Deep Discrete-Time Lagrangian Mechanics

Anonymous authors
Paper under double-blind review

Abstract

Ensuring physical laws such as the conservation law of energy is important for physics simulations. Recent studies demonstrated that neural networks successfully learn physical dynamics and conserve the system energy using symplectic integrators or a discrete gradient method. While their approaches depend on the canonical momentum or velocity, measuring an accurate velocity is troublesome because it is often measured by a linear interpolation of two points. Without an accurate velocity, a learned dynamics may be greatly different from the teacher system. In this paper, we propose a neural network based on discrete-time Lagrangian mechanics. The proposed approach learns the physical dynamics only from the position data and conserves the energy strictly in discrete-time by using a discrete gradient method. Experimental results on simulated physical systems demonstrated that our approach learns the energy surface in the state space accurately and conserves the modeled system energy strictly.

1 Introduction

Reliable physics simulations should ensure underlying physical laws such as the conservation law of energy. Recent studies demonstrated that neural networks learn a physical dynamics associated with the conservation law of energy. Hamiltonian neural networks (HNN) and Lagrangian neural networks (LNN) learn Hamiltonian and Lagrangian mechanics, ensuring the energy conservation in continuous time (Greydanus et al., 2019; Cranmer et al., 2020). However, the energy is no longer conserved after numerical integrators discretize the time. Symplectic recurrent neural networks (SRNN) and variational integrator networks (VIN) employed symplectic integrators, which conserve a modified energy in discrete time (Chen et al., 2020; Saemundsson et al., 2020). A discrete gradient method conserves the system energy strictly in discrete time (Hairer et al., 2006). Matsubara et al. (2020) proposed the automatic discrete differentiation algorithm, which makes a discrete gradient method applicable to neural networks, named as DGNet. See Table 1 for comparison.

As the system state, previous approaches used either velocity or canonical momentum as well as the position. Measuring an accurate velocity is troublesome because it is often measured by a linear interpolation of two points and suffers from an interpolation error. The Verlet method is a symplectic integrator applicable to Lagrangian mechanics and depends only on the position (Hairer et al., 2006). However, no method that conserves the system energy strictly in discrete time is available when only the position is available.

In this paper, we propose deep discrete-time Lagrangian mechanics, which extends the discrete gradient method for deep learning to Lagrangian mechanics described only with the position, and conserves the system energy in discrete time. We evaluate our approach on several simulated physical systems and demonstrate that it learns the energy surface in the state space more accurately than comparative approaches and conserves the system energy strictly.

2 Background

In this work, we focus on a physical system associated with the potential energy V, expressed as a function of the position $\boldsymbol{q}$, and the kinetic energy T, expressed as a function of the velocity $\dot{\boldsymbol{q}}$ or canonical momentum $\boldsymbol{p}$. Specifically, with a symmetric mass matrix $\boldsymbol{M}$,

$$T = \tfrac{1}{2}\dot{\boldsymbol{q}}^\top \boldsymbol{M}\dot{\boldsymbol{q}} = \tfrac{1}{2}\boldsymbol{p}^\top \boldsymbol{M}^{-1}\boldsymbol{p}. \tag{1}$$

Table 1: Comparison of Our Approach against Related Ones

	HNN	LNN	SRNN	VIN	DGNet	Ours
Learn Lagrangian mechanics		✓		✓		✓
Learn from finite difference			✓	✓	✓	✓
Conserve energy strictly					✓	✓
Learn from only position						✓

Hamiltonian Mechanics. In Hamiltonian mechanics, a system has a state $\boldsymbol{u} = (\boldsymbol{q}\ \ \boldsymbol{p})^\top$ and an energy function $\mathcal{H}$, called the Hamiltonian. The time evolution follows Hamilton's canonical equation

$$\frac{\mathrm{d}\boldsymbol{q}}{\mathrm{d}t} = \nabla_{\boldsymbol{p}}\mathcal{H}, \quad \frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}t} = -\nabla_{\boldsymbol{q}}\mathcal{H}, \tag{2}$$

which ensures the conservation law of energy. The chain-rule provides a simple proof as

$$\frac{\mathrm{d}\mathcal{H}}{\mathrm{d}t} = \frac{\partial\mathcal{H}}{\partial\boldsymbol{u}}\frac{\partial\boldsymbol{u}}{\partial t} = (\nabla_{\boldsymbol{q}}\mathcal{H}\ \ \nabla_{\boldsymbol{p}}\mathcal{H})(\nabla_{\boldsymbol{p}}\mathcal{H}\ \ -\nabla_{\boldsymbol{q}}\mathcal{H})^\top = 0. \tag{3}$$

HNN approximates the Hamiltonian $\mathcal{H}$ from the data using a neural network (Greydanus et al., 2019). When $\mathcal{H} = T + V$ and Eq. (1) is given, Eq. (2) is rewritten as follows (Zhong et al., 2020).

$$\frac{\mathrm{d}\boldsymbol{q}}{\mathrm{d}t} = \boldsymbol{M}^{-1}\boldsymbol{p}, \quad \frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}t} = -\nabla_{\boldsymbol{q}}V(\boldsymbol{q}). \tag{4}$$

Lagrangian Mechanics. In Lagrangian mechanics, a system has a state $\boldsymbol{u} = (\boldsymbol{q}\ \ \dot{\boldsymbol{q}})^\top$ and the Lagrangian $\mathcal{L} = T - V$. The Euler–Lagrangian equation $\frac{\mathrm{d}}{\mathrm{d}t}\nabla_{\dot{\boldsymbol{q}}}\mathcal{L} = \nabla_{\boldsymbol{q}}\mathcal{L}$ holds, and it admits the conservation law of energy. LNN approximates the Lagrangian $\mathcal{L}$ (Cranmer et al., 2020), and Deep Lagrangian networks (DeLaN) approximates the state-dependent mass-matrix $\boldsymbol{M}(\boldsymbol{q})$ (Lutter et al., 2019). Given Eq. (1), Newton's equation of motions is obtained as

$$\boldsymbol{M}\ddot{\boldsymbol{q}} = -\nabla_{\boldsymbol{q}}V(\boldsymbol{q}). \tag{5}$$

Symplectic Integrator. With a numerical integrator for a computer simulation, HNN and LNN do not conserve the system energy in general. Symplectic integrators approximate the underlying symplectic structure and conserve a modified energy in discrete time, but it is not equal to the system energy. SRNN employed the leapfrog integrator for Hamiltonian mechanics (Chen et al., 2020), and VIN employed the variational integrator for Lagrangian mechanics (Saemundsson et al., 2020).

Discrete Gradient Method. For conserving the system energy strictly, a discrete gradient method has been employed (Hairer et al., 2006; Furihata & Matsuo, 2010; Celledoni et al., 2012). A vector $\overline{\nabla}H$ that satisfies the following condition is a discrete gradient of a function H;

$$H(\boldsymbol{u}) - H(\boldsymbol{v}) = \overline{\nabla}H(\boldsymbol{u}, \boldsymbol{v}) \cdot (\boldsymbol{u} - \boldsymbol{v}),\ \overline{\nabla}H(\boldsymbol{u}, \boldsymbol{u}) = \nabla H(\boldsymbol{u}), \tag{6}$$

where $\cdot$ denotes an inner product. The first condition indicates a discrete-time counterpart of the chain-rule, and the second condition ensures the consistency with the continuous-time chain-rule. With a discrete gradient $\overline{\nabla}\mathcal{H}$ of the Hamiltonian $\mathcal{H}$, a discrete expression of Eq. (2) is given by

$$\frac{\boldsymbol{q}^{(n+1)} - \boldsymbol{q}^{(n)}}{\Delta t} = \overline{\nabla}_{\boldsymbol{p}}\mathcal{H}(\boldsymbol{u}^{(n+1)}, \boldsymbol{u}^{(n)}),\ \frac{\boldsymbol{p}^{(n+1)} - \boldsymbol{p}^{(n)}}{\Delta t} = -\overline{\nabla}_{\boldsymbol{q}}\mathcal{H}(\boldsymbol{u}^{(n+1)}, \boldsymbol{u}^{(n)}), \tag{7}$$

where the superscription $^{(n)}$ denotes the n-th time step, and Δt denotes the time step size. Thanks to the discrete-time chain-rule, a discrete-time version of Eq. (3) holds, and it ensures the conservation law of energy in discrete time. Matsubara et al. (2020) proposed the automatic discrete differential (ADD) algorithm, which obtains a discrete gradient of an arbitrary computational graph. With the algorithm, a neural network named DGNet learns discrete-time dynamics of a Hamiltonian mechanics and conserves the system energy strictly in discrete time. However, this approach is not directly applicable to Lagrangian mechanics.

3 METHODS

In this section, we propose a discrete gradient method for Lagrangian mechanics. We first begin with the discrete-time expression of Eq. (4). Since the mass matrix $\boldsymbol{M}$ is supposed symmetric, the finite difference of the potential energy $T(\boldsymbol{p})$ between n-th and $n+1$-st time steps is

$$T(\boldsymbol{p}^{(n+1)}) - T(\boldsymbol{p}^{(n)}) = \tfrac{1}{2}\boldsymbol{M}^{-1}(\boldsymbol{p}^{(n+1)} + \boldsymbol{p}^{(n)}) \cdot (\boldsymbol{p}^{(n+1)} - \boldsymbol{p}^{(n)}). \tag{8}$$

Since the gradient of the kinetic energy $T(\boldsymbol{p})$ is $\nabla T(\boldsymbol{p}) = \boldsymbol{M}^{-1}\boldsymbol{p} = \frac{1}{2}\boldsymbol{M}^{-1}(\boldsymbol{p}+\boldsymbol{p})$, $\overline{\nabla} T(\boldsymbol{p}^{(n)},\boldsymbol{p}^{(n+1)}) = \frac{1}{2}\boldsymbol{M}^{-1}(\boldsymbol{p}^{(n+1)}+\boldsymbol{p}^{(n)})$ satisfies the definition of a discrete gradient in Eq. (6). Then, the discrete-time expression of Eq. (4) can be written as

$$\frac{\boldsymbol{q}^{(n+1)}-\boldsymbol{q}^{(n)}}{\Delta t} = \frac{1}{2}\boldsymbol{M}^{-1}(\boldsymbol{p}^{(n+1)}+\boldsymbol{p}^{(n)}),\ \frac{\boldsymbol{p}^{(n+1)}-\boldsymbol{p}^{(n)}}{\Delta t} = -\overline{\nabla}_{\boldsymbol{q}} V(\boldsymbol{q}^{(n+1)},\boldsymbol{q}^{(n)}). \tag{9}$$

By transforming Eq. (9), we obtain

$$\boldsymbol{p}^{(n+1)} = -\boldsymbol{p}^{(n)} + 2\boldsymbol{M}\frac{\boldsymbol{q}^{(n+1)}-\boldsymbol{q}^{(n)}}{\Delta t},\ \boldsymbol{p}^{(n+1)} = \boldsymbol{p}^{(n)} - \Delta t \overline{\nabla}_{\boldsymbol{q}} V(\boldsymbol{q}^{(n+1)},\boldsymbol{q}^{(n)}). \tag{10}$$

Then, the canonical momentum $\boldsymbol{p}^{(n+1)}$ is rewritten as

$$\boldsymbol{p}^{(n+1)} = \boldsymbol{M}\frac{\boldsymbol{q}^{(n+1)}-\boldsymbol{q}^{(n)}}{\Delta t} - \frac{\Delta t}{2}\overline{\nabla}_{\boldsymbol{q}} V(\boldsymbol{q}^{(n+1)},\boldsymbol{q}^{(n)}) \tag{11}$$

By substituting $\boldsymbol{p}^{(n)}$ and $\boldsymbol{p}^{(n+1)}$ into Eq. (9), we eliminate the canonical momentum $\boldsymbol{p}$ and obtain a numerical scheme only with the position $\boldsymbol{q}$ as follows.

$$\boldsymbol{M}\frac{\boldsymbol{q}^{(n+1)}-2\boldsymbol{q}^{(n)}+\boldsymbol{q}^{(n-1)}}{(\Delta t)^2} = -\frac{1}{2}(\overline{\nabla}_{\boldsymbol{q}} V(\boldsymbol{q}^{(n+1)},\boldsymbol{q}^{(n)}) + \overline{\nabla}_{\boldsymbol{q}} V(\boldsymbol{q}^{(n)},\boldsymbol{q}^{(n-1)})). \tag{12}$$

Since Eq. (12) converges to Eq. (5) as the time step size vanishes, Eq. (12) is a discrete-time expression of Eq. (5), and therefore, Eq. (12) is considered as a discrete-time Lagrangian mechanics. We can estimate a next position $\boldsymbol{q}^{(n+1)}$ from the current and last positions $\boldsymbol{q}^{(n)}$ and $\boldsymbol{q}^{(n-1)}$ by implicitly solving Eq. (12).

We parameterized the potential energy $V(\boldsymbol{q})$ by a neural network and obtained its discrete gradient $\overline{\nabla}_{\boldsymbol{q}} V$ by the ADD algorithm (Matsubara et al., 2020). At the training phase, we minimized the mean squared error between the left- and right-hand sides of Eq. (12) for each time step n. Since the positions $\boldsymbol{q}$ at times $n-1$, n, and $n+1$ are known, the ADD algorithm was performed only once to obtain each of the discrete gradients $\overline{\nabla}_{\boldsymbol{q}} V(\boldsymbol{q}^{(n+1)},\boldsymbol{q}^{(n)})$ and $\overline{\nabla}_{\boldsymbol{q}} V(\boldsymbol{q}^{(n)},\boldsymbol{q}^{(n-1)})$. One can obtain the system energy $T+V$ at n-th time step since the potential energy V is modeled by a neural network, and the kinetic energy T is obtained by substituting Eq. (11) into Eq. (1).

4 Experiments and Results

Experiments. For evaluation, we employed physical systems examined by Matsubara et al. (2020), namely a mass-spring system, a pendulum system, and a 2-body system. After training, our approach solved an initial value problem and evaluated the long-term prediction in terms of the mean squared error. We also employed the leapfrog integrator, which was used by SRNN (Chen et al., 2020). After a deformation similar to that of our approach, the leapfrog integrator is expressed as

$$\boldsymbol{M}\frac{\boldsymbol{q}^{(n+1)}-2\boldsymbol{q}^{(n)}+\boldsymbol{q}^{(n-1)}}{(\Delta t)^2} = -\nabla_{\boldsymbol{q}} V(\boldsymbol{q}^{(n)}). \tag{13}$$

This scheme is sometimes called the Verlet method (Hairer et al., 2006), and it is also equivalent to the variational integrator, used by VIN (Saemundsson et al., 2020). We also employed the symplectic Euler method, and the Euler method after similar modifications. The details of the experiments are summarized in Appendix A.

Results. Table 2 summarizes the quantitative results averaged over 15 trials. The top two panels of Fig. 1 show the predicted trajectories of states and energies from the first two positions; only our approach and the leapfrog integrator are depicted for visibility. Both methods predicted the state at a similar level. For the mass-spring and pendulum systems, the proposed approach conserved the energy accurately, implying that it learned the energy surface in the state space. With the leapfrog integrator, the energy oscillates over a wide range because of the gap between the true energy and modified energy. The bottom panel of Fig. 1 and Table 3 summarize the variances of the modeled energy $\mathcal{H} = T+V$. The variance is significantly smaller with the proposed approach. Thanks to the discrete gradient, the proposed approach conserved the energy that the neural network modeled, in other words, the proposed approach provides a lower numerical error in the energy. In contrast, the numerical error of the leapfrog integrator is more significant than the modeling error.

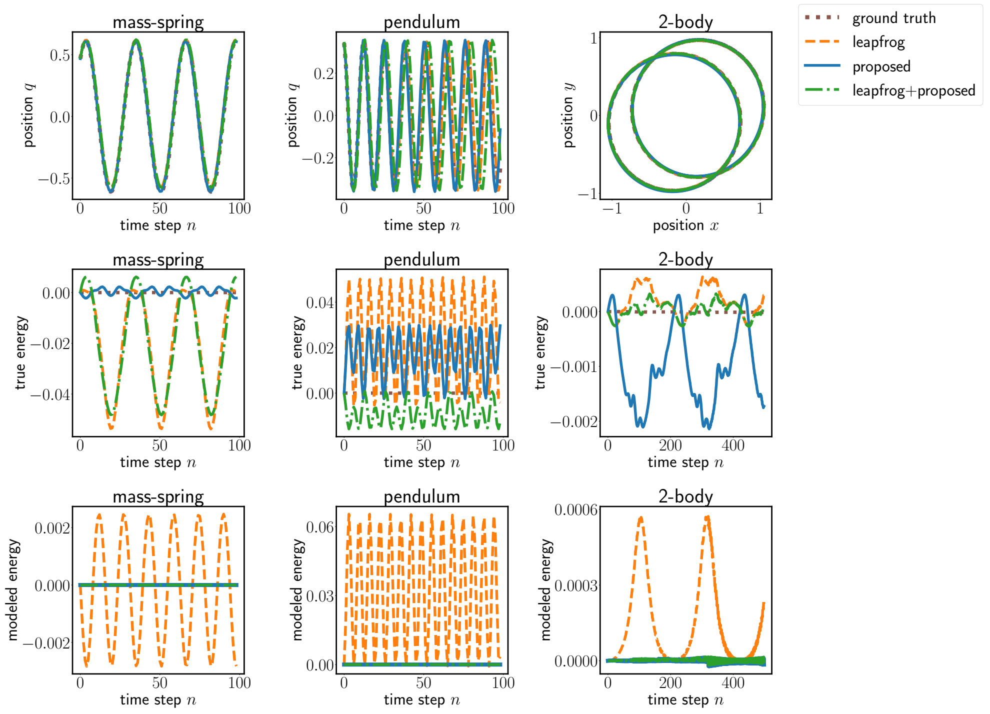

Figure 1: Results. (top) Position $\boldsymbol{q}$. (center) Energy. (bottom) Energy modeled by each model.

Table 2: Quantitative results across all three tasks.

Model	Mass-Spring		Pendulum		2-Body	
	Position q	Energy	Position q	Energy	Position q	Energy
Euler	4.48×10^{0}	1.23×10^{1}	2.96×10^{2}	8.54×10^{1}	3.71×10^{-1}	8.64×10^{-4}
symplectic Euler	7.09×10^{-2}	1.67×10^{-3}	**4.89×10^{-3}**	7.45×10^{-2}	**1.65×10^{-4}**	2.77×10^{-7}
leapfrog	7.09×10^{-2}	6.57×10^{-4}	**4.89×10^{-3}**	2.04×10^{-2}	**1.65×10^{-4}**	**1.14×10^{-7}**
proposed	**6.20×10^{-2}**	**4.52×10^{-5}**	5.43×10^{-3}	**2.20×10^{-3}**	7.31×10^{-3}	3.00×10^{-6}
leapfrog+proposed	8.30×10^{-2}	6.19×10^{-4}	1.47×10^{-1}	**8.38×10^{-4}**	**1.59×10^{-4}**	**1.01×10^{-7}**

Table 3: Modeled energy.

Model	Mass-Spring	Pendulum	2-Body
Euler	3.84×10^{-1}	3.85×10^{1}	1.35×10^{-4}
symplectic Euler	1.06×10^{-3}	7.15×10^{-2}	1.70×10^{-7}
leapfrog	3.10×10^{-5}	1.15×10^{-2}	2.78×10^{-9}
proposed	**2.11×10^{-12}**	**4.58×10^{-11}**	**2.42×10^{-11}**

For the 2-body system, the proposed approach suffered from a larger prediction error, which was caused by a modeling error. We further examined an approach named "leapfrog+proposed", which used the leapfrog integrator for training and the proposed approach for prediction, and found that it predicted the state and energy more accurately. Hence, the proposed approach provides a lower numerical error, but there exists room for improvement in a training scheme.

5 Conclusion

In this paper, we proposed the discrete-time Lagrangian mechanics, which is based on the discrete gradient method and implemented on a neural network. The proposed approach learns a certain kind of Lagrangian mechanics only from the position data, and it ensures the conservation law of energy strictly in discrete time. Future works include a state-dependent mass matrix $\boldsymbol{M}(\boldsymbol{q})$ and a training scheme based on deep equilibrium model (Bai et al., 2019).

References

S. Bai, J. Z. Kolter, and V. Koltun. Deep Equilibrium Models. In *Advances in Neural Information Processing Systems*, pp. 1–12, 2019.

E. Celledoni, V. Grimm, R. I. McLachlan, D. I. McLaren, D. O'Neale, B. Owren, and G. R.W. Quispel. Preserving energy resp. dissipation in numerical PDEs using the "Average Vector Field" method. *Journal of Computational Physics*, 231(20):6770–6789, 2012.

Z. Chen, J. Zhang, M. Arjovsky, and L. Bottou. Symplectic Recurrent Neural Networks. In *International Conference on Learning Representations*, pp. 1–23, 2020.

M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel, and S. Ho. Lagrangian Neural Networks. *ICLR 2020 Deep Differential Equations Workshop*, pp. 1–9, 2020.

D. Furihata and T. Matsuo. *Discrete Variational Derivative Method: A Structure-Preserving Numerical Method for Partial Differential Equations*. Chapman and Hall/CRC, 2010.

S. Greydanus, M. Dzamba, and J. Yosinski. Hamiltonian Neural Networks. In *Advances in Neural Information Processing Systems*, pp. 1–16, 2019.

E. Hairer, C. Lubich, and G. Wanner. *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*, volume 31 of *Springer Series in Computational Mathematics*. Springer, 2006.

D. P. Kingma and J. L. Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, pp. 1–15, 2015.

M. Lutter, C. Ritter, and J. Peters. Deep Lagrangian networks: Using physics as model prior for deep learning. In *International Conference on Learning Representations*, pp. 1–17, 2019.

T. Matsubara, A. Ishikawa, and T. Yaguchi. Deep energy-based modeling of discrete-time physics. In *Advances in Neural Information Processing Systems*, pp. 1–24, 2020.

S. Saemundsson, A. Terenin, K. Hofmann, and M. P. Deisenroth. Variational Integrator Networks for Physically Meaningful Embeddings. In *Artificial Intelligence and Statistics*, pp. 1–11, 2020.

Y. D. Zhong, B. Dey, and A. Chakraborty. Symplectic ODE-Net: Learning Hamiltonian Dynamics with Control. In *International Conference on Learning Representations*, pp. 1–17, 2020.

Supplementary Materials

A Details of Experiments

Comparative Models. We employed the leapfrog integrator, the symplectic Euler method, and the Euler method as the comparative models in Section 4. After a certain modification, these methods are applicable to datasets composed only of the positions $\boldsymbol{q}$. We provide details of the modification here. Given the state $\boldsymbol{u}^{(n)} = (\boldsymbol{q}^{(n)}, \dot{\boldsymbol{q}}^{(n)})$, the leapfrog integrator computes successive estimates $(\boldsymbol{q}^{(n+1)}, \dot{\boldsymbol{q}}^{(n+1)})$ as

$$\begin{aligned} \dot{\boldsymbol{q}}^{(n+\frac{1}{2})} &= \dot{\boldsymbol{q}}^{(n)} - \frac{1}{2}\Delta t \boldsymbol{M}^{-1}\nabla_{\boldsymbol{q}} V(\boldsymbol{q}^{(n)}), \\ \boldsymbol{q}^{(n+1)} &= \boldsymbol{q}^{(n)} + \Delta t \dot{\boldsymbol{q}}^{(n+\frac{1}{2})}, \\ \dot{\boldsymbol{q}}^{(n+1)} &= \dot{\boldsymbol{q}}^{(n+\frac{1}{2})} - \frac{1}{2}\Delta t \boldsymbol{M}^{-1}\nabla_{\boldsymbol{q}} V(\boldsymbol{q}^{(n+1)}). \end{aligned} \tag{14}$$

Only given the positions $\boldsymbol{q}^{(n-1)}$ and $\boldsymbol{q}^{(n)}$, the leapfrog integrator computes the successive position $\boldsymbol{q}^{(n+1)}$ as

$$\begin{aligned} \dot{\boldsymbol{q}}^{(n-\frac{1}{2})} &= \frac{\boldsymbol{q}^{(n)} - \boldsymbol{q}^{(n-1)}}{\Delta t}, \\ \dot{\boldsymbol{q}}^{(n+\frac{1}{2})} &= \dot{\boldsymbol{q}}^{(n-\frac{1}{2})} - \Delta t \boldsymbol{M}^{-1}\nabla_{\boldsymbol{q}} V(\boldsymbol{q}^{(n)}), \\ \boldsymbol{q}^{(n+1)} &= \boldsymbol{q}^{(n)} + \Delta t \dot{\boldsymbol{q}}^{(n+\frac{1}{2})}. \end{aligned} \tag{15}$$

The symplectic Euler method computes successive state $(\boldsymbol{q}^{(n+1)}, \dot{\boldsymbol{q}}^{(n+1)})$ as

$$\begin{aligned} \boldsymbol{q}^{(n+1)} &= \boldsymbol{q}^{(n)} + \Delta t \dot{\boldsymbol{q}}^{(n)}, \\ \dot{\boldsymbol{q}}^{(n+1)} &= \dot{\boldsymbol{q}}^{(n)} - \Delta t \boldsymbol{M}^{-1}\nabla_{\boldsymbol{q}} V(\boldsymbol{q}^{(n+1)}). \end{aligned} \tag{16}$$

Only given the positions, the successive state is

$$\begin{aligned} \dot{\boldsymbol{q}}^{(n-1)} &= \frac{\boldsymbol{q}^{(n)} - \boldsymbol{q}^{(n-1)}}{\Delta t}, \\ \dot{\boldsymbol{q}}^{(n)} &= \dot{\boldsymbol{q}}^{(n-1)} - \Delta t \boldsymbol{M}^{-1}\nabla_{\boldsymbol{q}} V(\boldsymbol{q}^{(n)}), \\ \boldsymbol{q}^{(n+1)} &= \boldsymbol{q}^{(n)} + \Delta t \dot{\boldsymbol{q}}^{(n)}. \end{aligned} \tag{17}$$

The Euler method is expressed as

$$\begin{aligned} \boldsymbol{q}^{(n+1)} &= \boldsymbol{q}^{(n)} + \Delta t \dot{\boldsymbol{q}}^{(n)}, \\ \dot{\boldsymbol{q}}^{(n+1)} &= \dot{\boldsymbol{q}}^{(n)} - \Delta t \boldsymbol{M}^{-1}\nabla_{\boldsymbol{q}} V(\boldsymbol{q}^{(n)}). \end{aligned} \tag{18}$$

Only given the positions, the successive state is

$$\begin{aligned} \dot{\boldsymbol{q}}^{(n-1)} &= \frac{\boldsymbol{q}^{(n)} - \boldsymbol{q}^{(n-1)}}{\Delta t}, \\ \dot{\boldsymbol{q}}^{(n)} &= \dot{\boldsymbol{q}}^{(n-1)} - \Delta t \boldsymbol{M}^{-1}\nabla_{\boldsymbol{q}} V(\boldsymbol{q}^{(n-1)}), \\ \boldsymbol{q}^{(n+1)} &= \boldsymbol{q}^{(n)} + \Delta t \dot{\boldsymbol{q}}^{(n)}. \end{aligned} \tag{19}$$

The potential energy V is parameterized by a neural network. We trained these models by minimizing the error between the predicted and ground truth positions $\boldsymbol{q}^{(n+1)}$. For predicting the successive position, the leapfrog integrator and the symplectic Euler method do the same. However, because of the velocity shifted by a half step, their energy at n-th time step differ from each other.

Neural Network Models. Following the study on DGNet (Matsubara et al., 2020), each method employed a neural network with three layers, 200 hidden units, and $\tanh$ activation functions. Each network had one input unit and one output unit in the mass-spring system and the pendulum system, and four input units and one output unit in the 2-body system. Each weight matrix was initialized as a random orthogonal matrix. Each network was trained using the Adam optimizer (Kingma & Ba, 2015) with a learning rate of 0.001 and a batch size of 200 in all experiments. We used 2,000 iterations for the mass-spring system, 10,000 iterations for the pendulum system, and 100,000 iterations for the 2-body system.

Datasets. For simplicity, we set the mass matrix M as the identity matrix in all experiments. We used 25 trajectories, each composed of 101 data points, and $\Delta t = 0.2$ for training on the mass-spring and pendulum systems, and 800 trajectories, each composed of 501 data points, and $\Delta t = 0.05$ for training on the 2-body system. We generated the training and test datasets by integrating the derivatives of the ground truth Hamiltonian using a Runge-Kutta method (specifically, the adaptive Dormand-Prince method).

Results. Just for an improved visibility, we depicted the results of each model separately in Figs. 2—4.

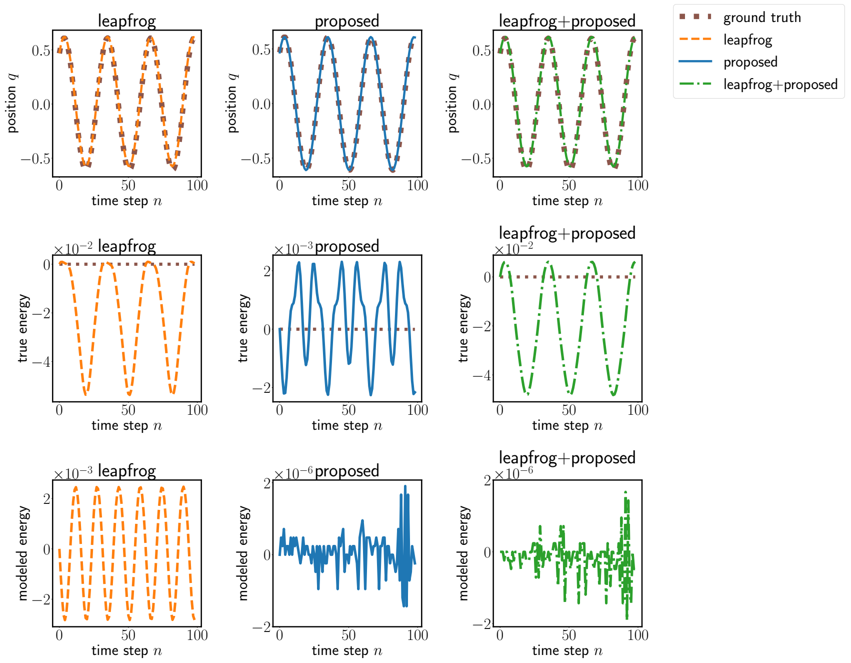

Figure 2: Results of the mass-spring system. (top) Position $\boldsymbol{q}$. (center) True energy. (bottom) Energy modeled by each model.

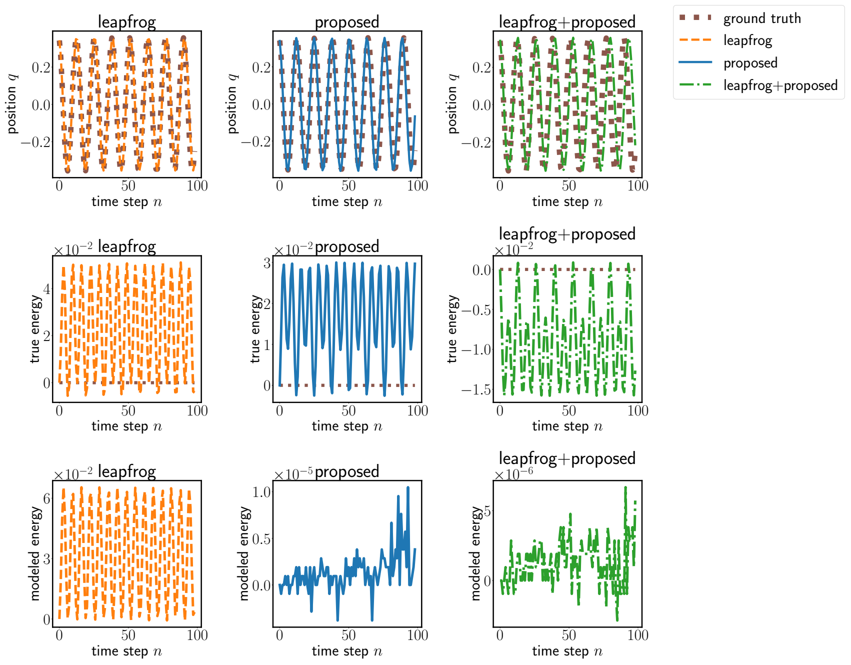

Figure 3: Results of the pendulum system. (top) Position $\boldsymbol{q}$. (center) True energy. (bottom) Energy modeled by each model.

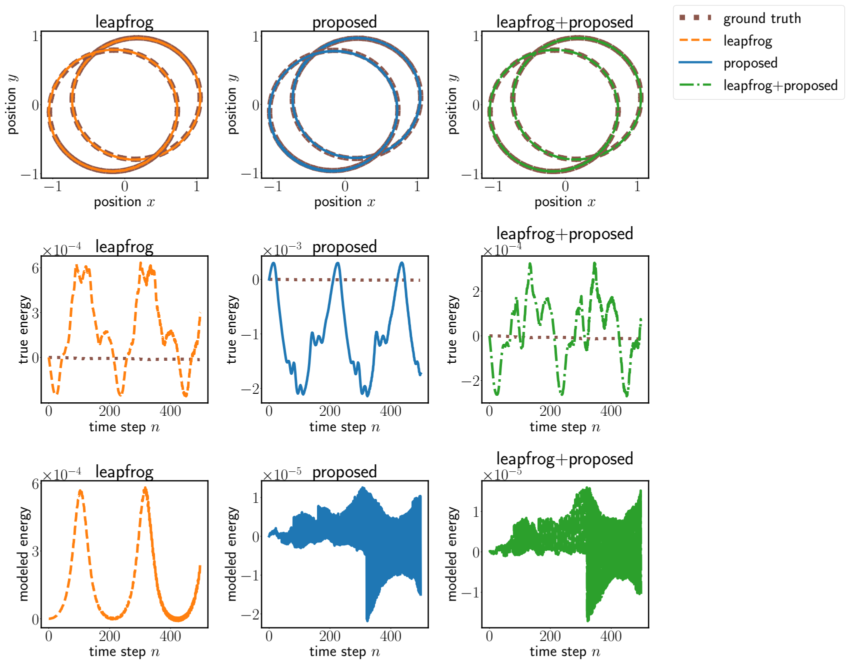

Figure 4: Results of the 2-body system. (top) Position $\boldsymbol{q}$. (center) True energy. (bottom) Energy modeled by each model.