Lie Group Forced Variational Integrator Networks for Learning and Control of Robot Systems

Valentin Duruisseaux VDURUISS@UCSD.EDU
Department of Mathematics, University of California San Diego, La Jolla, CA 92093

Thai Duong TDUONG@UCSD.EDU
Department of Electrical and Computer Engineering, University of California San Diego, La Jolla, CA 92093

Melvin Leok MLEOK@UCSD.EDU
Department of Mathematics, University of California San Diego, La Jolla, CA 92093

Nikolay Atanasov NATANASOV@UCSD.EDU
Department of Electrical and Computer Engineering, University of California San Diego, La Jolla, CA 92093

Editors: N. Matni, M. Morari, G. J. Pappas

Abstract

Incorporating prior knowledge of physics laws and structural properties of dynamical systems into the design of deep learning architectures has proven to be a powerful technique for improving their computational efficiency and generalization capacity. Learning accurate models of robot dynamics is critical for safe and stable control. Autonomous mobile robots, including wheeled, aerial, and underwater vehicles, can be modeled as controlled Lagrangian or Hamiltonian rigid-body systems evolving on matrix Lie groups. In this paper, we introduce a new structure-preserving deep learning architecture, the Lie group Forced Variational Integrator Network (LieFVIN), capable of learning controlled Lagrangian or Hamiltonian dynamics on Lie groups, either from position-velocity or position-only data. By design, LieFVINs preserve both the Lie group structure on which the dynamics evolve and the symplectic structure underlying the Hamiltonian or Lagrangian systems of interest. The proposed architecture learns surrogate discrete-time flow maps allowing accurate and fast prediction without numerical-integrator, neural-ODE, or adjoint techniques, which are needed for vector fields. Furthermore, the learnt discrete-time dynamics can be utilized with computationally scalable discrete-time (optimal) control strategies.

Keywords: Dynamics Learning, Variational Integrators, Symplectic Integrators, Structure-Preserving Neural Networks, Physics-Informed Machine Learning, Predictive Control, Lie Group Dynamics

1. Introduction

Dynamical systems evolve according to physics laws which can be described using differential equations. An accurate model of the dynamics of a control system is important, not only for predicting its future behavior, but also for designing control laws that ensure desirable properties such as safety, stability, and generalization to different operational conditions.

This paper considers the problem of learning dynamics: given a dataset of trajectories from a dynamical system, we wish to infer the update map that generates these trajectories and use it to predict the evolution of the system from different initial states. Models obtained from first principles are used extensively in practice but tend to over-simplify the underlying structure of dynamical systems, leading to prediction errors that cannot be corrected by optimizing over a few model parameters. Deep learning provides very expressive models for function approximation but standard neural networks struggle to learn the symmetries and conservation laws underlying dynamical systems, and as

a result do not generalize well. Deep learning models capable of learning and generalizing dynamics effectively (Willard et al., 2020) are typically over-parameterized and require large datasets and substantial training time, making them prohibitively expensive for applications such as robotics.

A recent research direction has been considering a hybrid approach, which encodes physical laws and geometric properties of the underlying system in the design of the neural network architecture or in the learning process. Prior physics knowledge can be used to construct physics-informed neural networks with improved design and efficiency and better generalization capacity, which take advantage of the function approximation power of neural networks to handle incomplete knowledge. In this paper, we consider learning controlled Lagrangian or Hamiltonian dynamics on Lie groups while preserving the symplectic structure underlying these systems and the Lie group constraints.

Symplectic maps possess numerous special properties and are closely related to Hamiltonian systems. Preserving the symplectic structure of a Hamiltonian system when constructing a discrete approximation of its flow map ensures the preservation of many aspects of the system such as total energy, and leads to physically well-behaved discrete solutions (Leimkuhler and Reich, 2004; Hairer et al., 2006; Holm et al., 2009; Blanes and Casas, 2017). It is thus important to have structure-preserving architectures which can learn flow maps and ensure that the learnt maps are symplectic. Many physics-informed approaches have recently been proposed to learn Hamiltonian dynamics and symplectic maps (Lutter et al., 2019b; Greydanus et al., 2019; Bertalan et al., 2019; Jin et al., 2020; Burby et al., 2020; Chen et al., 2020; Cranmer et al., 2020; Zhong et al., 2020a,b, 2021; Marco and Méhats, 2021; Rath et al., 2021; Chen et al., 2021; Offen and Ober-Blöbaum, 2022; Santos et al., 2022; Valperga et al., 2022; Mathiesen et al., 2022; Duruisseaux et al., 2023a).

Our physics-informed strategy, inspired by (Forced) Variational Integrator Networks ((F)VINs) (Sæmundsson et al., 2020; Havens and Chowdhary, 2021), differs from most of these approaches by learning a discrete-time symplectic approximation to the flow map of the dynamical system, instead of learning the vector field for the continuous-time dynamics. This allows fast prediction for simulation, planning and control without the need to integrate differential equations or use neural ODEs and adjoint techniques. Additionally, the learnt discrete-time dynamics can be combined with computationally scalable discrete-time control strategies.

The novelty of our approach with respect to (F)VINs resides in the enforcement not only of the preservation of symplecticity but also of the Lie group structure when learning a surrogate map for a controlled Lagrangian system which evolves on a Lie group. This is achieved by working in Lie group coordinates instead of Euclidean coordinates, by matching the training data to a parameterized forced Lie group variational integrator which evolves intrinsically on the Lie group. More specifically, we extend the discrete-time Euclidean formulation of FVINs with control from (Havens and Chowdhary, 2021) to Lie groups in a structure-preserving way, which is particularly relevant when considering robot systems (e.g., wheeled, aerial, and underwater vehicles) since they can often be modeled as controlled Lagrangian rigid-body systems evolving on Lie groups.

Given a learnt dynamical system, it is often desirable to control its behavior to achieve stabilization, tracking, or other control objectives. Control designs for continuous-time Hamiltonian systems rely on the Hamiltonian structure (Lutter et al., 2019a; Zhong et al., 2020a; Duong and Atanasov, 2021, 2022). Since the Hamiltonian captures the system energy, control techniques for stabilization inject additional energy into the system via the control input to ensure that the minimum of the total energy is at a desired equilibrium. For fully-actuated Hamiltonian systems, it is sufficient to shape the potential energy only using energy-shaping and damping-injection (ES-DI) (Van Der Schaft and Jeltsema, 2014). For under-actuated systems, both the kinetic and potential energies are shaped,

e.g., via interconnection and damping assignment passivity-based control (IDA-PBC) (Ortega et al., 2002; Van Der Schaft and Jeltsema, 2014; Acosta et al., 2014; Cieza and Reger, 2019). The most widely used control approach for discrete-time dynamics is based on Model Predictive Control (MPC) (Borrelli et al., 2017; Grüne and Pannek, 2017). MPC techniques determine an open-loop control sequence that solves a finite-horizon optimal control problem, apply the first few control inputs, and repeat the process. A key result in MPC is that an appropriate choice of terminal cost and terminal constraints in the sequence of finite-horizon problems can guarantee recursive feasibility and asymptotic optimality with respect to the infinite-horizon cost (Borrelli et al., 2017). The ability to learn a structure-preserving discrete-time model of a dynamics system enabled by this paper, also allows employing MPC techniques for optimal control of the learnt system dynamics.

2. Preliminaries

We first review the basic theory of continuous-time Lagrangian and Hamiltonian systems, before describing their underlying symplectic structure and how variational integrators preserve that structure. Finally, we discuss how external forcing and control can be added to variational integrators.

2.1. Geometric Mechanics

The set of tangent vectors to a manifold $\mathcal{Q}$ at a point $q \in \mathcal{Q}$ is a vector space called the tangent space $T_q\mathcal{Q}$ to $\mathcal{Q}$ at q. The disjoint union of all the tangent spaces to $\mathcal{Q}$ forms the tangent bundle $T\mathcal{Q} = \{(q, v) | q \in \mathcal{Q}, v \in T_q\mathcal{Q}\}$ of $\mathcal{Q}$. The vector space dual to the tangent space $T_q\mathcal{Q}$ is the cotangent space $T_q^*\mathcal{Q}$, and the vector bundle over $\mathcal{Q}$ whose fibers are the cotangent spaces of $\mathcal{Q}$ is the cotangent bundle $T^*\mathcal{Q} = \{(q, p) | q \in \mathcal{Q}, p \in T_q^*\mathcal{Q}\}$.

Given a manifold $\mathcal{Q}$, a Lagrangian is a function $L : T\mathcal{Q} \to \mathbb{R}$. Hamilton's Variational Principle states that $\delta \int_0^T L(q(t), \dot{q}(t))dt = 0$, where the variation is induced by an infinitesimal variation δq that vanishes at the endpoints. Hamilton's Principle is equivalent to the Euler–Lagrange equations

$$\frac{\partial L}{\partial q}(q, \dot{q}) - \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}(q, \dot{q})\right) = 0. \tag{1}$$

Given a Lagrangian L, we define the conjugate momentum $p \in T^*\mathcal{Q}$ via the Legendre transform $p = \frac{\partial L}{\partial \dot{q}}$, and obtain a Hamiltonian $H(q, p) = \sum_{j=1}^n p_j \dot{q}^j - L(q, \dot{q})\big|_{p_i = \frac{\partial L}{\partial \dot{q}^i}}$ on $T^*\mathcal{Q}$. There is a variational principle on the Hamiltonian side which is equivalent to Hamilton's equations and to the Euler–Lagrange equations (1) when the Legendre transform is diffeomorphic. For most mechanical systems, the Legendre transform is diffeomorphic and thus the Lagrangian and Hamiltonian formulations are equivalent. The approaches presented here are based on the Lagrangian formulation, but also apply to the equivalent Hamiltonian systems whenever they are well-defined.

2.2. Symplecticity

A smooth mapping $(q, p) \mapsto (\bar{q}, \bar{p})$ is symplectic if it preserves the symplectic two-form, that is $\sum_{i=1} \mathbf{d}q^i \wedge \mathbf{d}p_i = \sum_{i=1} \mathbf{d}\bar{q}^i \wedge \mathbf{d}\bar{p}_i$. Hamiltonian systems and symplectic flows are closely related: solutions to Hamiltonian systems are symplectic flows (Poincaré, 1899), and symplectic flows are locally Hamiltonian. When applied to Hamiltonian systems, symplectic integrators yield discrete approximations of the flow that preserve the symplectic two-form, which results in the preservation of many qualitative aspects of the dynamical system and leads to physically well-behaved solutions.

See (Leimkuhler and Reich, 2004; Hairer et al., 2006; Blanes and Casas, 2017) for a comprehensive presentation of geometric numerical integration.

2.3. Variational Integrators

Variational integrators are obtained by discretizing Hamilton's principle, instead of discretizing the equations of motion, are thus symplectic, preserve many invariants, and exhibit excellent long-time near-energy preservation (Marsden and West, 2001). The exact discrete Lagrangian generating the time-h flow can be represented in boundary-value form as $L_d^E(q_0, q_1) = \int_0^h L(q(t), \dot{q}(t))dt$, where $q(t)$ satisfies the Euler–Lagrange equations on $[0, h]$ with $q(0) = q_0$, $q(h) = q_1$. After constructing an approximation L_d to L_d^E, the Lagrangian variational integrator is defined implicitly by the discrete Euler–Lagrange equation, $D_2L_d(q_{k-1}, q_k) + D_1L_d(q_k, q_{k+1}) = 0$, which can also be written in Hamiltonian form, using discrete momenta p_k, as $p_k = -D_1L_d(q_k, q_{k+1})$ and $p_{k+1} = D_2L_d(q_k, q_{k+1})$, where D_i denotes a partial derivative with respect to the i-th argument. Many properties of the integrator, such as momentum conservation and error analysis guarantees, can be determined by analyzing the discrete Lagrangian, instead of analyzing the integrator directly.

Examples of variational integrators include Taylor (Schmitt et al., 2018), Galerkin (Marsden and West, 2001; Leok and Zhang, 2011), prolongation-collocation (Leok and Shingel, 2012), and constrained (Marsden and West, 2001; Duruisseaux and Leok, 2022) variational integrators. Variational integrators can also be developed for Hamiltonian dynamics (Lall and West, 2006; Leok and Zhang, 2011; Schmitt and Leok, 2017; Duruisseaux et al., 2021), and can be used with prescribed variable time-steps (Duruisseaux et al., 2021; Duruisseaux and Leok, 2023).

2.4. Forced Variational Integrators

External forcing and control can be added to variational integrators (Marsden and West, 2001; Ober-Blöbaum et al., 2011). Let $u(t)$ be the control parameter in some control manifold $\mathcal{U}$, and consider a Lagrangian control force $f_L : T\mathcal{Q} \times \mathcal{U} \to T^*\mathcal{Q}$. Hamilton's principle can be modified into the Lagrange–d'Alembert Principle

$$\delta \int_0^T L(q(t), \dot{q}(t))dt + \int_0^T f_L(q(t), \dot{q}(t), u(t)) \cdot \delta q(t)dt = 0, \tag{2}$$

where the variation is induced by an infinitesimal variation δq that vanishes at the endpoints. This variational principle is equivalent to the forced Euler–Lagrange equations

$$\frac{\partial L}{\partial q}(q, \dot{q}) - \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}(q, \dot{q})\right) + f_L(q, \dot{q}, u) = 0. \tag{3}$$

Using a discrete Lagrangian L_d, and discrete Lagrangian control forces $f_d^{\pm} : \mathcal{Q} \times \mathcal{Q} \times \mathcal{U} \to T^*\mathcal{Q}$ to approximate the virtual work of the Lagrangian control force f_L,

$$\int_{t_k}^{t_{k+1}} f_L(q(t), \dot{q}(t), u(t)) \cdot \delta q(t)dt \approx f_d^-(q_k, q_{k+1}, u_k) \cdot \delta q_k + f_d^+(q_k, q_{k+1}, u_k) \cdot \delta q_{k+1}, \tag{4}$$

one can obtain a forced variational integrator from the forced discrete Euler–Lagrange equations

$$D_2L_d(q_{k-1}, q_k) + D_1L_d(q_k, q_{k+1}) + f_d^+(q_{k-1}, q_k, u_{k-1}) + f_d^-(q_k, q_{k+1}, u_k) = 0, \tag{5}$$

which can also be written in Hamiltonian form as

$$p_k = -D_1L_d(q_k, q_{k+1}) - f_d^-(q_k, q_{k+1}, u_k), \quad p_{k+1} = D_2L_d(q_k, q_{k+1}) + f_d^+(q_k, q_{k+1}, u_k). \tag{6}$$

3. Problem Statement

We consider the problem of learning controlled Lagrangian dynamics. Given a position-velocity dataset of trajectories, we wish to infer the flow map generating these trajectories, while preserving the system's symplectic structure and constraining the updates to the Lie group on which it evolves. For example, a rigid-body robot system may be modeled as a Lagrangian system evolving on the Lie group $\mathrm{SE}(3)$ of rigid-body transformations. Learning its dynamics from trajectory data should respect kinematic and energy conservation. More precisely, we consider the following problem.

Problem 1 *Let $\mathcal{Q}$ be a Lie group and $\mathcal{D}_{T\mathcal{Q}}$ be a distance metric on $T\mathcal{Q}$. Given a dataset of position-velocity updates $\left\{\left(q_0^{(i)}, \dot{q}_0^{(i)}, u_0^{(i)}\right) \mapsto \left(q_1^{(i)}, \dot{q}_1^{(i)}\right)\right\}_{i=1}^N$ for a controlled Lagrangian dynamical system evolving on $\mathcal{Q}$, we wish to find a symplectic mapping $\Psi : T\mathcal{Q} \times \mathcal{U} \to T\mathcal{Q}$ which minimizes*

$$\sum_{i=1}^{N} \mathcal{D}_{T\mathcal{Q}}\left(\left(q_1^{(i)}, \dot{q}_1^{(i)}\right), \Psi\left(q_0^{(i)}, \dot{q}_0^{(i)}, u_0^{(i)}\right)\right). \tag{7}$$

4. Lie group Forced Variational Integrators Networks (LieFVINs)

To solve Problem 1, we introduce Lie group Forced Variational Integrators Networks (**LieFVINs**). Our main idea is to parametrize the updates of a forced Lie group variational integrator and match them with observed updates. We focus on specific forced $\mathrm{SO}(3)$ and $\mathrm{SE}(3)$ variational integrators, but the general strategy extends to any Lie group forced variational integrator.

4.1. The $\mathrm{SO}(3)$ and $\mathrm{SE}(3)$ Lie Groups

The 3-dimensional special orthogonal group $\mathrm{SO}(3) = \{R \in \mathbb{R}^{3\times 3} | RR^\top = \mathbb{I}_3, \det(R) = 1\}$, where $\mathbb{I}_k$ denotes the $k \times k$ identity matrix, is the Lie group of rotations about the origin in $\mathbb{R}^3$. The Lie algebra of $\mathrm{SO}(3)$ is the space of skew-symmetric matrices $\mathfrak{so}(3) = \{A \in \mathbb{R}^{3\times 3} | A^\top = -A\}$, with the matrix commutator $[A, B] = AB - BA$ as the Lie bracket. The sets $\mathbb{R}^3$ and $\mathfrak{so}(3)$ are isomorphic via the hat map $S(\cdot) : \mathbb{R}^3 \to \mathfrak{so}(3)$, defined by $S(x)y = x \times y$ for any $x, y \in \mathbb{R}^3$.

The Special Euclidean group in 3 dimensions, $\mathrm{SE}(3)$, is a semidirect product of $\mathbb{R}^3$ and $\mathrm{SO}(3)$ and is diffeomorphic to $\mathbb{R}^3 \times \mathrm{SO}(3)$. Elements of $\mathrm{SE}(3)$ can be written as $(x, R) \in \mathbb{R}^3 \times \mathrm{SO}(3)$, and the Lie algebra $\mathfrak{se}(3)$ of $\mathrm{SE}(3)$ is composed of elements $(y, A) \in \mathbb{R}^3 \times \mathfrak{so}(3)$.

The pose of a rigid body can be described by an element (x, R) of $\mathrm{SE}(3)$, consisting of position $x \in \mathbb{R}^3$ and orientation $R \in \mathrm{SO}(3)$. See Duruisseaux et al. (2023b, Appendix A) for more details about rigid-body kinematics.

4.2. Forced Variational Integrator on $\mathrm{SO}(3)$ and $\mathrm{SE}(3)$

On $\mathrm{SE}(3)$, $q = (x, R)$ and $\dot{q} = (v, \omega)$ where x is position, R is orientation, v is velocity, and ω is angular velocity. A Lagrangian on $\mathrm{SE}(3)$ is given by

$$L(x, R, v, \omega) = \frac{1}{2} v^\top m v + \frac{1}{2} \omega^\top J \omega - U(x, R), \tag{8}$$

where m is mass, $J \in \mathbb{R}^{3\times 3}$ is a symmetric positive-definite inertia matrix, U is potential energy.

Consider the continuous-time kinematics equation $\dot{R} = RS(\omega)$, with constant $\omega(t) \equiv \omega_k$ for a short period of time $t \in [t_k, t_{k+1})$ where $t_{k+1} = t_k + h$. Then, $R(t_{k+1}) = R(t_k)\exp(hS(\omega_k))$.

Thus, with $R_k := R(t_k)$, $R_{k+1} := R(t_{k+1})$ and $Z_k := \exp(hS(\omega_k))$, we obtain $R_{k+1} = R_k Z_k$ and for sufficiently small h, we have $Z_k \approx \mathbb{I}_3 + hS(\omega_k)$. With $(x_k, R_k) \in \mathrm{SE}(3)$, the discrete SE(3) kinematic equations are given by $R_{k+1} = R_k Z_k$ and $x_{k+1} = x_k + R_k y_k$ where $(y_k, Z_k) \in \mathrm{SE}(3)$, which ensures that the sequence of updates $\{(x_k, R_k)\}_k$ remains on SE(3).

Using the approximation $S(\omega_k) \approx \frac{1}{h}(Z_k - \mathbb{I}_3)$, we choose the discrete Lagrangian

$$\begin{aligned} L_d(x_k, R_k, y_k, Z_k) = \frac{m}{2h} y_k^\top y_k + \frac{1}{h}\mathrm{tr}\left([\mathbb{I}_3 - Z_k]J_d\right) \\ - (1-\alpha)hU(x_k, R_k) - \alpha h U(x_k + R_k y_k, R_k Z_k), \end{aligned} \tag{9}$$

where $\alpha \in [0,1]$ and $J_d = \frac{1}{2}\mathrm{tr}(J)\mathbb{I}_3 - J$. Equation (9) gives a simple approximation to the exact SE(3) discrete Lagrangian, while maintaining some flexibility in the two-point quadrature weights through the tunable parameter α. Higher-order approximations could also be used, but the resulting discrete equations of motion would typically be more complicated and expensive to evolve.

We denote $U_k = U(x_k, R_k)$ and define ξ_k via $S(\xi_k) = \frac{\partial U_k}{\partial R_k}^\top R_k - R_k^\top \frac{\partial U_k}{\partial R_k}$. In Duruisseaux et al. (2023b, Appendix B), we show that the forced discrete Euler–Lagrange equations associated to the discrete Lagrangian (9) and discrete control forces $f_{d_k}^{\pm} \equiv f_d^{\pm}(x_k, R_k, u_k)$ with R and x components $f_d^{R\pm}, f_d^{x\pm}$ can be written in Hamiltonian form, using $\pi_k = J\omega_k$ and $\gamma_k = mv_k$, as

$$hS(\pi_k) + hS(f_{d_k}^{R-}) + (1-\alpha)h^2 S(\xi_k) = Z_k J_d - J_d Z_k^\top, \tag{10}$$

$$R_{k+1} = R_k Z_k, \tag{11}$$

$$\pi_{k+1} = Z_k^\top \pi_k + (1-\alpha)hZ_k^\top \xi_k + \alpha h \xi_{k+1} + Z_k^\top f_{d_k}^{R-} + f_{d_k}^{R+}, \tag{12}$$

$$x_{k+1} = x_k + \frac{h}{m}\gamma_k - (1-\alpha)\frac{h^2}{m}\frac{\partial U_k}{\partial x_k} - \frac{h}{m} R_k f_{d_k}^{x-}, \tag{13}$$

$$\gamma_{k+1} = \gamma_k - (1-\alpha)h\frac{\partial U_k}{\partial x_k} - \alpha h \frac{\partial U_{k+1}}{\partial x_{k+1}} + R_k f_{d_k}^{x-} + R_{k+1} f_{d_k}^{x+}. \tag{14}$$

Given $(x_k, R_k, \gamma_k, \pi_k, u_k)$, we first solve equation (10) which is of the form $S(a) = ZJ_d - J_d Z^\top$ as outlined in Remark 1, and then get $R_{k+1} = R_k Z_k$. We then obtain π_{k+1}, x_{k+1} and γ_{k+1} from equations (12)-(14). The discrete equations of motion can be rewritten as an update from $(x_k, R_k, v_k, \omega_k, u_k)$ to $(x_{k+1}, R_{k+1}, v_{k+1}, \omega_{k+1})$ by using $\pi_k = J\omega_k$ and $\gamma_k = mv_k$.

Remark 1 *$S(a) = ZJ_d - J_d Z^\top$ can be converted into an equivalent vector equation*

$$\phi(\mathfrak{z}) \equiv a + a \times \mathfrak{z} + \mathfrak{z}(a^\top \mathfrak{z}) - 2J\mathfrak{z} = 0, \qquad \mathfrak{z} \in \mathbb{R}^3, \tag{15}$$

as shown in Duruisseaux et al. (2023b, Appendix C), using the Cayley transform

$$Z = Cay(\mathfrak{z}) \equiv (\mathbb{I}_3 + S(\mathfrak{z}))(\mathbb{I}_3 - S(\mathfrak{z}))^{-1} = \frac{1}{1 + \|\mathfrak{z}\|_2^2}\left((1 - \|\mathfrak{z}\|_2^2)\mathbb{I}_3 + 2S(\mathfrak{z}) + 2\mathfrak{z}\mathfrak{z}^\top\right). \tag{16}$$

The solution $Z = Cay(\mathfrak{z})$ to the original equation $S(a) = ZJ_d - J_d Z^\top$ can be obtained after solving this vector equation for $\mathfrak{z}$ by using (typically 2 or 3 steps of) Newton's method:

$$\mathfrak{z}^{(n+1)} = \mathfrak{z}^{(n)} - \left[\nabla\phi(\mathfrak{z}^{(n)})\right]^{-1}\phi(\mathfrak{z}^{(n)}), \qquad \nabla\phi(\mathfrak{z}) = S(a) + (a^\top \mathfrak{z})\mathbb{I}_3 + \mathfrak{z}a^\top - 2J. \tag{17}$$

4.3. Lie Group Forced Variational Integrator Networks (LieFVINs) on SE(3)

We now describe the construction of Lie group Forced Variational Integrator Networks (**LieFVINs**), for the forced variational integrator on SE(3) presented in Section 4.2. The idea is to parametrize the updates of the integrator and match them with observed updates. Here, we consider the case where position-velocity data is available, in which case the LieFVIN is based on equations (10)-(14). The case where only position data is available is presented in Duruisseaux et al. (2023b, Appendix E).

We parametrize m, $f_d^{\pm}$ and U as neural networks. The inertia J is a symmetric positive-definite matrix-valued function of (x, R) constructed via a Cholesky decomposition $J = LL^\top$ for a lower-triangular matrix L implemented as a neural network. Given J, we also obtain $J_d = \frac{1}{2}\text{tr}(J)\mathbb{I}_3 - J$. To deal with the implicit nature of equation (10), we propose two algorithms, based either on an explicit iterative solver or by penalizing deviations away from equation (10):

Algorithm Ia. Given position-velocity data $\{(x_0, R_0, v_0, \omega_0, u_0) \mapsto (x_1, R_1, v_1, \omega_1)\}$, minimize discrepancies between the observed $(x_1, R_1, v_1, \omega_1)$ quadruples and the predicted $(\tilde{x}_1, \tilde{R}_1, \tilde{v}_1, \tilde{\omega}_1)$ quadruples, obtained as follows: for each $(x_0, R_0, v_0, \omega_0, u_0)$ data tuple,

1. Get $f_{d_0}^{R\pm}$ and $f_{d_0}^{x\pm}$ from (x_0, R_0, u_0), and ξ_0 from $S(\xi_0) = \frac{\partial U_0}{\partial R_0}^\top R_0 - R_0^\top \frac{\partial U_0}{\partial R_0}$
2. Get $Z_0 = \text{Cay}(\mathcal{z})$ where $\mathcal{z}$ is obtained using a few steps of Newton's method to solve the vector equation (15) equivalent to $hS(J\omega_0) + hS(f_{d_0}^{R-}) + (1-\alpha)h^2 S(\xi_0) = ZJ_d - J_d Z^\top$
3. Compute $\tilde{R}_1 = R_0 Z_0$, and then get ξ_1 from $S(\xi_1) = \frac{\partial U_1}{\partial \tilde{R}_1}^\top \tilde{R}_1 - \tilde{R}_1^\top \frac{\partial U_1}{\partial \tilde{R}_1}$
4. Get $\tilde{\omega}_1$ from $J\tilde{\omega}_1 = Z_0^\top J\omega_0 + (1-\alpha)hZ_0^\top \xi_0 + \alpha h\xi_1 + Z_0^\top f_{d_0}^{R-} + f_{d_0}^{R+}$
5. Compute $\begin{bmatrix}\tilde{x}_1 \\ \tilde{v}_1\end{bmatrix} = \begin{bmatrix}x_0 \\ v_0\end{bmatrix} + \frac{1}{m}\begin{bmatrix} hmv_0 - (1-\alpha)h^2 m\frac{\partial U_0}{\partial x_0} - hR_0 f_{d_0}^{x-} \\ -(1-\alpha)h^2\frac{\partial U_0}{\partial x_0} - \alpha h^2 \frac{\partial U_1}{\partial x_1} + R_0 f_{d_0}^{x-} + R_1 f_{d_0}^{x+}\end{bmatrix}$

Algorithm Ib. Given position-velocity data $\{(x_0, R_0, v_0, \omega_0, u_0) \mapsto (x_1, R_1, v_1, \omega_1)\}$, minimize

- Discrepancies between the observed (x_1, v_1, ω_1) triples and the predicted $(\tilde{x}_1, \tilde{v}_1, \tilde{\omega}_1)$ triples
- Deviations away from the equation $hS(J\omega_0) + hS(f_{d_0}^{R-}) + (1-\alpha)h^2 S(\xi_0) = J_d Z_0 - Z_0^\top J_d$

where, for each $(x_0, R_0, v_0, \omega_0, u_0, R_1)$ data tuple,

1. $f_{d_0}^{R\pm}$ and $f_{d_0}^{x\pm}$ are obtained from (x_0, R_0, u_0), and ξ_0, ξ_1 from $S(\xi_k) = \frac{\partial U_k}{\partial R_k}^\top R_k - R_k^\top \frac{\partial U_k}{\partial R_k}$
2. $Z_0 = R_0^\top R_1$ and $\tilde{\omega}_1 = J^{-1}\left[Z_0^\top J\omega_0 + (1-\alpha)hZ_0^\top \xi_0 + \alpha h\xi_1 + Z_0^\top f_{d_0}^{R-} + f_{d_0}^{R+}\right]$
3. $\begin{bmatrix}\tilde{x}_1 \\ \tilde{v}_1\end{bmatrix} = \begin{bmatrix}x_0 \\ v_0\end{bmatrix} + \frac{1}{m}\begin{bmatrix} hmv_0 - (1-\alpha)h^2 m\frac{\partial U_0}{\partial x_0} - hR_0 f_{d_0}^{x-} \\ -(1-\alpha)h^2\frac{\partial U_0}{\partial x_0} - \alpha h^2 \frac{\partial U_1}{\partial x_1} + R_0 f_{d_0}^{x-} + R_1 f_{d_0}^{x+}\end{bmatrix}$

This general strategy extends to any other Lie group integrator. In particular, LieFVINs on SO(3) can be obtained from the algorithms above as the special case where x is constant, in which case we can disregard all the variables and operations in green. Lie group variational integrator networks without forces (**LieVINs**) can be obtained by setting $f_{d_0}^{R\pm} = f_{d_0}^{x\pm} = 0$. Note that the strategy behind Algorithm Ia enforces the structure of the system in a stronger way than in Algorithm Ib. However, for certain Lie groups and variational integrators, it might not be practical to use Newton's method to solve for the implicit updates, in which case Algorithm Ib is preferred.

4.4. Control Strategy

Given the discrete-time flow map Ψ learnt by a LieFVIN, we can formulate a Model Predictive Control (MPC) problem to design a discrete-time control policy for the dynamical system:

At each step $t_\ell = \ell h$,

1. Obtain an estimate $(\tilde{q}_\ell, \dot{\tilde{q}}_\ell)$ of the current state.
2. Solve a N-step finite horizon optimal control problem starting at $(\tilde{q}_\ell, \dot{\tilde{q}}_\ell)$, formulated as a constrained optimization problem: *Minimize the discrete cost function*

$$\mathcal{J}_d(U_\ell) = \sum_{k=0}^{N-1} \mathcal{C}_d(q_{\ell+k}, q_{\ell+k+1}, \dot{q}_{\ell+k}, u_{\ell+k}) + \Phi_d(q_{\ell+N-1}, q_{\ell+N}, \dot{q}_{\ell+N}, u_{\ell+N-1}), \quad (18)$$

   over admissible discrete controls $U_\ell = \{u_\ell, u_{\ell+1}, ..., u_{\ell+N-1}\}$, *subject to path constraints* $\mathcal{P}_d(q_{\ell+k}, q_{\ell+k+1}, \dot{q}_{\ell+k}, u_{\ell+k}) \geq 0$ *for* $k = 1, ..., N-1$ *and to the termination condition* $\mathcal{T}_d(q_{\ell+N-1}, q_{\ell+N}, \dot{q}_{\ell+N}, u_{\ell+N-1}) = 0$, *and where the evolution of the controlled system is prescribed by the surrogate symplectic map* Ψ *learnt by the LieFVIN.*
3. Apply the resulting optimal control u_ℓ^* to the system in state $(\tilde{q}_\ell, \dot{\tilde{q}}_\ell)$ until $t_{\ell+1} = (\ell+1)h$.

Note that the Lie group constraints do not need to be added as path constraints since they are automatically satisfied to (almost) machine precision, by the design of the LieFVINs. In our experiments, we use the PyTorch MPC framework[1] (Tassa et al., 2014; Amos et al., 2018).

5. Evaluation

We now demonstrate our approach to learn and control a planar pendulum and a crazyflie quadrotor. More details about our implementation can be found in Duruisseaux et al. (2023b, Appendix D), and our Python/PyTorch code is available at https://thaipduong.github.io/LieFVIN/

5.1. Pendulum

We consider a planar pendulum with dynamics $\ddot{\varphi} = -15 \sin \varphi + 3u$, where φ is the angle with respect to its downward position $\varphi = 0$ and $u \in \mathbb{R}$ is a control input. The mass of the pendulum, the potential energy, and input coefficient are given by $m = 1/3$, $U(\varphi) = 5(1 - \cos \varphi)$, $g(\varphi) = 1$. We collected $\{(\cos \varphi, \sin \varphi, \dot{\varphi})\}$ data from an OpenAI Gym environment (Zhong et al., 2020a). LieFVIN was trained with position-velocity data as described in Algorithm Ia with $\alpha = 0.5$. The forces were specified as $f_d^{R+} = 0$ and $f_d^{R-} = g(q)u$, where $g(q)$ is a neural network.

Figures 1(a), (b), (c) show that the LieFVIN model learned the correct inertia matrix J, control gain $g(q)$, and potential energy U (up to a constant offset). Without control input, i.e., $f_d^{R\pm} = 0$, we use the dynamics model learnt from short-term trajectories of 10 steps of 0.02s to generate long-term predictions (2000 steps, i.e. 40s). Figure 1(d) shows that the total energy of the learnt system fluctuates but stays close to the ground truth value. The fluctuation comes from the discretization errors in equations (10)-(14) and model errors for the learnt quantities J, U, and $g(q)$. Note that the $SO(3)$ constraint errors remain very small, around 10^{-14} (see Figure 1(e)). The phase portraits and

1. Code: https://locuslab.github.io/mpc.pytorch/

the learnt dynamics are close to the ground-truth ones, illustrating the ability to generate long-term predictions using the model learnt from short-term data. Meanwhile, a Multilayer Perceptron black-box model, described in Duruisseaux et al. (2023b, Appendix D.1), struggles to infer the $\mathrm{SO}(3)$ constraints from data (Figure 1(e)(g)) and is not able to conserve the total energy (Figure 1(d)).

The learnt dynamics model is combined with MPC as described in Section 4.4 to drive the pendulum from downward position $\varphi = 0$ to a stabilized upright position $\varphi^* = \pi$, $\dot{\varphi}^* = 0$, with input constraint $|u| \leq 20$. Figure 1(h) plots the angle φ, angular velocity $\dot{\varphi}$, and control input u, showing that the pendulum is successfully stabilized using the learnt discrete dynamics model.

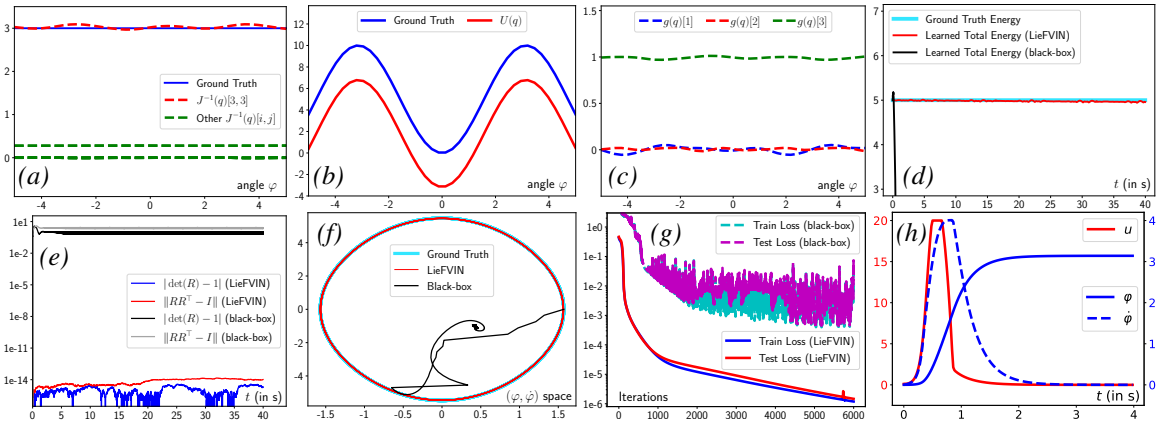

Figure 1: Evaluation of $\mathrm{SO}(3)$ LieFVIN on a pendulum. We learned the inertia matrix (a), potential energy (b), and input coefficient (c), with the loss function shown in (g). The learnt model respects the energy conservation law (d), $\mathrm{SO}(3)$ constraints (e), and phase portraits (f). The control from MPC is shown in (h). Meanwhile, a black-box model struggles to infer the $\mathrm{SO}(3)$ constraints from data (e)(g) and is not able to conserve energy (d).

5.2. Crazyflie Quadrotor

We demonstrate that our $\mathrm{SE}(3)$ dynamics learning and control approach can achieve trajectory tracking for an under-actuated system by considering a Crazyflie quadrotor simulated using PyBullet (Panerati et al., 2020). The control input $\mathbf{u} = [f, \boldsymbol{\tau}]$ includes the thrust $f \in \mathbb{R}_{\geq 0}$ and torque vector $\boldsymbol{\tau} \in \mathbb{R}^3$ generated by the 4 rotors. LieFVIN is trained as in Algorithm Ib with $\alpha = 0.5$. The forces are specified as $f_d^{x\pm} = 0.5 g_x(q) u$ and $f_d^{R\pm} = 0.5 g_R(q) u$ where $g_x(q), g_R(q)$ are neural networks.

Figures 2(a)-(e) show that LieFVIN learned the correct mass m, inertia matrix J, control gains $g_x(q)$ and $g_R(x)$, and potential energy $U(q)$ (up to a constant offset). Without control input, i.e., $f_d^{R\pm} = 0$, we use the dynamics model learnt from short-term trajectories of 5 steps of 0.02s to generate long-term predictions (2000 steps, i.e. 40s). Figure 2(f) shows that the total energy of the system has bounded fluctuations while $\mathrm{SO}(3)$ constraint errors are around 10^{-14}, verifying the near-energy conservation and manifold constraints guaranteed by our approach.

The learnt model is then combined with MPC as in Section 4.4 to track a diamond-shaped trajectory, with control input constraints $0 \leq f \leq 0.595$, $|\tau| \leq 10^{-3}[5.9 \quad 5.9 \quad 7.4]^\top$. Figure 3 displays the robot trajectory and plots the tracking errors over time, showing that the quadrotor successfully completes the task.

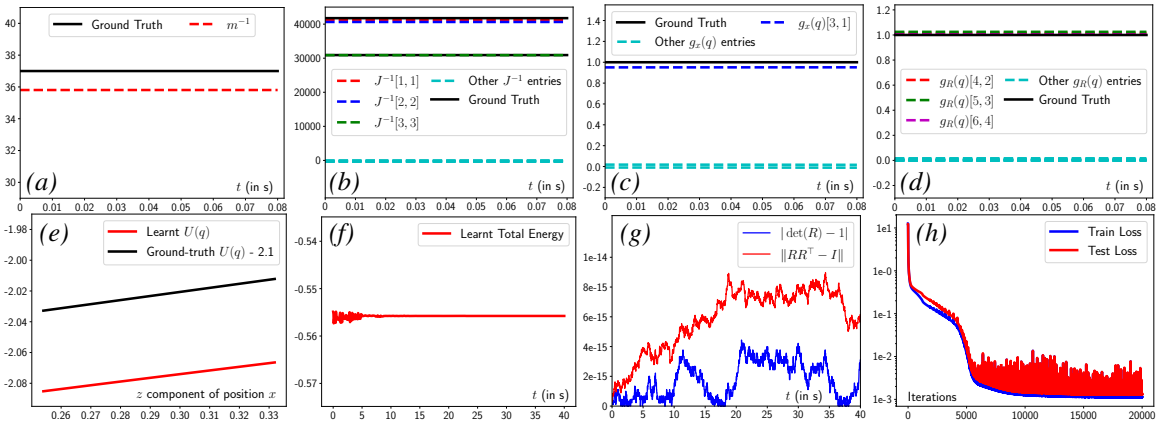

Figure 2: LieFVIN learns the correct mass m (a), inertia matrix J (b), input coefficients $g_x(q)$ (c) and $g_R(q)$ (d), potential energy $U(q)$ (e). The learnt model respects the energy conservation law (f), SO(3) constraints (g). The evolution of the loss function is shown in (h).

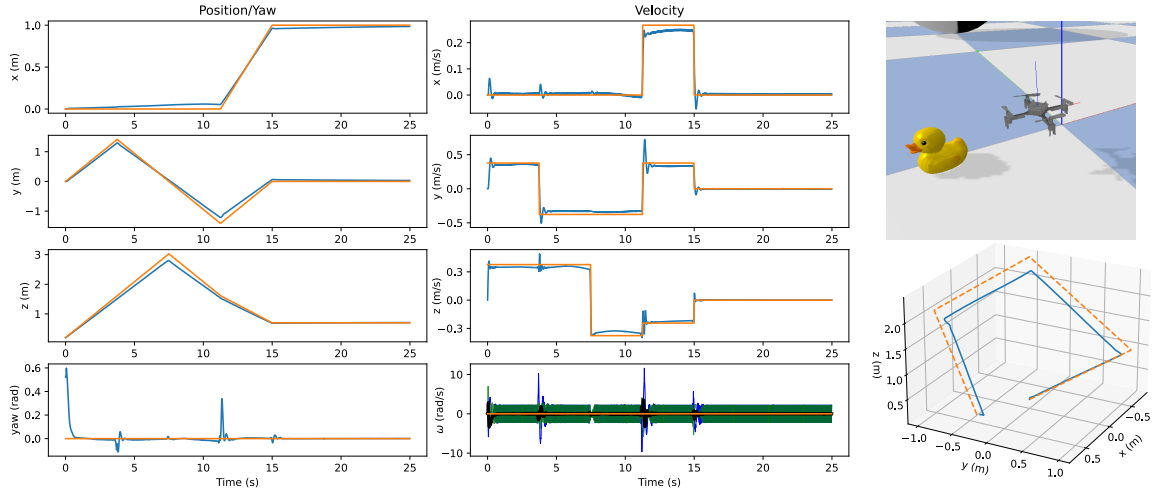

Figure 3: Trajectory tracking with the learned quadrotor model. The tracking errors (left) between reference trajectory (orange) and the actual trajectory, and the robot trajectory (lower right) show that the task is completed successfully.

6. Conclusion

We introduced a new structure-preserving deep learning strategy to learn discrete-time flow maps for controlled Lagrangian or Hamiltonian dynamics on a Lie group, from position-velocity or position-only data. The resulting maps evolve intrinsically on the Lie group and preserve the symplecticity underlying the systems of interest, which allows to generate physically well-behaved long-term predictions based on short-term trajectories data. Learning discrete-time flow maps instead of vector fields yields better prediction without requiring the use of a numerical integrator, neural ODE, or adjoint techniques. The proposed approach can also be combined with discrete-time optimal control strategies, for instance to achieve stabilization and tracking for robot systems on SE(3). Possible future directions include extensions to multi-link robots and multi-agent systems (e.g. on $(\mathrm{SE}(3))^n$).

Acknowledgments

The authors gratefully acknowledge support from NSF under grants CCF-2112665, DMS-1345013, DMS-1813635 and from AFOSR under grant FA9550-18-1-0288.

References

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

B. Amos, I. Jimenez, J. Sacks, B. Boots, and J. Z. Kolter. Differentiable MPC for End-to-end Planning and Control. In *Advances in Neural Information Processing Systems*, 2018.

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):121107, 2019. doi: 10.1063/1.5128231.

S. Blanes and F. Casas. *A Concise Introduction to Geometric Numerical Integration*. 2017. ISBN 9781482263442. doi: 10.1201/b21563.

F. Borrelli, A. Bemporad, and M. Morari. *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

J. W. Burby, Q. Tang, and R. Maulik. Fast neural Poincaré maps for toroidal magnetic fields. *Plasma Physics and Controlled Fusion*, 63(2):024001, 2020. doi: 10.1088/1361-6587/abcbaa.

Y. Chen, T. Matsubara, and T. Yaguchi. Neural symplectic form: learning Hamiltonian equations on general coordinate systems. In *Advances in Neural Information Processing Systems*, 2021.

Z. Chen, J. Zhang, M. Arjovsky, and L. Bottou. Symplectic Recurrent Neural Networks. *International Conference on Learning Representations*, 2020.

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

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

T. Duong and N. Atanasov. Hamiltonian-based Neural ODE Networks on the SE(3) Manifold For Dynamics Learning and Control. In *Proceedings of Robotics: Science and Systems*, 2021. doi: 10.15607/RSS.2021.XVII.086.

T. Duong and N. Atanasov. Adaptive control of SE(3) Hamiltonian dynamics with learned disturbance features. *IEEE Control Systems Letters*, 2022.

V. Duruisseaux and M. Leok. Accelerated optimization on Riemannian manifolds via discrete constrained variational integrators. *Journal of Nonlinear Science*, 32(42), 2022.

V. Duruisseaux and M. Leok. Time-adaptive Lagrangian variational integrators for accelerated optimization on manifolds. *Journal of Geometric Mechanics*, 15(1):224–255, 2023. ISSN 1941-4889.

V. Duruisseaux, J. Schmitt, and M. Leok. Adaptive Hamiltonian variational integrators and applications to symplectic accelerated optimization. *SIAM Journal on Scientific Computing*, 43(4): A2949–A2980, 2021.

V. Duruisseaux, J. W. Burby, and Q. Tang. Approximation of nearly-periodic symplectic maps via structure-preserving neural networks. *Scientific Reports, Collection on "Physics-informed Machine Learning and its real-world applications"*, 2023a. doi: 10.1038/s41598-023-34862-w.

V. Duruisseaux, T. Duong, M. Leok, and N. Atanasov. Lie group forced variational integrator networks for learning and control of robot systems. *Extended version, arXiv preprint:* https://arxiv.org/pdf/2211.16006.pdf, 2023b.

J. Gallier and J. Quaintance. *Differential Geometry and Lie Groups: A Computational Perspective*. Geometry and Computing. Springer International Publishing, 2020. ISBN 9783030460402.

S. Greydanus, M. Dzamba, and J. Yosinski. Hamiltonian neural networks. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

L. Grüne and J. Pannek. *Nonlinear model predictive control*. Springer, 2017.

E. Hairer, C. Lubich, and G. Wanner. *Geometric Numerical Integration*, volume 31 of *Springer Series in Computational Mathematics*. Springer-Verlag, 2006.

A. Havens and G. Chowdhary. Forced variational integrator networks for prediction and control of mechanical systems. *arXiv preprint arXiv:2106.02973*, 2021.

D. Holm, T. Schmah, and C. Stoica. *Geometric Mechanics and Symmetry: From Finite to Infinite Dimensions*. Oxford Texts in Applied and Engineering Mathematics. OUP Oxford, 2009. ISBN 9780199212910.

P. Jin, Z. Zhang, A. Zhu, Y. Tang, and G. E. Karniadakis. SympNets: Intrinsic structure-preserving symplectic networks for identifying Hamiltonian systems. *Neural Networks*, 132(C), 12 2020. doi: 10.1016/j.neunet.2020.08.017.

D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2014.

S. Lall and M. West. Discrete variational Hamiltonian mechanics. *J. Phys. A*, 39(19):5509–5519, 2006.

T. Lee. Computational geometric mechanics and control of rigid bodies. *Ph.D. dissertation, University of Michigan*, 2008.

T. Lee, M. Leok, and N. H. McClamroch. *Global Formulations of Lagrangian and Hamiltonian Dynamics on Manifolds: A Geometric Approach to Modeling and Analysis*. Interaction of Mechanics and Mathematics. Springer International Publishing, 2017. ISBN 9783319569536.

B. Leimkuhler and S. Reich. *Simulating Hamiltonian Dynamics*, volume 14 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, 2004.

M. Leok and T. Shingel. Prolongation-collocation variational integrators. *IMA J. Numer. Anal.*, 32 (3):1194–1216, 2012.

M. Leok and J. Zhang. Discrete Hamiltonian variational integrators. *IMA Journal of Numerical Analysis*, 31(4):1497–1532, 2011.

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)*, 2019a.

M. Lutter, C. Ritter, and J. Peters. Deep Lagrangian networks: Using physics as model prior for deep learning. In *International Conference on Learning Representations*, 2019b.

D. Marco and F. Méhats. Symplectic learning for Hamiltonian neural networks. *arXiv preprint arXiv:2106.11753*, 2021.

J. E. Marsden and T. S. Ratiu. *Introduction to mechanics and symmetry*, volume 17 of *Texts in Applied Mathematics*. Springer-Verlag, New York, second edition, 1999.

J. E. Marsden and M. West. Discrete mechanics and variational integrators. *Acta Numer.*, 10: 357–514, 2001.

F. B. Mathiesen, B. Yang, and J. Hu. Hyperverlet: A symplectic hypersolver for Hamiltonian systems. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(4):4575–4582, June 2022. doi: 10.1609/aaai.v36i4.20381.

S. Ober-Blöbaum, O. Junge, and J. E. Marsden. Discrete mechanics and optimal control: An analysis. *ESAIM: Control, Optimisation and Calculus of Variations*, 17(2):322–352, 2011. doi: 10.1051/cocv/2010012.

C. Offen and S. Ober-Blöbaum. Symplectic integration of learned Hamiltonian systems. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 32(1):013122, 2022. doi: 10.1063/5.0065913.

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

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.

H. Poincaré. *Les méthodes nouvelles de la mécanique céleste, Volume 3*. Gauthier-Villars, Paris, 1899.

K. Rath, C. G. Albert, B. Bischl, and U. von Toussaint. Symplectic Gaussian process regression of maps in Hamiltonian systems. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 31(5): 053121, 2021. doi: 10.1063/5.0048129.

S. Sæmundsson, A. Terenin, K. Hofmann, and M. P. Deisenroth. Variational integrator networks for physically structured embeddings. In *AISTATS*, 2020.

S. Santos, M. Ekal, and R. Ventura. Symplectic momentum neural networks - using discrete variational mechanics as a prior in deep learning. In *Learning for Dynamics and Control Conference*, pages 584–595, 2022.

J. M. Schmitt and M. Leok. Properties of Hamiltonian variational integrators. *IMA Journal of Numerical Analysis*, 38(1):377–398, 03 2017.

J. M. Schmitt, T. Shingel, and M. Leok. Lagrangian and Hamiltonian Taylor variational integrators. *BIT Numerical Mathematics*, 58:457–488, 2018. doi: 10.1007/s10543-017-0690-9.

Y. Tassa, N. Mansard, and E. Todorov. Control-limited differential dynamic programming. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1168–1175, 2014.

R. Valperga, K. Webster, D. Turaev, V. Klein, and J. Lamb. Learning reversible symplectic dynamics. In *Learning for Dynamics and Control Conference*, volume 168, pages 906–916. PMLR, 2022.

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.

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

Y. D. Zhong, B. Dey, and A. Chakraborty. Symplectic ODE-Net: Learning Hamiltonian dynamics with control. In *International Conference on Learning Representations*, 2020a.

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*, 2020b.

Y. D. Zhong, B. Dey, and A. Chakraborty. Benchmarking energy-conserving neural networks for learning dynamics from data. In *Learning for Dynamics and Control*, volume 144, pages 1218–1229. PMLR, 2021.

Appendix A. Rigid-body kinematics on SE(3)

We present a brief introduction to rigid-body kinematics on SE(3), mostly extracted from Chapters 2, 6, 7 of Lee et al. (2017).

A rigid body is an idealization of a real mechanical system, defined as a collection of material particles such that the relative distance between any two particles in the body does not change (i.e, the body does not deform). The configuration of a rigid body is a representation of its position and attitude in 3-D space. The kinematics of a rigid body describe how its configuration changes under the influence of linear and angular velocity.

A.1. Rotational Rigid-Body Motion

If a rigid body has fixed position but can rotate arbitrarily in $\mathbb{R}^3$, then its configuration can be represented by a rotation matrix. Hence, the manifold of rotation matrices, SO(3), is the configuration manifold for rigid-body rotational motion. Since the dimension of SO(3) is three, rigid-body rotations have three degrees of freedom.

We use two Euclidean frames: an arbitrary reference frame and another frame fixed to the rigid body which rotates with along with it (with origin selected at the center of mass of the rigid body). A rotation matrix $R \in \text{SO}(3)$ is a linear transformation on $\mathbb{R}^3$ between the body-fixed and reference frames:

- if $v \in \mathbb{R}^3$ represents a vector in the body frame, then $Rv \in \mathbb{R}^3$ represents the same vector in the reference frame,
- if $v \in \mathbb{R}^3$ represents a vector in the reference frame, then $R^\top v \in \mathbb{R}^3$ represents the same vector in the body frame.

We can describe the rotation of the rigid body through the rotation of the body-fixed frame: the configuration of a rotating rigid body is the linear transformation that relates the representation of a vector in the body-fixed frame to its representation in the reference frame. Suppose that $R(t) \in \text{SO}(3)$ represents the rotational motion of a rigid body. Differentiating the orthogonality condition $R^\top R = \mathbb{I}_3$, we get $\dot{R}^\top R = -R^\top \dot{R}$ which implies that $R^\top \dot{R}$ remains skew-symmetric at all time. Thus, there exists a skew-symmetric matrix $\xi(t) \in \mathfrak{so}(3)$ such that $R^\top \dot{R} = \xi$, from which we can obtain the rotational kinematics:

$$\dot{R} = R\xi. \tag{19}$$

Using the isomorphism between the Lie algebra $\mathfrak{so}(3)$ and $\mathbb{R}^3$ given by $\xi = S(\omega)$ for $\omega \in \mathbb{R}^3$ and $\xi \in \mathfrak{so}(3)$, we can rewrite the rotational kinematics as

$$\dot{R} = RS(\omega), \tag{20}$$

where $\omega \in \mathbb{R}^3$ is referred to as the angular velocity vector of the rigid body expressed in the body frame. Thus, the rotational kinematics describe the rate of change $\dot{R}$ of the configuration in terms of the angular velocity $\omega \in \mathbb{R}^3$ represented in the body frame.

A.2. General Rigid-Body Motion

General rigid-body motion can be described by a combination of rotations and translations. As before, we use two inertial frames: a first arbitrary reference frame and another frame fixed to the

rigid body which translates and rotates with the rigid body (with origin usually selected at the center of mass of the rigid body). The trasnlational configuration of the rigid body characterizes the motion of the body-fixed frame origin and an be selected to lie in the configuration manifold $\mathbb{R}^3$.

The configuration manifold for a rigid-body that is simultaneously translating and rotating can be selected as the semidirect product of $\mathbb{R}^3$ and $\mathrm{SO}(3)$. We can represent the configuration via Therefore, we can $(R, x) \in \mathrm{SE}(3)$ in the sense that $R \in \mathrm{SO}(3)$ is the orientation and $x \in \mathbb{R}^3$ is the position of the body-fixed frame in the reference frame. Consequently, $\mathrm{SE}(3)$ can be viewed as the configuration manifold for general rigid-body motion.

As before, the rotational kinematics describe the rate of change $\dot{R}$ of the configuration in terms of the angular velocity vector $\omega \in \mathbb{R}^3$ of the rigid body represented in the body-fixed frame:

$$\dot{R} = RS(\omega). \tag{21}$$

Now, the translational velocity vector $v \in \mathbb{R}^3$ of the rigid body (i.e., of the origin of the body-fixed frame) is the time derivative of the position vector from the origin of the reference frame to the origin of the body-fixed frame. In the reference frame, the translational velocity vector $\dot{x} \in \mathbb{R}^3$ of the rigid body is

$$\dot{x} = Rv. \tag{22}$$

These are referred to as the translational kinematics of the rigid body. Altogether, we obtain the kinematics for general rigid-body motion:

$$\dot{R} = RS(\omega), \qquad \dot{x} = Rv. \tag{23}$$

Appendix B. Derivation of the forced variational integrator on SE(3)

In this appendix, we will derive the forced discrete Euler–Lagrange equations in Lagrangian form (equations (77)-(79)) and in Hamiltonian form (equations (10)-(14)) associated to the discrete Lagrangian L_d and discrete control forces $f_d^{\pm}$ on $\mathrm{SE}(3)$ presented in Section 4.2.

Consider a Lie group G with associated Lie algebra $\mathfrak{g} = T_eG$. In what follows, $\mathrm{L} : G \times G \to G$ denotes the left action on G, defined by $\mathrm{L}_q h = qh$ for all $q, h \in G$. The adjoint operator is denoted by $\mathrm{Ad}_q : \mathfrak{g} \to \mathfrak{g}$, and $\mathrm{Ad}_q^* : \mathfrak{g}^* \to \mathfrak{g}^*$ denotes the corresponding coadjoint. We refer the reader to (Marsden and Ratiu, 1999; Lee et al., 2017; Gallier and Quaintance, 2020) for a more detailed description of Lie group theory and mechanics on Lie groups.

Given a discrete Lagrangian $L_d(g_k, z_k)$ on the Lie group G, the forced discrete Euler–Lagrange equations are given by

$$g_{k+1} = g_k \star z_k, \tag{24}$$

$$\mathrm{T}_e^*\mathrm{L}_{z_{k-1}} D_2 L_{d_{k-1}} - \mathrm{Ad}^*_{z_k^{-1}} \left(\mathrm{T}_e^*\mathrm{L}_{z_k} D_2 L_{d_k}\right) + \mathrm{T}_e^*\mathrm{L}_{g_k} D_1 L_{d_k} + f_{d_k}^- + f_{d_{k-1}}^+ = 0, \tag{25}$$

where $L_{d_k} = L_d(g_k, z_k)$ and $f_{d_k}^{\pm} = f_d^{\pm}(g_k, g_{k+1}, u_k)$.

Using the discrete Legendre transform

$$\mu_k = \mathrm{Ad}^*_{z_k^{-1}} \left(\mathrm{T}_e^*\mathrm{L}_{z_k} D_2 L_{d_k}\right) - \mathrm{T}_e^*\mathrm{L}_{g_k} D_1 L_{d_k} - f_{d_k}^-, \tag{26}$$

we can rewrite the equations of motion in Hamiltonian form as

$$\mu_k = \mathrm{Ad}^*_{z_k^{-1}} \left(\mathrm{T}_e^*\mathrm{L}_{z_k} D_2 L_{d_k}\right) - \mathrm{T}_e^*\mathrm{L}_{g_k} D_1 L_{d_k} - f_{d_k}^-, \tag{27}$$

$$\mu_{k+1} = \mathrm{T}_e^*\mathrm{L}_{z_k} D_2 L_{d_k} + f_{d_k}^+ = \mathrm{Ad}_{z_k}^*(\mu_k + \mathrm{T}_e^*\mathrm{L}_{g_k} D_1 L_{d_k} + f_{d_k}^-) + f_{d_k}^+, \tag{28}$$

$$g_{k+1} = g_k \star z_k. \tag{29}$$

On SE(3), with $g_k = (x_k, R_k) \in \mathrm{SE}(3)$ and $z_k = (y_k, Z_k) \in \mathrm{SE}(3)$, the discrete kinematics equations $g_{k+1} = g_k \star z_k$ are given by

$$R_{k+1} = R_k Z_k \qquad \text{and} \qquad x_{k+1} = x_k + R_k y_k, \tag{30}$$

so that $\{(x_k, R_k)\}$ remains on SE(3). Using the kinematics equation $\dot{R} = RS(\omega)$, the matrix $S(\omega_k)$ can be approximated via

$$S(\omega_k) = R_k^\top \dot{R}_k \approx R_k^\top \frac{R_{k+1} - R_k}{h} = \frac{1}{h}(Z_k - \mathbb{I}_3). \tag{31}$$

With the discrete Lagrangian

$$\begin{aligned} L_d(x_k, R_k, y_k, Z_k) = \frac{m}{2h} y_k^\top y_k + \frac{1}{h}\mathrm{tr}\left([\mathbb{I}_3 - Z_k] J_d\right) \\ - (1-\alpha) h U(x_k, R_k) - \alpha h U(x_k + R_k y_k, R_k Z_k), \end{aligned} \tag{32}$$

it can be shown by proceeding as in (Lee, 2008) that the forced discrete Euler–Lagrange equations are given by

$$\frac{1}{h}(J_d Z_{k-1} - Z_{k-1}^\top J_d) - \frac{1}{h}(Z_k J_d - J_d Z_k^\top) + hS(\xi_k) + S(f_{d_k}^{R-}) + S(f_{d_{k-1}}^{R+}) = 0, \tag{33}$$

$$\frac{m}{h} R_k^\top (x_k - x_{k-1}) - \frac{m}{h} R_k^\top (x_{k+1} - x_k) - h R_k^\top \frac{\partial U_k}{\partial x_k} + f_{d_k}^{x-} + f_{d_{k-1}}^{x+} = 0, \tag{34}$$

$$R_{k+1} = R_k Z_k, \tag{35}$$

where $f_{d_k}^{x\pm}$ and $f_{d_k}^{R\pm}$ denote the x and R components of the discrete forces $f_{d_k}^{\pm}$.

This can be simplified into the forced discrete Euler–Lagrange equations

$$h^2 S(\xi_k) + hS(f_{d_k}^{R-}) + hS(f_{d_{k-1}}^{R+}) + (J_d Z_{k-1} - Z_{k-1}^\top J_d) = Z_k J_d - J_d Z_k^\top, \tag{36}$$

$$x_{k+1} = 2x_k - x_{k-1} - \frac{h^2}{m}\frac{\partial U_k}{\partial x_k} + \frac{h}{m} R_k (f_{d_k}^{x-} - f_{d_{k-1}}^{x+}), \tag{37}$$

$$R_{k+1} = R_k Z_k. \tag{38}$$

Using the discrete Legendre transforms

$$S(\pi_k) = \frac{1}{h}(Z_k J_d - J_d Z_k^\top) - (1-\alpha) hS(\xi_k) - S(f_{d_k}^{R-}), \tag{39}$$

$$\nu_k = \frac{m}{h} R_k^\top (x_{k+1} - x_k) + (1-\alpha) h R_k^\top \frac{\partial U_k}{\partial x_k} - f_{d_k}^{x-}, \tag{40}$$

we get

$$S(\pi_{k+1}) = \frac{1}{h}(J_d Z_k - Z_k^\top J_d) + \alpha h S(\xi_{k+1}) + S(f_{d_k}^{R+}), \tag{41}$$

$$\nu_{k+1} = \frac{m}{h} R_{k+1}^\top (x_{k+1} - x_k) - \alpha h R_{k+1}^\top \frac{\partial U_{k+1}}{\partial x_{k+1}} + f_{d_k}^{x+}. \tag{42}$$

With $\gamma = R\nu$, equation (42) can be rewritten as

$$\gamma_{k+1} = \frac{m}{h}(x_{k+1} - x_k) - \alpha h \frac{\partial U_{k+1}}{\partial x_{k+1}} + R_{k+1} f_{d_k}^{x+}. \tag{43}$$

Overall, we obtain the following implicit discrete equations of motion in Hamiltonian form:

$$S(\pi_k) = \frac{1}{h}(Z_k J_d - J_d Z_k^\top) - (1-\alpha) h S(\xi_k) - S(f_{d_k}^{R-}), \tag{44}$$

$$\gamma_k = \frac{m}{h}(x_{k+1} - x_k) + (1-\alpha) h \frac{\partial U_k}{\partial x_k} - R_k f_{d_k}^{x-}, \tag{45}$$

$$R_{k+1} = R_k Z_k, \tag{46}$$

$$S(\pi_{k+1}) = \frac{1}{h}(J_d Z_k - Z_k^\top J_d) + \alpha h S(\xi_{k+1}) + S(f_{d_k}^{R+}), \tag{47}$$

$$\gamma_{k+1} = \frac{m}{h}(x_{k+1} - x_k) - \alpha h \frac{\partial U_{k+1}}{\partial x_{k+1}} + R_{k+1} f_{d_k}^{x+}. \tag{48}$$

Equations (44) and (45) give

$$h S(\pi_k) + (1-\alpha) h^2 S(\xi_k) = Z_k J_d - J_d Z_k^\top - h S(f_{d_k}^{R-}), \tag{49}$$

$$x_{k+1} = x_k + \frac{h}{m}\gamma_k - (1-\alpha)\frac{h^2}{m}\frac{\partial U_k}{\partial x_k} - \frac{h}{m} R_k f_{d_k}^{x-}. \tag{50}$$

Equation (47) can be rewritten using equation (44) as:

$$S(\pi_{k+1}) = Z_k^\top S(\pi_k) Z_k + (1-\alpha) h Z_k^\top S(\xi_k) Z_k + \alpha h S(\xi_{k+1}) + Z_k^\top S(f_{d_k}^{R-}) Z_k + S(f_{d_k}^{R+}). \tag{51}$$

Since $Z^\top S(\eta) Z = S(Z^\top \eta)$ for any $Z \in \mathrm{SO}(3)$ and $\eta \in \mathfrak{so}(3)$, we get

$$\pi_{k+1} = Z_k^\top \pi_k + (1-\alpha) h Z_k^\top \xi_k + \alpha h \xi_{k+1} + Z_k^\top f_{d_k}^{R-} + f_{d_k}^{R+}. \tag{52}$$

Finally, equation (48) can be rewritten using equation (45) as

$$\gamma_{k+1} = \gamma_k - (1-\alpha) h \frac{\partial U_k}{\partial x_k} - \alpha h \frac{\partial U_{k+1}}{\partial x_{k+1}} + R_k f_{d_k}^{x-} + R_{k+1} f_{d_k}^{x+}. \tag{53}$$

Overall, this gives the forced variational integrator (10)-(14):

$$h S(\pi_k) + (1-\alpha) h^2 S(\xi_k) = Z_k J_d - J_d Z_k^\top - h S(f_{d_k}^{R-}), \tag{54}$$

$$R_{k+1} = R_k Z_k, \tag{55}$$

$$\pi_{k+1} = Z_k^\top \pi_k + (1-\alpha) h Z_k^\top \xi_k + \alpha h \xi_{k+1} + Z_k^\top f_{d_k}^{R-} + f_{d_k}^{R+}, \tag{56}$$

$$x_{k+1} = x_k + \frac{h}{m}\gamma_k - (1-\alpha)\frac{h^2}{m}\frac{\partial U_k}{\partial x_k} - \frac{h}{m} R_k f_{d_k}^{x-}, \tag{57}$$

$$\gamma_{k+1} = \gamma_k - (1-\alpha) h \frac{\partial U_k}{\partial x_k} - \alpha h \frac{\partial U_{k+1}}{\partial x_{k+1}} + R_k f_{d_k}^{x-} + R_{k+1} f_{d_k}^{x+}. \tag{58}$$

Appendix C. Transforming the equation $S(a) = ZJ_d - J_dZ^\top$

Plugging the Cayley transform

$$Z = \text{Cay}(\hat{\zeta}) \equiv (\mathbb{I}_3 + S(\zeta))(\mathbb{I}_3 - S(\zeta))^{-1}, \tag{59}$$

into the equation

$$S(a) = ZJ_d - J_dZ^\top, \tag{60}$$

and using the fact that $(\mathbb{I}_3 \pm S(\zeta))^\top = (\mathbb{I}_3 \mp S(\zeta))$ gives

$$S(a) = (\mathbb{I}_3 + S(\zeta))(\mathbb{I}_3 - S(\zeta))^{-1}J_d - J_d(\mathbb{I}_3 + S(\zeta))^{-1}(\mathbb{I}_3 - S(\zeta)). \tag{61}$$

Now, $(\mathbb{I}_3 \pm S(\zeta))$ and $(\mathbb{I}_3 \mp S(\zeta))^{-1}$ commute, so we can rewrite the previous equation as

$$S(a) = (\mathbb{I}_3 - S(\zeta))^{-1}(\mathbb{I}_3 + S(\zeta))J_d - J_d(\mathbb{I}_3 - S(\zeta))(\mathbb{I}_3 + S(\zeta))^{-1}. \tag{62}$$

Multiplying both sides of equation (62) on the left by $(\mathbb{I}_3 - S(\zeta))$ and on the right by $(\mathbb{I}_3 + S(\zeta))$ gives

$$(\mathbb{I}_3 - S(\zeta))S(a)(\mathbb{I}_3 + S(\zeta)) = (\mathbb{I}_3 + S(\zeta))J_d(\mathbb{I}_3 + S(\zeta)) - (\mathbb{I}_3 - S(\zeta))J_d(\mathbb{I}_3 - S(\zeta)), \tag{63}$$

which can be simplified into

$$S(a) - S(\zeta)S(a) + S(a)S(\zeta) - S(\zeta)S(a)S(\zeta) = 2S(\zeta)J_d + 2J_dS(\zeta). \tag{64}$$

Using $S(\zeta)J_d + J_dS(\zeta) = S(J\zeta)$ and the general formulas

$$-S(y)S(x) + S(x)S(y) = S(S(x)y), \qquad S(x)S(y)S(x) = -(y^\top x)S(x), \tag{65}$$

we can simplify equation (64) into

$$S(a) + S(S(a)\zeta) + (a^\top \zeta)S(\zeta) = 2S(J\zeta). \tag{66}$$

This can be rewritten in the desired vector form

$$a + a \times \zeta + (a^\top \zeta)\zeta - 2J\zeta = 0. \tag{67}$$

Appendix D. Implementation details

In this appendix, we provide additional details concerning the implementation of the LieFVINs for the planar pendulum on SO(3) and for the crazyflie quadrotor on SE(3). In particular, we detail the structure of the neural networks, the data generation process, and the training process.

To train the dynamics model with Algorithm Ia, we minimize the loss function

$$\mathcal{L}_{\text{Ia}}(\theta) = \sum_{i=1}^{N} \|x_1 - \tilde{x}_1\|^2 + \left\|\log\left(\tilde{R}_1 R_1^\top\right)^\vee\right\|^2 + \|v_1 - \tilde{v}_1\|^2 + \|\omega_1 - \tilde{\omega}_1\|^2, \tag{68}$$

while we use the following loss function for Algorithm Ib

$$\mathcal{L}_{\text{Ib}}(\theta) = \sum_{i=1}^{N} \|x_1 - \tilde{x}_1\|^2 + \|v_1 - \tilde{v}_1\|^2 + \|\omega_1 - \tilde{\omega}_1\|^2 + \left\| hS(J\omega_0) + hS(f_{d_0}^{R-}) + (1-\alpha)h^2 S(\xi_0) - J_d Z_0 + Z_0^\top J_d \right\|^2. \tag{69}$$

The network parameters θ are updated using Adam (Kingma and Ba, 2014), where the gradients $\partial\mathcal{L}/\partial\theta$ are calculated by back-propagation.

In the descriptions of the network architectures below, the first number is the input dimension while the last number is the output dimension. The hidden layers are listed in-between with their dimensions and activation functions.

D.1. Pendulum

We use neural networks to represent the inertial matrix $J(q) = L(q)L(q)^\top + \epsilon$, the potential energy $U(q)$ and the input gains $g(q)$ as follows:

- $L(q)$: 9 - 10 Tanh - 10 Tanh - 10 Linear - 6
- $U(q)$: 9 - 10 Tanh - 10 Tanh - 10 Linear - 1
- $g(q)$: 9 - 10 Tanh - 10 Tanh - 10 Linear - 3

The training data of the form $\{(\cos\varphi, \sin\varphi, \dot{\varphi})\}$ was collected from an OpenAI Gym environment, provided by (Zhong et al., 2020a). The control inputs are sampled in $[-3, 3]$ and applied to the planar pendulum for 10 time intervals of 0.02s to generate 512 state-control trajectories. The SO(3) LieFVIN, as described in Algorithm Ia with $\alpha = 0.5$, was trained with a fixed learning rate of 10^{-3} for 10000 iterations.

For comparison, we also learned the dynamics using a black-box model which is a multilayer perceptron MLP$(q, \dot{q}, u)$ with architecture [22 - 1000 Tanh - 1000 Tanh - 1000 Linear - 18].

To drive the pendulum from downward position $\varphi = 0$ to a stabilized upright position $\varphi^* = \pi$, $\dot{\varphi}^* = 0$, with input constraint $|u| \leq 20$, the running cost C_d and terminal cost Φ_d in the MPC problem are chosen to be

$$C_d(R_{\ell+k}, \omega_{\ell+k}, u_{\ell+k}) = \text{tr}(\mathbb{I}_3 - R^{*\top} R_{\ell+k}) + 0.1\|\omega_{\ell+k}\|^2 + 10^{-4}\|u_{\ell+k}\|^2, \tag{70}$$

$$\Phi_d(R_{\ell+k}, \omega_{\ell+k}, u_{\ell+k}) = \text{tr}(\mathbb{I}_3 - R^{*\top} R_{\ell+k}) + 0.1\|\omega_{\ell+k}\|^2 + 10^{-4}\|u_{\ell+k}\|^2. \tag{71}$$

*

D.2. Crazyflie Quadrotor

We use neural networks to represent the mass $m = r^2$, inertial matrix $J(q) = LL^\top + \epsilon$, the potential energy $U(q)$ and the input gains $g(q) = \begin{bmatrix} g_x(q) & g_R(q) \end{bmatrix}$ as follows:

- r: 1D pytorch parameter
- L: 3×3 upper-triangular parameter matrix

- $U(q)$: 9 - 10 Tanh - 10 Tanh- 10 Tanh - 10 Linear - 1
- $g(q)$: 9 - 10 Tanh - 10 Tanh- 10 Tanh - 10 Linear - 24

To obtain the training data, the quadrotor was controlled from a random starting point to 36 different desired poses using a PID controller, yielding 36 4-second trajectories. The trajectories were used to generate a dataset of $N = 2700$ position-velocity updates $\{(q_0, \dot{q}_0, u_0) \mapsto (q_1, \dot{q}_1)\}$ with time step 0.02s. The SE(3) LieFVIN, as described in Algorithm Ib with $\alpha = 0.5$, was trained with a decaying learning rate initialized at 5×10^{-3} for 20000 iterations.

To track a diamond-shaped trajectory using the model learnt by the LieFVIN, with control input constraints $0 \leq f \leq 0.595,\ |\tau| \leq 10^{-3}[5.9\ \ 5.9\ \ 7.4]^\top$, the running cost C_d and terminal cost Φ_d in the MPC problem are chosen to be

$$\begin{aligned} C_d(x_{\ell+k}, R_{\ell+k}, v_{\ell+k}, \omega_{\ell+k}, u_{\ell+k}) \ &= \ 1.2\|x_{\ell+k}\|^2 + 10^{-5}\ \mathrm{tr}(\mathbb{I}_3 - R_{\ell+k}) + 1.2\|v_{\ell+k}\|^2 \\ &\quad + 10^{-4}\|\omega_{\ell+k}\|^2 + 10^{-6}\|u_{\ell+k}\|^2, \end{aligned} \tag{72}$$

$$\begin{aligned} \Phi_d(x_{\ell+k}, R_{\ell+k}, v_{\ell+k}, \omega_{\ell+k}, u_{\ell+k}) \ &= \ 1.2\|x_{\ell+k}\|^2 + 10^{-5}\ \mathrm{tr}(\mathbb{I}_3 - R_{\ell+k}) + 1.2\|v_{\ell+k}\|^2 \\ &\quad + 10^{-4}\|\omega_{\ell+k}\|^2 + 10^{-6}\|u_{\ell+k}\|^2. \end{aligned} \tag{73}$$

Appendix E. Learning and controlling Lagrangian systems from position data

E.1. Problem Statement

We now consider the problem of learning controlled Lagrangian dynamics only from position data: given a position-only dataset of trajectories for a Lagrangian system, we wish to infer the update map that generates these trajectories, while preserving the symplectic structure underlying the dynamical system and constraining the updates to the Lie group on which it evolves. More precisely, we wish to solve the following problem:

Problem 2 *Given a dataset of position-only updates* $\left\{\left(q_0^{(i)}, q_1^{(i)}, u_0^{(i)}, u_1^{(i)}\right) \mapsto q_2^{(i)}\right\}_{i=1}^N$ *for a controlled Lagrangian system evolving on the Lie group* $\mathcal{Q}$*, we wish to find a symplectic mapping* $\Psi : \mathcal{Q} \times \mathcal{Q} \times \mathcal{U} \times \mathcal{U} \to \mathcal{Q}$ *which minimizes*

$$\sum_{i=1}^{N} \mathcal{D}_{\mathcal{Q}}\left(q_2^{(i)}, \Psi\left(q_0^{(i)}, q_1^{(i)}, u_0^{(i)}, u_1^{(i)}\right)\right), \tag{74}$$

where $\mathcal{D}_{\mathcal{Q}}$ *is a distance metric on* $\mathcal{Q}$.

E.2. Forced Variational Integrator in Lagrangian Form

As before, we choose the discrete Lagrangian

$$\begin{aligned} L_d(x_k, R_k, y_k, Z_k) = \frac{m}{2h} y_k^\top y_k + \frac{1}{h}\mathrm{tr}\left([\mathbb{I}_3 - Z_k]J_d\right) \\ - (1-\alpha)hU(x_k, R_k) - \alpha h U(x_k + R_k y_k, R_k Z_k), \end{aligned} \tag{75}$$

where $\alpha \in [0,1]$ and $J_d = \frac{1}{2}\text{tr}(J)\mathbb{I}_3 - J$. We also define U_k and ξ_k via

$$U_k = U(x_k, R_k) \quad \text{and} \quad S(\xi_k) = \frac{\partial U_k}{\partial R_k}^\top R_k - R_k^\top \frac{\partial U_k}{\partial R_k}. \tag{76}$$

It is shown in Appendix B that the forced discrete Euler–Lagrange equations associated to the discrete Lagrangian (75) and the discrete control forces $f_{d_k}^\pm \equiv f_d^\pm(x_k, R_k, u_k)$ are given by

$$h^2 S(\xi_k) + hS(f_{d_k}^{R-}) + hS(f_{d_{k-1}}^{R+}) + (J_d Z_{k-1} - Z_{k-1}^\top J_d) = Z_k J_d - J_d Z_k^\top, \tag{77}$$

$$x_{k+1} = 2x_k - x_{k-1} - \frac{h^2}{m}\frac{\partial U_k}{\partial x_k} + \frac{h}{m} R_k (f_{d_k}^{x-} - f_{d_{k-1}}^{x+}), \tag{78}$$

$$R_{k+1} = R_k Z_k. \tag{79}$$

Since $(J_d Z_{k-1} - Z_{k-1}^\top J_d) \in \mathfrak{so}(3)$, equation (77) can be rewritten as $S(a) = Z_k J_d - J_d Z_k^\top$ with

$$a = h^2 \xi_k + h f_{d_k}^{R-} + h f_{d_{k-1}}^{R+} + S^{-1}(J_d Z_{k-1} - Z_{k-1}^\top J_d). \tag{80}$$

Given $(x_{k-1}, x_k, R_{k-1}, R_k, u_{k-1}, u_k)$, we first solve $S(a) = ZJ_d - J_d Z^\top$ for $Z = Z_k$ as outlined in Remark 1, and then get $R_{k+1} = R_k Z_k$. We then update x_{k+1} using equation (78).

E.3. Lie Group Forced Variational Integrator Networks (LieFVINs)

We now describe the construction of Lie group Forced Variational Integrator Networks for the forced variational integrator on SE(3) presented in Appendix E.2, in the case where only position data is available. The LieFVIN is based on the discrete forced Euler–Lagrange equations (77)-(79). As before, the main idea is to parametrize the updates of the forced variational integrator and match them with the observed updates.

We parametrize m, $f_d^\pm$ and U as neural networks, and the matrix J is a symmetric positive-definite matrix-valued function of (x, R) constructed via a Cholesky decomposition $J = LL^\top$ for a lower-triangular matrix L implemented as a neural network. We can also get $J_d = \frac{1}{2}\text{tr}(J)\mathbb{I}_3 - J$. To deal with the implicit nature of equation (77), we propose two algorithms, based either on an explicit iterative solver or by penalizing deviations away from equation (77):

Algorithm IIa. Given $(x_0, x_1, R_0, R_1, u_0, u_1) \mapsto (x_2, R_2)$ data, minimize discrepancies between the observed (x_2, R_2) pairs and the predicted $(\tilde{x}_2, \tilde{R}_2)$ pairs obtained as follows:

For each $(x_0, x_1, R_0, R_1, u_0, u_1)$ data tuple,

1. Get $f_{d_0}^{R\pm}$, $f_{d_1}^{R\pm}$, $f_{d_0}^{x\pm}$,$f_{d_1}^{x\pm}$ from $(x_0, x_1, R_0, R_1, u_0, u_1)$, and $S(\xi_1) = \frac{\partial U_1}{\partial R_1}^\top R_1 - R_1^\top \frac{\partial U_1}{\partial R_1}$
2. Get $\tilde{R}_2 = R_1 \text{Cay}(\zeta)$ where ζ is obtained using a few steps of Newton's method to solve the vector equation (15) equivalent to
$$h^2 S(\xi_1) + hS(f_{d_1}^{R-} + f_{d_0}^{R+}) + (J_d Z_0 - Z_0^\top J_d) = ZJ_d - J_d Z$$
3. Compute $\tilde{x}_2 = 2x_1 - x_0 - \frac{h^2}{m}\frac{\partial U_1}{\partial x_1} + \frac{h}{m} R_1 (f_{d_1}^{x-} + f_{d_0}^{x})$

Algorithm IIb. Given $(x_0, x_1, R_0, R_1, u_0, u_1) \mapsto (x_2, R_2)$ data, minimize

- Discrepancies between observed x_2 and predicted $\tilde{x}_2 = 2x_1 - x_0 - \frac{h^2}{m}\frac{\partial U_1}{\partial x_1} + \frac{h}{m}R_1(f_{d_1}^{x-} + f_{d_0}^{x})$
- Deviations away from the equation

$$J_d(R_0^\top R_1 + R_2^\top R_1) - (R_1^\top R_0 + R_1^\top R_2)J_d + h^2\left(\frac{\partial U_1}{\partial R_1}^\top R_1 - R_1^\top \frac{\partial U_1}{\partial R_1}\right) + hS(f_{d_1}^{R-} + f_{d_0}^{R+}) = 0$$

This general strategy extends to any other Lie group integrator. In particular, LieFVINs on $\mathrm{SO}(3)$ can be obtained from the algorithms above as the special case where x is constant, in which case we can disregard all the variables and operations in green. Lie group variational integrator networks without forces (**LieVINs**) can be obtained by setting $f_{d_0}^{R\pm} = f_{d_0}^{x\pm} = 0$. Note that the strategy behind Algorithm IIa enforces the structure of the system in a stronger way than in Algorithm IIb. However, for certain Lie groups and variational integrators, it might not be practical to use Newton's method to solve for the implicit updates, in which case Algorithm IIb is preferred.

When combined with MPC as described in Section 4.4, the initial conditions $(q_{\ell-1}, q_\ell)$ for the optimal control problems can be obtained either from the position estimates $(\tilde{q}_{\ell-1}, \tilde{q}_\ell)$ or from (position,velocity) estimates $(\tilde{q}_\ell, \dot{\tilde{q}}_\ell)$ with finite difference approximations. As before, the Lie group constraints for the system do not need to be added as path constraints since they are automatically satisfied to (almost) machine precision, by the design of the LieFVINs.