. \tag{33}$$

Specifically, the control input $\mathbf{u}$ should be chosen so that (11) and (33) are equal. This usual matching equation design does not directly apply to trajectory tracking problems, especially for under-actuated systems [36, 40].

Consider a desired state trajectory $\mathbf{x}^*(t) = (\mathfrak{q}^*(t), \mathfrak{p}^*(t))$ that the system should track. Let $\mathbf{x}_e = (\mathfrak{q}_e, \mathfrak{p}_e)$ denote the time-varying error state between the system state $\mathbf{x}$ and the desired trajectory $\mathbf{x}^*$. Let $\mathbf{R}_e = \mathbf{R}^{*\top}\mathbf{R} = \begin{bmatrix}\mathbf{r}_{e1} & \mathbf{r}_{e2} & \mathbf{r}_{e3}\end{bmatrix}^\top$ represent the rotation error between the current rotation matrix $\mathbf{R}$ and the desired one $\mathbf{R}^*$. The error $\mathfrak{q}_e$ in the generalized coordinates is:

$$\mathfrak{q}_e = \begin{bmatrix} \mathbf{R}^{*\top}(\mathbf{p} - \mathbf{p}^*) \\ \mathbf{r}_{e1} \\ \mathbf{r}_{e2} \\ \mathbf{r}_{e3} \end{bmatrix}. \tag{34}$$

The error in the generalized momenta is $\mathfrak{p}_e = \mathfrak{p} - \mathfrak{p}^*$. For under-actuated trajectory tracking, the desired total energy should be defined in terms of the error state as:

$$\mathcal{H}_d(\mathfrak{q}_e, \mathfrak{p}_e) = \frac{1}{2}\mathfrak{p}_e^\top \mathbf{M}_d^{-1}(\mathfrak{q}_e)\mathfrak{p}_e + V_d(\mathfrak{q}_e), \tag{35}$$

where $\mathbf{M}_d(\mathfrak{q}_e)$ and $V_d(\mathfrak{q}_e)$ are the desired generalized mass and potential energy. The new states $\mathfrak{q}_e$ and $\mathfrak{p}_e$ satisfy the desired dynamics:

$$\begin{bmatrix} \dot{\mathfrak{q}}_e \\ \dot{\mathfrak{p}}_e \end{bmatrix} = (\mathcal{J}_d(\mathfrak{q}_e, \mathfrak{p}_e) - \mathcal{R}_d(\mathfrak{q}_e, \mathfrak{p}_e)) \begin{bmatrix} \frac{\partial \mathcal{H}_d}{\partial \mathfrak{q}_e} \\ \frac{\partial \mathcal{H}_d}{\partial \mathfrak{p}_e} \end{bmatrix}, \tag{36}$$

leading to the following matching equations for the control input design:

$$\begin{aligned} \mathcal{G}(\mathfrak{q}, \mathfrak{p})\mathbf{u} = (\mathcal{J}_d(\mathfrak{q}_e, \mathfrak{p}_e) - \mathcal{R}_d(\mathfrak{q}_e, \mathfrak{p}_e)) \begin{bmatrix} \frac{\partial \mathcal{H}_d}{\partial \mathfrak{q}_e} \\ \frac{\partial \mathcal{H}_d}{\partial \mathfrak{p}_e} \end{bmatrix} \\ -(\mathcal{J}(\mathfrak{q}, \mathfrak{p}) - \mathcal{R}(\mathfrak{q}, \mathfrak{p})) \begin{bmatrix} \frac{\partial \mathcal{H}}{\partial \mathfrak{q}} \\ \frac{\partial \mathcal{H}}{\partial \mathfrak{p}} \end{bmatrix} + \begin{bmatrix} \dot{\mathfrak{q}} \\ \dot{\mathfrak{p}} \end{bmatrix} - \begin{bmatrix} \dot{\mathfrak{q}}_e \\ \dot{\mathfrak{p}}_e \end{bmatrix}. \end{aligned} \tag{37}$$

Choosing the following desired inter-connection matrix and dissipation matrix:

$$\mathcal{J}_d(\mathfrak{q}_e, \mathfrak{p}_e) = \begin{bmatrix} \mathbf{0} & \mathbf{J}_1 \\ -\mathbf{J}_1^\top & \mathbf{J}_2 \end{bmatrix}, \quad \mathcal{R}_d(\mathfrak{q}_e, \mathfrak{p}_e) = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{K_d} \end{bmatrix}, \tag{38}$$

and plugging $\mathcal{J}(\mathfrak{q}, \mathfrak{p})$ and $\mathcal{R}(\mathfrak{q}, \mathfrak{p})$ from (19) into the matching equations in (37), leads to:

$$\mathbf{0} = \mathbf{J}_1 \frac{\partial \mathcal{H}_d}{\partial \mathfrak{p}_e} - \mathfrak{q}^\times \frac{\partial \mathcal{H}}{\partial \mathfrak{p}} + \dot{\mathfrak{q}} - \dot{\mathfrak{q}}_e, \tag{39}$$

$$\begin{aligned} \mathbf{g}(\mathfrak{q})\mathbf{u} = \mathfrak{q}^{\times\top} \frac{\partial \mathcal{H}}{\partial \mathfrak{q}} - \mathbf{J}_1^\top \frac{\partial \mathcal{H}_d}{\partial \mathfrak{q}_e} + \mathbf{J}_2 \frac{\partial \mathcal{H}_d}{\partial \mathfrak{p}_e} - \mathfrak{p}^\times \frac{\partial \mathcal{H}}{\partial \mathfrak{p}} \\ -\mathbf{K_d} \frac{\partial \mathcal{H}_d}{\partial \mathfrak{p}_e} + \dot{\mathfrak{p}} - \dot{\mathfrak{p}}_e. \end{aligned} \tag{40}$$

Note that (39) is satisfied if we choose $\mathbf{J}_1 = \mathfrak{q}_e^\times$, i.e., $\dot{\mathfrak{q}}_e = \mathfrak{q}_e^\times \mathbf{M}_d(\mathfrak{q}_e)^{-1}\mathfrak{p}_e$. The desired control input can be obtained from (40) as the sum $\mathbf{u} = \mathbf{u}_{ES} + \mathbf{u}_{DI}$ of an energy-shaping component $\mathbf{u}_{ES}$ and a damping-injection component $\mathbf{u}_{DI}$:

$$\begin{aligned} \mathbf{u}_{ES} = \mathbf{g}^\dagger(\mathfrak{q}) \Bigg(\mathfrak{q}^{\times\top} \frac{\partial \mathcal{H}}{\partial \mathfrak{q}} - \mathfrak{q}_e^{\times\top} \frac{\partial \mathcal{H}_d}{\partial \mathfrak{q}_e} + \mathbf{J}_2 \frac{\partial \mathcal{H}_d}{\partial \mathfrak{p}_e} \\ -\mathfrak{p}^\times \frac{\partial \mathcal{H}}{\partial \mathfrak{p}} + \dot{\mathfrak{p}} - \dot{\mathfrak{p}}_e \Bigg), \end{aligned} \tag{41}$$

$$\mathbf{u}_{DI} = -\mathbf{g}^\dagger(\mathfrak{q})\mathbf{K_d} \frac{\partial \mathcal{H}_d}{\partial \mathfrak{p}_e}, \tag{42}$$

where $\mathbf{g}^\dagger(\mathfrak{q}) = \left(\mathbf{g}^\top(\mathfrak{q})\mathbf{g}(\mathfrak{q})\right)^{-1}\mathbf{g}^\top(\mathfrak{q})$ is the pseudo-inverse of $\mathbf{g}(\mathfrak{q})$. The control input $\mathbf{u}_{ES}$ exists as long as the desired $\mathbf{M}_d(\mathfrak{q}_e)$ and $\mathbf{J}_2$ satisfy the following matching condition:

$$\begin{aligned} \mathbf{g}^\perp(\mathfrak{q}) \Bigg(\mathfrak{q}^{\times\top} \frac{\partial \mathcal{H}}{\partial \mathfrak{q}} - \mathfrak{q}_e^{\times\top} \frac{\partial \mathcal{H}_d}{\partial \mathfrak{q}_e} + \mathbf{J}_2 \frac{\partial \mathcal{H}_d}{\partial \mathfrak{p}_e} \\ - \mathfrak{p}^\times \frac{\partial \mathcal{H}}{\partial \mathfrak{p}} + \dot{\mathfrak{p}} - \dot{\mathfrak{p}}_e \Bigg) = 0, \end{aligned} \tag{43}$$

where $\mathbf{g}^\perp(\mathfrak{q})$ is a maximal-rank left annihilator of $\mathbf{g}(\mathfrak{q})$, i.e., $\mathbf{g}^\perp(\mathfrak{q})\mathbf{g}(\mathfrak{q}) = \mathbf{0}$.

To avoid solving the PDE equations in (43) needed for the IDA-PBC controller, the parameters of the desired Port-Hamiltonian dynamics can be further specified to satisfy

$\mathbf{M}_d(\mathfrak{q}_e) = \mathbf{M}(\mathfrak{q})$ and $\mathbf{J}_2 \equiv 0$. With this choice, we add the following Hamiltonian energy term:

$$\mathcal{H}_a(\mathfrak{q},\mathfrak{p}) = -\mathcal{H}(\mathfrak{q},\mathfrak{p}) + \frac{1}{2}(\mathbf{p}-\mathbf{p}^*)^\top \mathbf{K_p}(\mathbf{p}-\mathbf{p}^*) + \frac{1}{2}\operatorname{tr}(\mathbf{K_R}(\mathbf{I}-\mathbf{R}^{*\top}\mathbf{R})) + \frac{1}{2}(\mathfrak{p}-\mathfrak{p}^*)^\top \mathbf{M}^{-1}(\mathfrak{q})(\mathfrak{p}-\mathfrak{p}^*) \tag{44}$$

to reshape the open-loop Hamiltonian $\mathcal{H}(\mathfrak{q},\mathfrak{p})$ into a desired total energy $\mathcal{H}_d(\mathfrak{q},\mathfrak{p}) = \mathcal{H}(\mathfrak{q},\mathfrak{p}) + \mathcal{H}_a(\mathfrak{q},\mathfrak{p})$ that is minimized along the desired trajectory $\mathbf{x}^* = (\mathfrak{q}^*, \mathfrak{p}^*)$. For an $SE(3)$ rigid-body system with constant generalized mass matrix $\mathbf{M}_d = \mathbf{M}$ and $\mathbf{J}_2 = 0$, the energy-shaping term in (41) and the damping-injection term in (42) simplify as follows:

$$\begin{aligned}\mathbf{u}_{ES}(\mathfrak{q},\mathfrak{p}) &= \mathbf{g}^\dagger(\mathfrak{q})\left(\mathfrak{q}^{\times\top}\frac{\partial V}{\partial \mathfrak{q}} - \mathfrak{p}^\times \mathbf{M}^{-1}\mathfrak{p} - \mathbf{e}(\mathfrak{q},\mathfrak{q}^*) + \dot{\mathfrak{p}}^*\right),\\ \mathbf{u}_{DI}(\mathfrak{q},\mathfrak{p}) &= -\mathbf{g}^\dagger(\mathfrak{q})\mathbf{K_d}\mathbf{M}^{-1}(\mathfrak{p}-\mathfrak{p}^*),\end{aligned} \tag{45}$$

where the generalized coordinate error between $\mathfrak{q}$ and $\mathfrak{q}^*$ is:

$$\mathbf{e}(\mathfrak{q},\mathfrak{q}^*) := \mathfrak{q}_e^{\times\top}\frac{\partial V_d}{\partial \mathfrak{q}_e} = \begin{bmatrix} \mathbf{R}^\top \mathbf{K_p}(\mathbf{p}-\mathbf{p}^*) \\ \frac{1}{2}\left(\mathbf{K_R}\mathbf{R}^{*\top}\mathbf{R} - \mathbf{R}^\top\mathbf{R}^*\mathbf{K_R}^\top\right)^\vee \end{bmatrix}. \tag{46}$$

Without requiring a priori knowledge of the system parameters, the control design in (45) and (46) offers a unified control approach for $SE(3)$ Hamiltonian systems that achieves trajectory tracking, if permissible by the system's degree of under-actuation.

VIII. Evaluation

We verify the effectiveness of our Hamiltonian-based neural ODE network for dynamics learning and control on the $SE(3)$ manifold using two fully-actuated systems (a pendulum and a rigid body) and one under-actuated system (a quadrotor). The implementation details for the experiments are provided in Appendix X-A.

A. Pendulum

We consider a pendulum with the following dynamics:

$$\ddot{\varphi} = -15\sin\varphi + 3u, \tag{47}$$

where φ is the angle of the pendulum with respect to its vertically downward position and u is a scalar control input. The ground-truth mass, potential energy, and the input coefficient are: $m = 1/3, V(\varphi) = 5(1-\cos\varphi)$, and $g(\varphi) = 1$, respectively. We collected data of the form $\{(\cos\varphi, \sin\varphi, \dot{\varphi})\}$ from an OpenAI Gym environment, provided by [43], with the dynamics in (47). To illustrate our manifold-constrained neural ODE learning, we viewed φ as a yaw angle and convert $(\cos\varphi, \sin\varphi)$ into a rotation matrix:

$$\mathbf{R} = \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{48}$$

We used $\boldsymbol{\omega} = [0, 0, \dot{\varphi}]$ for angular velocity and remove position $\mathbf{p}$ and linear velocity $\mathbf{v}$ from the Hamiltonian dynamics in (21), (22), (23), (24), restricting the system to the $SO(3)$ manifold with generalized coordinates $\mathfrak{q} = [\mathbf{r}_1^\top \quad \mathbf{r}_2^\top \quad \mathbf{r}_3^\top]^\top \in \mathbb{R}^9$.

As described in Sec. VI-A, control inputs $\mathbf{u}^{(i)}$ were sampled randomly and applied to the pendulum for five time intervals of $0.05s$, forming a dataset $\mathcal{D} = \left\{t_{0:N}^{(i)}, \mathfrak{q}_{0:N}^{(i)}, \boldsymbol{\omega}_{0:N}^{(i)}, \mathbf{u}^{(i)}\right)\Big\}_{i=1}^{D}$ with $N = 5$ and $D = 5120$. We trained the $SO(3)$ Hamiltonian neural ODE network as described in Sec. VI-C for 1000 iterations with no pre-training.

As noted in [43], since the generalized momenta $\mathfrak{p}$ are not available in the dataset, the dynamics of $\mathfrak{q}$ in (47) do not change if $\mathfrak{p}$ is scaled by a factor $\beta > 0$. This is also true in our formulation as scaling $\mathfrak{p}$ leaves the dynamics of $\mathfrak{q}$ in (22) and (25) unchanged. To emphasize this scale invariance, let $\mathbf{M}_\beta(\mathfrak{q}) = \beta\mathbf{M}(\mathfrak{q})$, $V_\beta(\mathfrak{q}) = \beta V(\mathfrak{q})$, $\mathbf{g}_\beta(\mathfrak{q}) = \beta\mathbf{g}(\mathfrak{q})$, and:

$$\begin{aligned}\mathfrak{p}_\beta &= \mathbf{M}_\beta(\mathfrak{q})\boldsymbol{\omega} = \beta\mathfrak{p}, \qquad \dot{\mathfrak{p}}_\beta = \beta\dot{\mathfrak{p}},\\ \mathcal{H}_\beta(\mathfrak{q},\mathfrak{p}) &= \frac{1}{2}\mathfrak{p}_\beta^\top \mathbf{M}_\beta^{-1}(\mathfrak{q})\mathfrak{p}_\beta + V_\beta(\mathfrak{q}) = \beta\mathcal{H}(\mathfrak{q},\mathfrak{p}),\\ \frac{\partial \mathcal{H}_\beta(\mathfrak{q},\mathfrak{p})}{\partial \mathfrak{p}_\beta} &= \mathbf{M}_\beta^{-1}(\mathfrak{q})\mathfrak{p}_\beta = \frac{\partial H(\mathfrak{q},\mathfrak{p})}{\partial \mathfrak{p}},\end{aligned} \tag{49}$$

guaranteeing that (21), (22), and (25) still hold.

Fig. 1 shows the training and testing behavior of our $SO(3)$ Hamiltonian ODE network. Fig. 1b and 1d show that the $\left[\mathbf{M}(\mathfrak{q})^{-1}\right]_{3,3}$ entry of the mass inverse and the $[\mathbf{g}(\mathfrak{q})]_3$ entry of the input matrix are close to their correct values of 3 and 1, respectively, while the other entries are close to zero. Fig. 1c indicates a constant gap between the learned and the ground-truth potential energy, which can be explained by the relativity of potential energy. The learned pendulum dynamics are illustrated by the phase portrait of a test trajectory in Fig. 1e, which coincides with the ground-truth portrait.

We tested stabilization of the pendulum based on the learned dynamics to the stable equilibrium at the downward position $\varphi = 0$ and to the unstable equilibrium at the upward position $\varphi = \pi$. Since the pendulum is a fully-actuated system and the desired state has zero velocity, potential energy shaping is enough to drive the system to the desired state $(\mathfrak{q}^*, \mathbf{0})$. Our energy-based controller in Sec. VII achieved this by setting $\mathbf{J}_2 = \mathfrak{p}^\times$, $\mathbf{M}_d(\mathfrak{q}) = \mathbf{M}(\mathfrak{q})$ in (41), leading to:

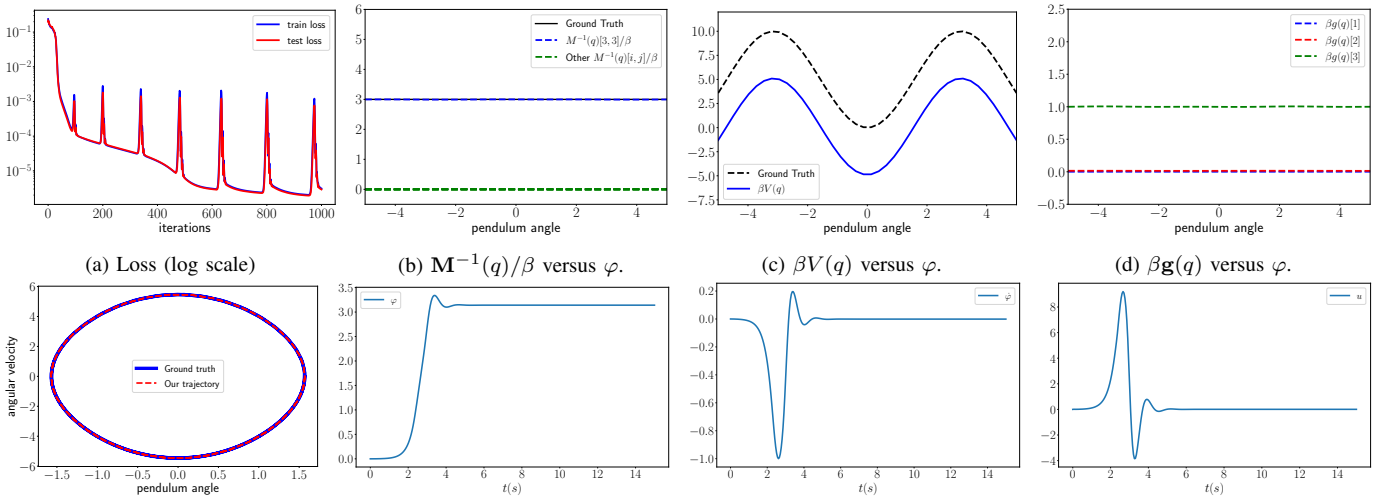
$$\mathbf{u} = \mathbf{u}_{ES} + \mathbf{u}_{DI} = \mathbf{g}_{\boldsymbol{\omega}}^\dagger(\mathfrak{q})\sum_{i=1}^{3}\mathbf{r}_i \times \frac{\partial \mathcal{H}_a}{\partial \mathbf{r_i}} - \mathbf{g}_{\boldsymbol{\omega}}^\dagger \mathbf{K_d}(\mathfrak{q})\boldsymbol{\omega}, \tag{50}$$

where the additional energy $\mathcal{H}_a(\mathfrak{q},\mathfrak{p})$ was simplified by removing the position error:

$$\mathcal{H}_a(\mathfrak{q},\mathfrak{p}) = -V(\mathfrak{q}) + \frac{1}{2}\operatorname{tr}(\mathbf{K_R}(\mathbf{I}-\mathbf{R}^{*\top}\mathbf{R})). \tag{51}$$

The controlled angle φ and angular velocity $\dot{\varphi}$ as well as the control inputs $\mathbf{u}$ with gains $\mathbf{K_R} = \mathbf{I}$ and $\mathbf{K_d} = 0.4\mathbf{I}$ are shown over time in Fig. 1f, 1g, and 1h. We can see that the controller was able to smoothly drive the pendulum from $\phi = 0$ to $\phi = \pi$, relying only on the learned dynamics.

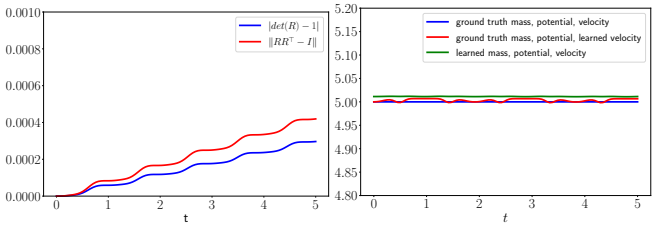
Lastly, Fig. 2a verifies that the $SO(3)$ constraints are satisfied by plotting $|\det\mathbf{R} - 1|$ and $\|\mathbf{R}\mathbf{R}^\top - \mathbf{I}\|$ from the learned model along a 5-second trajectory rollout with zero input, initialized at $\phi = \pi/2$. To verify the energy conservation as well, we calculated the Hamiltonian via (9) along the trajectory



(a) Loss (log scale) (b) $\mathbf{M}^{-1}(q)/\beta$ versus φ. (c) $\beta V(q)$ versus φ. (d) $\beta \mathbf{g}(q)$ versus φ.

(e) Portrait of a test trajectory. (f) Controlled angle φ. (g) Controlled angular velocity $\dot{\varphi}$. (h) Control input u.

Fig. 1: Evaluation of our $SO(3)$ Hamiltonian neural ODE network on a pendulum system with scale factor $\beta = 4.6$.



(a) SO(3) constraints. (b) Total energy.

Fig. 2: $SO(3)$ constraints and total energy along a trajectory rollout from the learned pendulum model, initialized at $\phi = \pi/2$.

using: (1) ground-truth mass, potential energy, and velocity; (2) ground-truth mass and potential energy but velocity rolled out from the learned dynamics; and (3) all mass, potential energy and velocity rolled out from the learned dynamics. The constant Hamiltonian in Fig. 2b verifies that, with no control input, our model obeys the law of energy conservation and remains close to the ground-truth energy after scaling by β.

B. Fully-actuated Rigid Body

Next, we consider a fully-actuated rigid body with mass $m = 0.027$ and inertia matrix $\mathbf{J} = 10^{-5}\text{diag}([1.4, 1.4, 2.17])$. This can be viewed as an abstraction of any mobile robot that can be modeled as a single rigid body, such as car-like, fixed-wing, and quadrotor robots. For example, a hexarotor UAV with fixed-tilt rotors pointing in different directions is fully actuated [30]. The ground-truth dynamics follow (21), (22), (23), and (24) with generalized coordinates $\mathfrak{q} = [\mathbf{p}^\top \ \mathbf{r}_1^\top \ \mathbf{r}_2^\top \ \mathbf{r}_3^\top]^\top \in \mathbb{R}^{12}$, generalized velocity $\boldsymbol{\zeta} = [\mathbf{v}^\top \ \boldsymbol{\omega}^\top]^\top \in \mathbb{R}^6$, generalized mass $\mathbf{M}_1(\mathfrak{q}) = m\mathbf{I}$, $\mathbf{M}_2(\mathfrak{q}) = \mathbf{J}$, potential energy $V(\mathfrak{q}) = mgz$, and the input matrix $\mathbf{g}(\mathfrak{q}) = \mathbf{I}$. The control input $\mathbf{u}$ is a 6-dimensional wrench (i.e., 3-dimensional force and 3-dimensional torque).

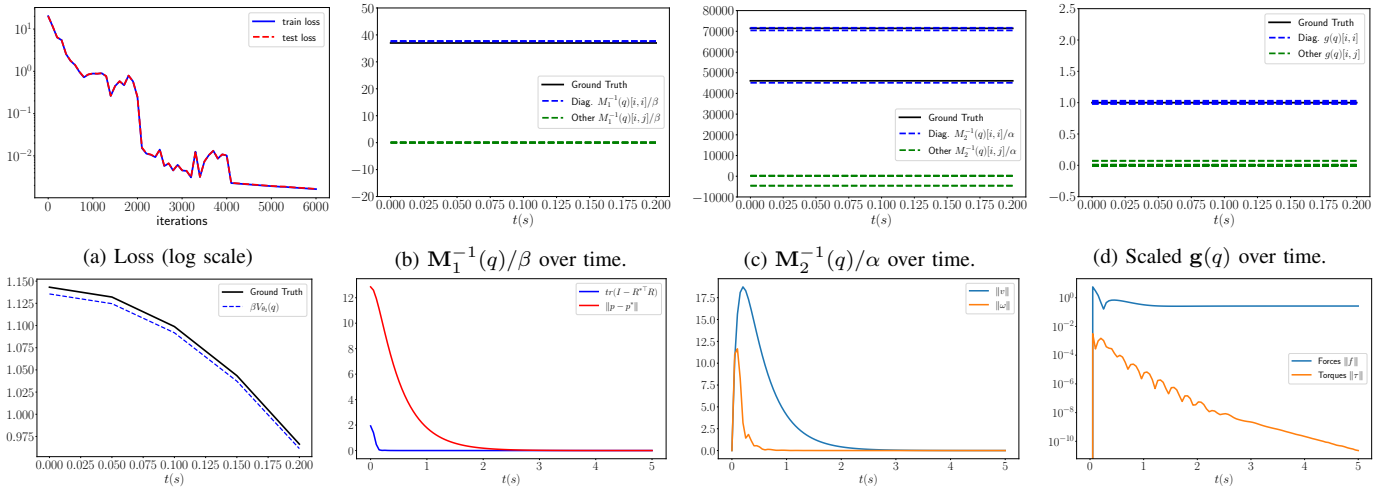
As described in Sec. VI-A, control inputs $\mathbf{u}^{(i)}$ were sampled randomly and applied to the system for one time step of $0.05s$, forming a dataset $\mathcal{D} = \left\{t_{0:N}^{(i)}, \mathfrak{q}_{0:N}^{(i)}, \boldsymbol{\zeta}_{0:N}^{(i)}, \mathbf{u}^{(i)})\right\}_{i=1}^{D}$ with $N = 1$ and $D = 11520$. The $SE(3)$ ODE network was trained as described in Sec. VI-C for 6000 iterations. The neural networks modeling $\mathbf{M}_1(\mathfrak{q})$ and $\mathbf{M}_2(\mathfrak{q})$ were pre-trained to output an identity matrix.

Similar to the pendulum system in Sec. VIII-A, the dynamics of $\mathfrak{q}$ do not change if $\mathfrak{p}$ is scaled. In fact, as the ground-truth generalized mass is $\mathbf{M}_1(\mathfrak{q}) = m\mathbf{I}$, and the potential energy $V(\mathfrak{q}) = mgz$ only depends on the position $\mathbf{p}$, the generalized momenta $\mathfrak{p}_{\boldsymbol{\omega}}$ and $\mathfrak{p}_{\mathbf{v}}$ can be scaled, respectively, by two different factors $\alpha > 0$ and $\beta > 0$. In other words, $\mathbf{M}_1(\mathfrak{q})$, $V(\mathfrak{q})$, and $\mathbf{g}_{\mathbf{v}}(\mathfrak{q})$ can be scaled by β and $\mathbf{M}_2(\mathfrak{q})$ and $\mathbf{g}_{\boldsymbol{\omega}}(\mathfrak{q})$ can be scaled by α without changing the dynamics of $\mathfrak{q}$ and $\boldsymbol{\zeta}$ in (21), (22), and (25).

Fig. 3 shows the training and testing performance of the $SE(3)$ Hamiltonian ODE network. The training loss is shown in Fig. 3a in log scale. Fig. (3b), 3c, and 3d show that the diagonal entries of the scaled mass matrices $\mathbf{M}_1(\mathfrak{q})$, $\mathbf{M}_2(\mathfrak{q})$ and the input matrix $\mathbf{g}(\mathfrak{q})$ converge to the ground-truth values while the remaining entries are close to $\mathbf{0}$. Fig. 3e shows that the learned potential energy closely resembles the ground-truth values up to scaling.

We tested regulation of the fully-actuated rigid body based on its learned dynamics from an initial position $\mathbf{p} = \mathbf{0}$ and randomized initial orientation to a desired position $\mathbf{p}^* = [1, 2, 5]$ and desired orientation $\mathbf{R}^* = \mathbf{I}$. As for the pendulum, potential energy shaping was enough to drive the system to a desired state $(\mathfrak{q}^*, 0)$ and was achieved by setting $\mathbf{J}_2 = \mathfrak{p}^\times$, $\mathbf{M}_d(\mathfrak{q}) = \mathbf{M}(\mathfrak{q})$ in (41). The additional energy $\mathcal{H}_a(\mathfrak{q}, \mathfrak{p})$ was simplified to:

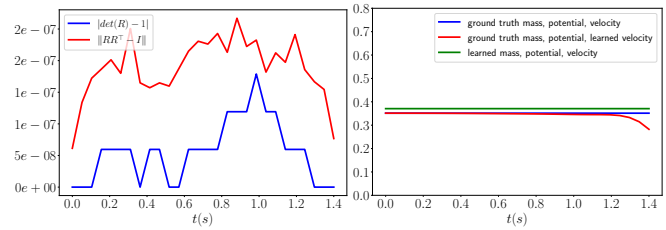
$$\begin{aligned}\mathcal{H}_a(\mathfrak{q}, \mathfrak{p}) \quad &= \quad -V(\mathfrak{q}) + \frac{1}{2}(\mathbf{p} - \mathbf{p}^*)^\top \mathbf{K}_{\mathbf{p}}(\mathbf{p} - \mathbf{p}^*) \\ &\quad + \frac{1}{2}\,\text{tr}(\mathbf{K}_{\mathbf{R}}(\mathbf{I} - \mathbf{R}^{*\top}\mathbf{R})). \end{aligned} \tag{52}$$



(a) Loss (log scale) (b) $\mathbf{M}_1^{-1}(q)/\beta$ over time. (c) $\mathbf{M}_2^{-1}(q)/\alpha$ over time. (d) Scaled $\mathbf{g}(q)$ over time.

(e) $\beta V(q)$ over time. (f) Controlled state errors. (g) Controlled velocity. (h) Control input $\mathbf{u}$.

Fig. 3: Evaluation of our $SE(3)$ Hamiltonian neural ODE network on a fully-actuated rigid body with scaling $\alpha = 0.0032$ and $\beta = 0.864$.



(a) SO(3) constraints. (b) Total energy.

Fig. 4: $SO(3)$ constraints and total energy along a trajectory rollout with zero control input from the learned rigid body model.

The controller becomes $\mathbf{u} = \mathbf{u}_{ES} + \mathbf{u}_{DI}$ where:

$$\mathbf{u}_{ES}(\mathfrak{q}, \mathfrak{p}) = \mathbf{g}_{\boldsymbol{\theta}_3}^{\dagger}(\mathfrak{q}) \begin{bmatrix} \sum_{i=1}^{3} \mathbf{r}_i \times \frac{\partial \mathcal{H}_a}{\partial \mathbf{r}_i} \\ -\mathbf{R}^\top \frac{\partial \mathcal{H}_a}{\partial \mathbf{p}} \end{bmatrix}, \quad (53)$$

$$\mathbf{u}_{DI}(\mathfrak{q}, \mathfrak{p}) = -\mathbf{g}_{\boldsymbol{\theta}_3}^{\dagger}(\mathfrak{q}) \mathbf{K}_\mathbf{d} \boldsymbol{\zeta}. \quad (54)$$

With control gains $\mathbf{K_p} = \mathbf{K_R} = 0.5\mathbf{I}$ and $\mathbf{K_d} = 0.25\text{diag}(10^{-3}, 10^{-3}, 10^{-3}, 1.0, 1.0, 1.0)$, the errors $\text{tr}(\mathbf{I} - \mathbf{R}^{*\top}\mathbf{R})$ and $\|\mathbf{p} - \mathbf{p}^*\|$ and the velocities $\mathbf{v}$, $\boldsymbol{\omega}$ go to 0 and regulation is achieved successfully, as shown in Fig. 3f and 3g. Fig. 3h shows the control input $\mathbf{u}$, in which the torque component goes to $\mathbf{0}$ while the force component approaches a constant value compensating for gravity.

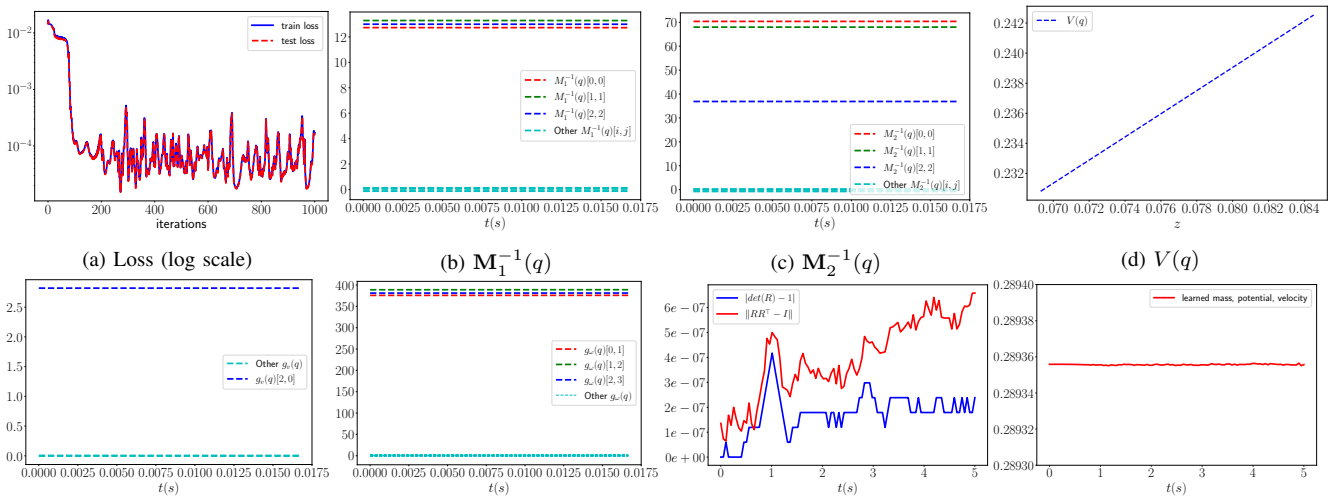
As for the pendulum, we also verified that predicted orientation trajectories from the learned model satisfy the $SO(3)$ constraints. Fig. 4a shows $|\det \mathbf{R} - 1|$ and $\|\mathbf{R}\mathbf{R}^\top - \mathbf{I}\|$ along a trajectory rollout. As before, we also calculated the Hamiltonian via (9) along the trajectory with no control input using: 1) ground-truth mass, potential energy and velocity; 2) ground-truth mass and potential energy but velocity rolled out from the learned dynamics; and 3) all mass, potential energy and velocity rolled out from the learned dynamics. Fig. 2b shows that the Hamiltonian stays constant for cases 1) and 3) but not for case 2) after 25 time steps. This indicates that the learned dynamics, by design, are guaranteed to obey the law of energy conservation with respect to the learned mass and potential energy, rather than the ground-truth ones. This discrepancy can be fixed by increasing the length of the training sequence N (in this experiment, $N = 1$) or by avoiding rollout predictions of more than 25 steps with the model without additional training.

C. Crazyflie Quadrotor

Finally, we demonstrate that our $SE(3)$ dynamics learning and control approach can achieve trajectory tracking for an under-actuated system. We consider a Crazyflie quadrotor, shown in Fig. 7a, simulated in the physics-based simulator PyBullet [28]. The control input $\mathbf{u} = [f, \boldsymbol{\tau}]$ includes a thrust $f \in \mathbb{R}_{\geq 0}$ and a torque vector $\boldsymbol{\tau} \in \mathbb{R}^3$ generated by the 4 rotors. The generalized coordinates and velocity are $\mathfrak{q} = [\mathbf{p}^\top \quad \mathbf{r}_1^\top \quad \mathbf{r}_2^\top \quad \mathbf{r}_3^\top]^\top \in \mathbb{R}^{12}$ and $\boldsymbol{\zeta} = [\mathbf{v}^\top \quad \boldsymbol{\omega}^\top]^\top \in \mathbb{R}^6$ as before.

The quadrotor was controlled from a random starting point to 18 different desired poses using a PID controller provided by [28], providing 18 2.5-second trajectories. The trajectories were used to generate a dataset $\mathcal{D} = \{t_{0:N}^{(i)}, \mathfrak{q}_{0:N}^{(i)}, \boldsymbol{\zeta}_{0:N}^{(i)}, \mathbf{u}^{(i)})\}_{i=1}^{D}$ with $N = 5$ and $D = 1080$. The $SE(3)$ Hamiltonian ODE network was trained, as described in Sec. VI-C, for 1000 iterations with $\mathbf{M}_1(\mathfrak{q})$ and $\mathbf{M}_2(\mathfrak{q})$ pre-trained to output an identity matrix.

Our training and test results are shown in Fig. 5. The learned generalized mass and inertia converged to constant diagonal matrices: $\mathbf{M}_1^{-1}(\mathfrak{q}) \approx 12.8\mathbf{I}$, $\mathbf{M}_2^{-1}(\mathfrak{q}) \approx \text{diag}([70, 70, 36])$. The input matrix $\mathbf{g}_\mathbf{v}(\mathfrak{q})$ converged to a constant matrix whose entry $\left[\mathbf{g}_\mathbf{v}(\mathfrak{q})\right]_{2,0} \approx 2.8$ while other entries were closed to 0, consistent with the fact that the thrust only affects the linear velocity along the z axis in the body-fixed frame. The input matrix $\mathbf{g}_{\boldsymbol{\omega}}(\mathfrak{q})$ converged to $\sim 380\mathbf{I}$ as the torques affects all components of the angular velocity $\boldsymbol{\omega}$. The potential energy $V(\mathfrak{q})$ was linear in the height z, agreed with the gravitational



(a) Loss (log scale) (b) $\mathbf{M}_1^{-1}(q)$ (c) $\mathbf{M}_2^{-1}(q)$ (d) $V(q)$

(e) $\mathbf{g}_\mathbf{v}(q)$ (f) $\mathbf{g}_{\boldsymbol{\omega}}(q)$ (g) SO(3) constraints. (h) Total energy.

Fig. 5: Evaluation of the $SE(3)$ Hamiltonian neural ODE network on an under-actuated Crazyflie quadrotor in the PyBullet simulator [28].

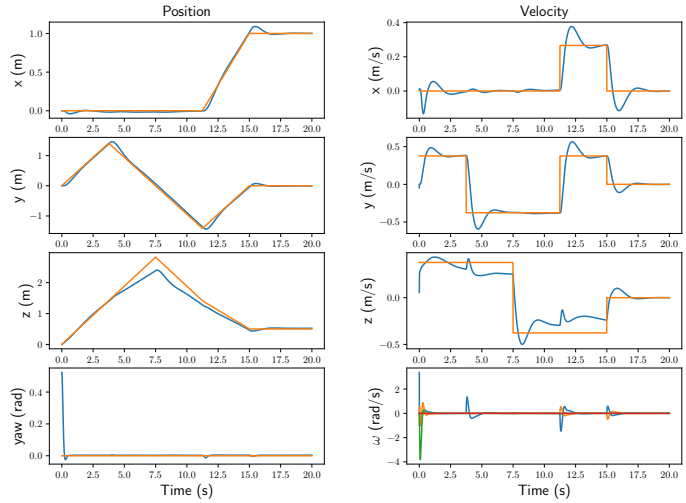
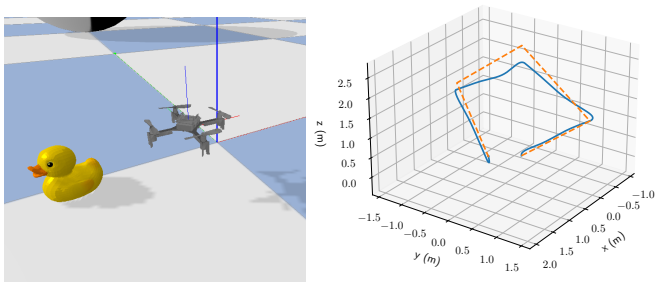


Fig. 6: Crazyflie quadrotor trajectory (blue) tracking a desired diamond-shaped trajectory (orange) shown in Fig. 7.



(a) Crazyflie simulator (b) Trajectory tracking

Fig. 7: Trajectory tracking experiment with a Crazyflie quadrotor in the PyBullet simulator [28].

potential.

We verified that predicted orientation trajectories from the learned model satisfy the $SO(3)$ constraints. Fig. 5g plots two near-zero quantities $|\det \mathbf{R} - 1|$ and $\|\mathbf{R}\mathbf{R}^\top - \mathbf{I}\|$, by rolling out our learned dynamics for 5 seconds. We also calculated the Hamiltonian via (9) using the learned generalized mass matrix and the velocity rolled out from the learned dynamics. Fig. 5h shows a constant total energy along the 5-second trajectory without control input, obeying the law of energy conservation.

Finally, we verified our energy-based controller for under-actuated systems in Sec. VII by driving the drone to track a pre-defined trajectory. We are given the desired position $\mathbf{p}^*$ and the desired heading ψ^* by the trajectory and construct an appropriate choice of $\mathbf{R}^*$, $\mathfrak{p}^*$ to be used with the energy-based controller in (45). The desired momenta are constructed as follows:

$$\begin{aligned}\mathfrak{p}^* &= \mathbf{M}\begin{bmatrix}\mathbf{R}^\top\dot{\mathbf{p}}^* \\ \mathbf{R}^\top\mathbf{R}^*\boldsymbol{\omega}^*\end{bmatrix}, \\ \dot{\mathfrak{p}}^* &= \mathbf{M}\begin{bmatrix}\mathbf{R}^\top\ddot{\mathbf{p}}^* - \hat{\boldsymbol{\omega}}\mathbf{R}^\top\dot{\mathbf{p}}^* \\ \mathbf{R}^\top\mathbf{R}^*\dot{\boldsymbol{\omega}}^* - \hat{\boldsymbol{\omega}}\mathbf{R}^\top\mathbf{R}^*\boldsymbol{\omega}^*\end{bmatrix}.\end{aligned} \tag{55}$$

The control input (45) becomes:

$$\begin{aligned}\mathbf{u} \;=\; \mathbf{g}^\dagger(\mathfrak{q})\Big(&\mathfrak{q}^{\times\top}\frac{\partial V(\mathfrak{q})}{\partial \mathfrak{q}} - \mathfrak{p}^\times\mathbf{M}^{-1}\mathfrak{p} - \mathbf{e}(\mathfrak{q},\mathfrak{q}^*) \\ &- \mathbf{K_d}\mathbf{M}^{-1}(\mathfrak{p}-\mathfrak{p}^*) + \dot{\mathfrak{p}}^*\Big).\end{aligned} \tag{56}$$

By expanding the terms in (56), we have:

$$\mathfrak{p}^\times\mathbf{M}^{-1}\mathfrak{p} \;=\; \mathfrak{p}^\times\boldsymbol{\zeta} = \begin{bmatrix}\hat{\mathfrak{p}}_\mathbf{v}\boldsymbol{\omega} \\ \hat{\mathfrak{p}}_{\boldsymbol{\omega}}\boldsymbol{\omega} + \hat{\mathfrak{p}}_\mathbf{v}\mathbf{v}\end{bmatrix}, \tag{57}$$

$$\mathbf{M}^{-1}(\mathfrak{p}-\mathfrak{p}^*) \;=\; \begin{bmatrix}\mathbf{v} - \mathbf{R}^\top\dot{\mathbf{p}}^* \\ \boldsymbol{\omega} - \mathbf{R}^\top\mathbf{R}^*\boldsymbol{\omega}^*\end{bmatrix}, \tag{58}$$

$$\mathfrak{q}^{\times\top}\frac{\partial V(\mathfrak{q})}{\partial \mathfrak{q}} \;=\; \begin{bmatrix}\mathbf{R}^\top\frac{\partial V(\mathfrak{q})}{\partial \mathbf{p}} \\ \sum_{i=1}^3 \hat{\mathbf{r}}_i\frac{\partial V(\mathfrak{q})}{\partial \mathbf{r}_i}\end{bmatrix}. \tag{59}$$

Choosing the control gain $\mathbf{K_d}$ of the form $\mathbf{K_d} = \begin{bmatrix} \mathbf{K}_v & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_\omega \end{bmatrix}$, the control input can be written explicitly as

$$\mathbf{u} = \mathbf{g}^\dagger(\mathfrak{q}) \begin{bmatrix} \mathbf{b_v} \\ \mathbf{b}_{\boldsymbol{\omega}} \end{bmatrix}, \tag{60}$$

where

$$\begin{aligned} \mathbf{b_v} &= \mathbf{R}^\top \frac{\partial V(\mathfrak{q})}{\partial \mathbf{p}} - \hat{\mathfrak{p}}_\mathbf{v}\boldsymbol{\omega} - \mathbf{R}^\top \mathbf{K_p}(\mathbf{p} - \mathbf{p}^*) \\ &\quad -\mathbf{K_v}(\mathbf{v} - \mathbf{R}^\top \dot{\mathbf{p}}^*) + \mathbf{M}_1(\mathbf{R}^\top \ddot{\mathbf{p}}^* - \hat{\boldsymbol{\omega}}\mathbf{R}^\top \dot{\mathbf{p}}^*), \quad (61) \\ \mathbf{b}_{\boldsymbol{\omega}} &= \sum_{i=1}^{3} \hat{\mathbf{r}}_i \frac{\partial V(\mathfrak{q})}{\partial \mathbf{r}_i} - \mathbf{K}_{\boldsymbol{\omega}}(\boldsymbol{\omega} - \mathbf{R}^\top \mathbf{R}^* \boldsymbol{\omega}^*) \\ &\quad -(\hat{\mathfrak{p}}_{\boldsymbol{\omega}}\boldsymbol{\omega} + \hat{\mathfrak{p}}_\mathbf{v}\mathbf{v}) - \frac{1}{2}\left(\mathbf{K_R}\mathbf{R}^{*\top}\mathbf{R} - \mathbf{R}^\top \mathbf{R}^* \mathbf{K_R}^\top\right)^\vee \\ &\quad +\mathbf{M}_2(\mathbf{R}^\top \mathbf{R}^* \dot{\boldsymbol{\omega}}^* - \hat{\boldsymbol{\omega}}\mathbf{R}^\top \mathbf{R}^* \boldsymbol{\omega}^*). \quad (62) \end{aligned}$$

Note that $\mathbf{b_v} \in \mathbb{R}^3$ is the desired thrust in the body frame that depend only on the desired position $\mathbf{p}^*$ and the current pose. It is transformed to the world frame as $\mathbf{R}\mathbf{b_v}$, representing the thrust in the world frame. Inspired by [16], the vector $\mathbf{R}\mathbf{b_v}$ should be the z axis of the body frame, i.e., the third column $\mathbf{b}_3^*$ of the desired rotation matrix $\mathbf{R}^*$. The second column $\mathbf{b}_2^*$ of the desired rotation matrix $\mathbf{R}^*$ can be chosen so that it has the desired yaw angle ψ^* and is perpendicular to $\mathbf{b}_3^*$. This can be done by projecting the second column of the yaw's rotation matrix $\mathbf{b}_2^\psi = [-\sin\psi, \cos\psi, 0]$ onto the plane perpendicular to $\mathbf{b}_3^*$. We have $\mathbf{R}^* = [\mathbf{b}_1^* \quad \mathbf{b}_2^* \quad \mathbf{b}_3^*]$ where:

$$\mathbf{b}_3^* = \frac{\mathbf{R}\mathbf{b_v}}{\|\mathbf{R}\mathbf{b_v}\|}, \mathbf{b}_1^* = \frac{\mathbf{b}_2^\psi \times \mathbf{b}_3^*}{\|\mathbf{b}_2^\psi \times \mathbf{b}_3^*\|}, \mathbf{b}_2^* = \mathbf{b}_3^* \times \mathbf{b}_1^*, \tag{63}$$

and $\hat{\boldsymbol{\omega}}^* = \mathbf{R}^{*\top}\dot{\mathbf{R}}^*$. The derivative $\dot{\mathbf{R}}^*$ is calculated as follows.

$$\begin{aligned} \dot{\mathbf{b}}_3^* &= \mathbf{b}_3^* \times \frac{\mathbf{R}\dot{\mathbf{b}}_\mathbf{v}}{\|\mathbf{R}\mathbf{b_v}\|} \times \mathbf{b}_3^*, \quad (64) \\ \dot{\mathbf{b}}_1^* &= \mathbf{b}_1^* \times \frac{\dot{\mathbf{b}}_2^\psi \times \mathbf{b}_3^* + \mathbf{b}_2^\psi \times \dot{\mathbf{b}}_3^*}{\|\mathbf{b}_2^\psi \times \mathbf{b}_3^*\|} \times \mathbf{b}_1^*, \quad (65) \\ \dot{\mathbf{b}}_2^* &= \dot{\mathbf{b}}_3^* \times \mathbf{b}_1^* + \mathbf{b}_3^* \times \dot{\mathbf{b}}_1^*. \quad (66) \end{aligned}$$

By plugging $\mathbf{R}^*$ and $\boldsymbol{\omega}^*$ back in $\mathbf{b}_{\boldsymbol{\omega}}$, we obtain the complete control input $\mathbf{u}$ in (60).

Fig. 7b qualitatively shows that the drone controlled by our energy-based controller successfully finished the task. Since the learned generalized mass $\mathbf{M}_1$ and inertia $\mathbf{M}_2$ converged to constant diagonal matrices, the control gains were chosen as follows: $\mathbf{K_p} = 5\mathbf{M}_1, \mathbf{K_v} = 2.5\mathbf{M}_1, \mathbf{K_R} = 250\mathbf{M}_2, \mathbf{K}_{\boldsymbol{\omega}} = 20\mathbf{M}_2$. Fig. 5 quantitatively plots the tracking errors for position, yaw angles, linear and angular velocity. Our controller's computation time was $3.5ms$ per control input, including forward passes of the learned neural networks, showing that it is suitable for fast real-time applications.

IX. Conclusion

This paper proposes a neural ODE network design for rigid body dynamics learning that captures $SE(3)$ kinematics and Hamiltonian dynamics constraints by construction. It also developed a general control approach for under-actuated trajectory tracking based on the learned $SE(3)$ Hamiltonian dynamics. The learning and control designs are not system-specific and thus can be applied to any mobile robot, whose state evolves on $SE(3)$. These techniques have the potential to enable mobile robots to adapt their models online, in response to changing operational conditions or structural damage, and, yet, maintain stability and autonomous operation. Future work will focus on comparison with other dynamics learning approaches, extending our formulation to allow learning the kinematic and dynamic structure of multi-rigid body systems and provide safe and stable adaptive control, in the presence of noise and disturbances.

X. Appendix

A. Implementation Details

We used fully-connected neural networks whose architecture is shown below. The first number is the input dimension while the last number is the output dimension. The numbers in between are the hidden layers' dimensions and activation functions. The value of ε in (26) is set to 0.01.

1) Pendulum:
   - Input dimension: 9. Action dimension: 1.
   - $\mathbf{L}(\mathfrak{q})$: 9 - 300 Tanh - 300 Tanh - 300 Tanh - 300 Linear - 6.
   - $V(\mathfrak{q})$: 9 - 50 Tanh - 50 Tanh - 50 Linear - 1.
   - $\mathbf{g}(\mathfrak{q})$: 9 - 300 Tanh - 300 Tanh - 300 Linear - 3.
2) Rigid body:
   - Input dimension: 12. Action dimension: 6.
   - $\mathbf{L}_1(\mathfrak{q})$ only takes the position $\mathbf{p} \in \mathbb{R}^3$ as input: 3 - 400 Tanh - 400 Tanh - 400 Tanh - 400 Linear - 6.
   - $\mathbf{L}_2(\mathfrak{q})$ only takes the rotation matrix $\mathbf{R} \in \mathbb{R}^{3\times3}$ as input: 9 - 400 Tanh - 400 Tanh - 400 Tanh - 400 Linear - 6.
   - $V(\mathfrak{q})$: 12 - 400 Tanh - 400 Tanh - 400 Linear - 1.
   - $\mathbf{g}(\mathfrak{q})$: 12 - 400 Tanh - 400 Tanh - 400 Linear - 36.
3) Quadrotor:
   - Input dimension: 12. Action dimension: 4.
   - $\mathbf{L}_1(\mathfrak{q})$ only takes the position $\mathbf{p} \in \mathbb{R}^3$ as input: 3 - 400 Tanh - 400 Tanh - 400 Tanh - 400 Linear - 6.
   - $\mathbf{L}_2(\mathfrak{q})$ only takes the rotation matrix $\mathbf{R} \in \mathbb{R}^{3\times3}$ as input: 9 - 400 Tanh - 400 Tanh - 400 Tanh - 400 Linear - 6.
   - $V(\mathfrak{q})$: 12 - 400 Tanh - 400 Tanh - 400 Linear - 1.
   - $\mathbf{g}(\mathfrak{q})$: 12 - 400 Tanh - 400 Tanh - 400 Linear - 24.

Acknowledgments

We gratefully acknowledge support from NSF RI IIS-2007141 and ARL DCIST CRA W911NF-17-2-0181.

References

[1] J. Á. Acosta, M. Sanchez, and A. Ollero. Robust control of underactuated aerial manipulators via IDA-PBC. In *IEEE Conference on Decision and Control (CDC)*. IEEE, 2014.

[2] T. D. Barfoot. *State Estimation for Robotics*. Cambridge University Press, 2017.

[3] T. Bertalan, F. Dietrich, I. Mezić, and I. G. Kevrekidis. On learning Hamiltonian systems from data. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 29(12), 2019.

[4] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[5] Z. Chen, J. Zhang, M. Arjovsky, and L. Bottou. Symplectic recurrent neural networks. *International Conference on Learning Representations (ICLR)*, 2020.

[6] K. Chua, R. Calandra, R. McAllister, and S. Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[7] O. B. Cieza and J. Reger. IDA-PBC for underactuated mechanical systems in implicit Port-Hamiltonian representation. In *European Control Conference (ECC)*, 2019.

[8] M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel, and S. Ho. Lagrangian neural networks. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020.

[9] M. Deisenroth and C. E. Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *International Conference on Machine Learning (ICML)*, 2011.

[10] J. Delmerico and D. Scaramuzza. A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[11] M. Finzi, K. A. Wang, and A. G. Wilson. Simplifying Hamiltonian and Lagrangian neural networks via explicit constraints. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020.

[12] P. Forni, D. Jeltsema, and G. A. Lopes. Port-Hamiltonian formulation of rigid-body attitude control. *IFAC-PapersOnLine*, 48(13), 2015.

[13] S. Greydanus, M. Dzamba, and J. Yosinski. Hamiltonian neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

[14] J. K. Gupta, K. Menda, Z. Manchester, and M. J. Kochenderfer. A general framework for structured learning of mechanical systems. *arXiv preprint arXiv:1902.08705*, 2019.

[15] D. D. Holm. *Geometric Mechanics*. World Scientific Publishing Company, 2008.

[16] T. Lee, M. Leok, and N. H. McClamroch. Geometric tracking control of a quadrotor UAV on SE(3). In *IEEE Conference on Decision and Control (CDC)*, 2010.

[17] T. Lee, M. Leok, and N. H. McClamroch. *Global formulations of Lagrangian and Hamiltonian dynamics on manifolds*. Springer, 2017.

[18] B. Leimkuhler and S. Reich. *Simulating Hamiltonian dynamics*. Cambridge University Press, 2004.

[19] L. Ljung. System identification. *Wiley encyclopedia of electrical and electronics engineering*, 1999.

[20] A. I. Lurie. *Analytical mechanics*. Springer Science & Business Media, 2013.

[21] M. Lutter, K. Listmann, and J. Peters. Deep Lagrangian Networks for end-to-end learning of energy-based control for under-actuated systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[22] M. Lutter, C. Ritter, and J. Peters. Deep Lagrangian networks: using physics as model prior for deep learning. In *International Conference on Learning Representations (ICLR)*, 2018.

[23] K. M. Lynch and F. C. Park. *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, 2017.

[24] A. J. Maciejewski. Hamiltonian formalism for euler parameters. *Celestial Mechanics*, 37(1), 1985.

[25] S. A. S. Mohamed, M. Haghbayan, T. Westerlund, J. Heikkonen, H. Tenhunen, and J. Plosila. A survey on odometry for autonomous navigation systems. *IEEE Access*, 7, 2019.

[26] D. Nguyen-Tuong and J. Peters. Model learning for robot control: a survey. *Cognitive processing*, 12(4), 2011.

[27] R. Ortega, M. W. Spong, F. Gómez-Estern, and G. Blankenstein. Stabilization of a class of underactuated mechanical systems via interconnection and damping assignment. *IEEE Transactions on Automatic Control*, 47(8), 2002.

[28] J. Panerati, H. Zheng, S. Zhou, J. Xu, A. Prorok, and A. P. Schöllig. Learning to fly: a pybullet gym environment to learn the control of multiple nano-quadcopters. https://github.com/utiasDSL/gym-pybullet-drones, 2020.

[29] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Multistep neural networks for data-driven discovery of nonlinear dynamical systems. *arXiv preprint arXiv:1801.01236*, 2018.

[30] S. Rajappa, M. Ryll, H. H. Bülthoff, and A. Franchi. Modeling, control and design optimization for a fully-actuated hexarotor aerial vehicle with tilted propellers. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[31] R. Rashad, F. Califano, and S. Stramigioli. Port-Hamiltonian passivity-based control on SE(3) of a fully actuated UAV for aerial physical interaction near-hovering. *IEEE Robotics and Automation Letters*, 4(4), 2019.

[32] M. A. Roehrl, T. A. Runkler, V. Brandtstetter, M. Tokic, and S. Obermayer. Modeling system dynamics with physics-informed neural networks based on Lagrangian mechanics. *arXiv preprint arXiv:2005.14617*, 2020.

[33] L. Ruthotto and E. Haber. Deep neural networks motivated by partial differential equations. *Journal of Mathematical Imaging and Vision*, 2019.

[34] A. Sanchez-Gonzalez, N. Heess, J. T. Springenberg, J. Merel, M. Riedmiller, R. Hadsell, and P. Battaglia.

Graph networks as learnable physics engines for inference and control. In *International Conference on Machine Learning (ICML)*, 2018.

[35] R. Shivarama and E. P. Fahrenthold. Hamilton's equations with Euler parameters for rigid body dynamics modeling. *Journal of Dynamic Systems, Measurement, and Control*, 126(1), 2004.

[36] C. Souza, G. V. Raffo, and E. B. Castelan. Passivity based control of a quadrotor UAV. *IFAC Proceedings Volumes*, 47(3), 2014.

[37] P. Toth, D. J. Rezende, A. Jaegle, S. Racanière, A. Botev, and I. Higgins. Hamiltonian generative networks. In *International Conference on Learning Representations (ICLR)*, 2019.

[38] A. Van Der Schaft and D. Jeltsema. Port-Hamiltonian systems theory: An introductory overview. *Foundations and Trends in Systems and Control*, 1(2-3), 2014.

[39] R. Wang, R. Walters, and R. Yu. Incorporating symmetry into deep dynamics models for improved generalization. In *International Conference on Learning Representations (ICLR)*, 2021.

[40] Z. Wang and P. Goldsmith. Modified energy-balancing-based control for the tracking problem. *IET Control Theory & Applications*, 2(4), 2008.

[41] J. Willard, X. Jia, S. Xu, M. Steinbach, and V. Kumar. Integrating physics-based modeling with machine learning: A survey. *arXiv preprint arXiv:2003.04919*, 2020.

[42] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou. Information theoretic MPC for model-based reinforcement learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[43] Y. D. Zhong, B. Dey, and A. Chakraborty. Symplectic ODE-Net: learning Hamiltonian dynamics with control. In *International Conference on Learning Representations (ICLR)*, 2019.

[44] Y. D. Zhong, B. Dey, and A. Chakraborty. Dissipative SymODEN: Encoding Hamiltonian dynamics with dissipation and control into deep learning. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020.