TANGO: Time-reversal Latent GraphODE for Multi-Agent Dynamical Systems

Zijie Huang[1*], **Wanjia Zhao**[2*], **Jingdong Gao**[1], **Ziniu Hu**[1], **Xiao Luo**[1]
Yadi Cao[1], **Yuanzhou Chen**[1], **Yizhou Sun**[1], **Wei Wang**[1]
[1]University of California, Los Angeles, CA, USA,   [2]Zhejiang University, China
{zijiehuang,jingdonggao,bull,yzsun,weiwang}@cs.ucla.edu
wanjiazhao@zju.edu.cn

Abstract

Learning complex multi-agent system dynamics from data is crucial across many domains. Existing physics-informed approaches, like Hamiltonian Neural Network, introduce inductive bias by strictly following energy conservation law. However, many real-world systems do not strictly conserve energy. Thus, we focus on Time-Reversal Symmetry, a broader physical principle indicating that system dynamics should remain invariant when time is reversed. This principle not only preserves energy in conservative systems but also serves as a strong inductive bias for non-conservative, reversible systems. In this paper, we propose a simple-yet-effective self-supervised regularization term as a soft constraint that aligns the forward and backward trajectories predicted by a continuous graph neural network-based ordinary differential equation (GraphODE). In addition, we theoretically show that our regularization essentially minimizes higher-order Taylor expansion terms during the ODE integration steps, which enables our model to be more noise-tolerant and even applicable to irreversible systems[2].

1 Introduction

The intricate nature of multi-agent systems often necessitates vast amounts of training data. Vanilla data-driven neural simulators trained on limited datasets tend to be less generalizable, and can violate physical properties of a system. As depicted in Figure 1 (a.1), the learned energy curve of a baseline model (LG-ODE) (Huang et al., 2020) is prone to explosion, even though the ground-truth energy remains constant. Existing works like Hamiltonian- Neural Nets and ODE (Greydanus et al., 2019; Sanchez-Gonzalez et al., 2019) strictly enforce the energy conservation law, leading to more accurate predictions. However, not all real-world systems adhere to strict energy conservation, such as n-body spring systems with periodic external forces or frictions shown in Figure 1 (a.2) and (a.3). For these systems, applying strict energy conservation constraint proposed by Greydanus et al. (2019); Sanchez-Gonzalez et al. (2019) could lead to inferior performance. As shown in Figure 1(b.1), for classical and deterministic mechanics, energy-conservative systems also obey time-reversal symmetry (Tolman, 1938). Meanwhile, time-reversible systems also encompass non-conservative systems, such as Stokes flow (Pozrikidis, 2001), which has significant real-world applications (Kim & Karrila, 2013). Therefore, by ensuring system dynamics are approximately time-reversal invariant, we can design more realistic and effective neural simulators for dynamical system modeling.

We propose a simple-yet-effective self-supervised regularization term as a soft constraint that aligns forward and backward trajectories predicted by our model, which has GraphODE as its backbone.

---

[*] Equally Contributed
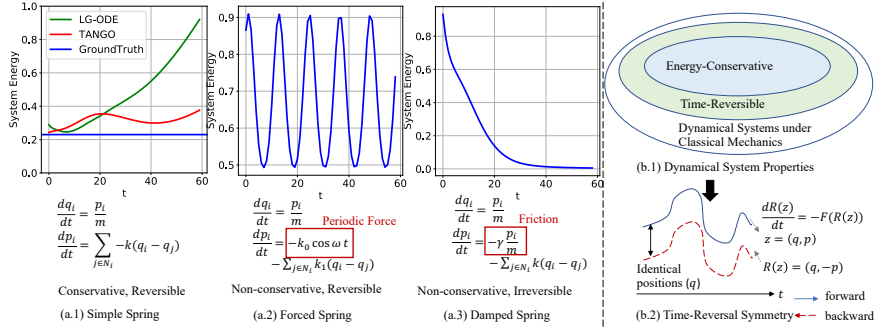
[2]Code implementation can be found at here.

Figure 1: (a) Three n-body spring systems characterized by their energy conservation and time reversal properties. p, q, m denote momentum, position and mass, respectively. Proof of energy conservation and time reversal for these systems can be found in Appendix C (b) Classification of classical mechanical systems based on (Tolman, 1938; Lamb & Roberts, 1998)

We name our model TANGO: **T**ime-Reversal L**a**te**n**t **G**raph **O**DE. This time-reversal loss does not need additional labels beyond ground-truth observations, while effectively imposing time-reversal symmetry to enable more accurate model predictions. We also theoretically explain why the proposed time-reversal symmetry loss could in general help learn more fine-grained and long-context system dynamics from the numerical aspect, benefiting even irreservible systems.

2 Preliminaries

A system is said to have *Time-Reversal Symmetry* if its dynamics remain the same when the flow of time is reversed (Noether, 1971). Formally, let's consider a multi-agent dynamical system described in the matrix form $\frac{d\boldsymbol{Z}(t)}{dt} = F(\boldsymbol{Z}(t))$, where $\boldsymbol{Z}(t) \in \mathbb{R}^{N \times d}$ is the time-varying state matrix of all agents. The system is said to follow the *Time-Reversal Symmetry* if it satisfies

$$\frac{dR(\boldsymbol{Z}(t))}{dt} = -F(R(\boldsymbol{Z}(t)), \tag{1}$$

where $R(\cdot)$ is a reversing operator[3].Time-Reversal Symmetry indicates that after the reversing operation $R(\cdot)$, the gradient of any point of the trajectory $Z(t)$ will be reversed (in the opposite direction).

Now we introduce a time evolution operator $\phi_\tau : \boldsymbol{Z}(t) \mapsto \phi_\tau(\boldsymbol{Z}(t)) = \boldsymbol{Z}(t + \tau)$ for arbitrary $t, \tau \in \mathbb{R}$, which satisfies $\phi_{\tau_1} \circ \phi_{\tau_2} = \phi_{\tau_1 + \tau_2}$ for all $\tau_1, \tau_2 \in \mathbb{R}$[4]. The time evolution operator helps us to move forward or backward through time, thus forming a trajectory. Based on Lamb & Roberts (1998), we can integrate Eqn. 1 and have:

$$R \circ \phi_t = \phi_{-t} \circ R = \phi_t^{-1} \circ R, \tag{2}$$

which means that moving forward t steps and then turning backward is equivalent to firstly turning backward and then moving to the other direction t steps. Eqn. 2 has been widely used to describe time-reversal systems in existing literature (Huh et al., 2020; Valperga et al., 2022; Roberts & Quispel, 1992). Nevertheless, Eqn 2 is equivalent to $R \circ \phi_t \circ R \circ \phi_t = I$, where I denotes identity mapping, which is more intuitive to understand and more straightforward to guide the design of our time-reversal regularizer. It means if we move t steps forward and then turn backward and move for t more steps, it shall restore back to the same state. We prove the equivalence in Appendix B.1

3 Method

We represent a multi-agent dynamical system as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes the node set of N agents[5] and $\mathcal{E}$ denotes the set of edges representing their physical interactions, which for simplicity we assumed to be static over time. We denote $\boldsymbol{X}(t) \in \mathbb{R}^{N \times d}$ as the feature matrix at

---

[3]Taking a Hamiltonian system $(\boldsymbol{q}, \boldsymbol{p}, t)$ as an example, where $\boldsymbol{q}, \boldsymbol{p}$ denote position and momentum, the reversing operator $R : (\boldsymbol{q}, \boldsymbol{p}, t) \mapsto (\boldsymbol{q}, -\boldsymbol{p}, -t)$ makes the momentum reverse and traverse back over time. (Lamb & Roberts, 1998).

[4]$\circ$ denotes composition

[5]Following (Kipf et al., 2018), we use "agents" to denote "objects" in dynamical systems, which is different from "intelligent agent" in AI.
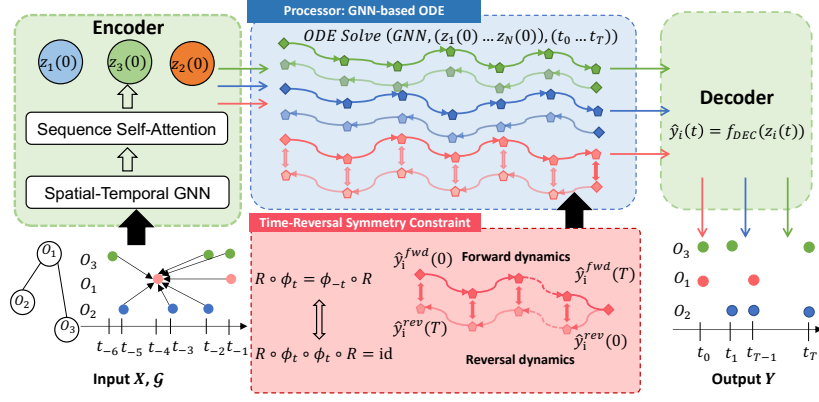
Figure 2: Overall framework of TANGO

timestamp t for all agents, with d as the dimension of features. Model input consists of trajectories of such feature matrices over K timestamps $X(t_{1:K}) = \{\boldsymbol{X}(t_1), \boldsymbol{X}(t_2), \ldots, \boldsymbol{X}(t_K)\}$. To incorporate *Time-Reversal Symmetry* in modeling multi-agent dynamical systems, we propose TANGO, a novel GraphODE framework with a flexible regularization, as depicted in Figure 2.

Forward Trajectory Prediction and Reconstruction Loss. We utilize the GraphODE framework described in Appendix A to model multi-agent dynamical systems, which follows the encoder-processor-decoder architecture. Specifically, we first utilize a Transformer-based spatial-temporal GNN described in Huang et al. (2020) as our encoder $f_{\text{ENC}}(\cdot)$ to compute the initial states from observed trajectories. We then utilize the GNN operator described in Kipf et al. (2018) as our ODE function $g(\cdot)$, which drives the system to move forward and output the forward trajectories for latent states $\boldsymbol{z}_i^{\text{fwd}}(t)$ at each continuous time t and each agent i. Finally, we employ a Multilayer Perceptron (MLP) as a decoder to predict output dynamics based on the latent states. We summarize the whole procedure as:

$$\dot{\boldsymbol{z}}_i^{\text{fwd}}(t) := \frac{d\boldsymbol{z}_i^{\text{fwd}}(t)}{dt} = g(\boldsymbol{z}_1^{\text{fwd}}(t), \boldsymbol{z}_2^{\text{fwd}}(t), \cdots \boldsymbol{z}_N^{\text{fwd}}(t)),$$
$$\text{where} \quad \boldsymbol{z}_0^{\text{fwd}}(t) = f_{\text{ENC}}(X(t_{1:K}), \mathcal{G}), \quad \hat{\boldsymbol{y}}_i^{\text{fwd}}(t) = f_{\text{DEC}}(\boldsymbol{z}_i^{\text{fwd}}(t)) \tag{3}$$

To train such a GraphODE from data, we can use the reconstruction loss that minimizes the L2 distance between predicted forward trajectories $\hat{\boldsymbol{y}}_i^{\text{fwd}}(t)$ and the ground truth trajectories $\boldsymbol{y}_i(t)$

$$\mathcal{L}_{pred} = \sum_{i=1}^{N} \sum_{t=0}^{T} \left\| \boldsymbol{y}_i(t) - \hat{\boldsymbol{y}}_i^{\text{fwd}}(t) \right\|_2^2 \tag{4}$$

Reversed Trajectory Prediction and Regularization Loss. We now design a novel time-reversal symmetry loss as a soft constraint to flexibly regulate systems' total energy to prevent it from changing sharply. Specifically, based on the equivalent definition of time-reversal symmetry, we first compute the backward trajectories $\boldsymbol{z}^{\text{rev}}$ starting from the ending state of the forward one, traversed back over time:

$$\dot{\boldsymbol{z}}_i^{\text{rev}}(t) := \frac{d\boldsymbol{z}_i^{\text{rev}}(t)}{dt} = -g(\boldsymbol{z}_1^{\text{rev}}(t), \boldsymbol{z}_2^{\text{rev}}(t), \cdots \boldsymbol{z}_N^{\text{rev}}(t)),$$
$$\text{where} \quad \boldsymbol{z}_i^{\text{rev}}(0) = \boldsymbol{z}_i^{\text{fwd}}(T), \quad \hat{\boldsymbol{y}}_i^{rev}(t) = f_{\text{DEC}}(\boldsymbol{z}_i^{\text{rev}}(t)). \tag{5}$$

Next, if the system follows *Time-Reversal Symmetry*, the forward and backward trajectories shall exactly overlap. We thus design the reversal loss by minimizing the L2 distances between model forward and backward trajectories:

$$\mathcal{L}_{reverse} = \sum_{i=1}^{N} \sum_{t=0}^{T} \left\| \hat{\boldsymbol{y}}_i^{\text{fwd}}(t) - \hat{\boldsymbol{y}}_i^{\text{rev}}(T-t) \right\|_2^2 \tag{6}$$

Finally, we jointly train TANGO as a weighted combination of the two losses:

$$\mathcal{L} = \mathcal{L}_{pred} + \alpha \mathcal{L}_{reverse} = \sum_{i=1}^{N} \sum_{t=0}^{T} \left\| \boldsymbol{y}_i(t) - \hat{\boldsymbol{y}}_i^{\text{fwd}}(t) \right\|_2^2 + \alpha \sum_{i=1}^{N} \sum_{t=0}^{T} \left\| \hat{\boldsymbol{y}}_i^{\text{fwd}}(t) - \hat{\boldsymbol{y}}_i^{\text{rev}}(T-t) \right\|_2^2, \tag{7}$$

where α is a positive coefficient to balance the two losses based on different targeted systems.

Table 1: Evaluation results on MSE (10^{-2}).

Method	LatentODE	HODEN	TRS-ODEN	TRS-ODEN$_{GNN}$	LGODE	TANGO	TANGO$_{\mathcal{L}_{rev}=\text{gt-rev}}$	TANGO$_{\mathcal{L}_{rev}=\text{rev2}}$
Simple Spring	5.2622	3.0039	3.6785	<u>1.4115</u>	1.7429	**1.1178**	1.1313	1.6786
Forced Spring	5.0277	4.0668	4.4465	<u>2.1102</u>	1.8929	**1.4525**	1.5254	1.9786
Damped Spring	3.3419	8.7950	1.7595	<u>0.5951</u>	0.9718	**0.5944**	0.6171	0.9692
Pendulum	2.6894	741.2296	741.4988	596.0319	<u>1.4156</u>	**1.2527**	1.6158	1.5631

Theoretical Analysis of Time-Reversal Symmetry Loss. We next theoretically show that the time-reversal loss also numerically reduces prediction errors, making it valuable for a wide range of classical mechanics systems. Specifically, it minimizes higher-order Taylor expansion terms during ODE integration.

Theorem 1. *Let Δt denote the integration step size in an ODE solver and T be the prediction length. The reconstructed loss $\mathcal{L}_{pred}$ defined in Eq 4 is $\mathcal{O}(T^3 \Delta t^2)$. The time-reversal loss $\mathcal{L}_{reverse}$ defined in Eqn 6 is $\mathcal{O}(T^5 \Delta t^4)$.*

We prove Theorem 1 in Appendix B.2. From Theorem 1, we can see two nice properties of our proposed time-reversal loss: 1) Regarding the relationship to Δt, $\mathcal{L}_{reverse}$ is optimizing a high-order term Δt^4, which forces the model to predict fine-grained physical properties such as jerk (the derivatives of accelerations). In comparison, the reconstruction loss optimizes Δt^2, which mainly guides the model to predict the locations/velocities. Therefore, the combined loss enables our model to be more noise-tolerable; 2) $\mathcal{L}_{reverse}$ is more sensitive to total sequence length (T^5), thus it provides more regularization for long-context prediction, a key challenge for dynamic modeling.

Three additional justifications about our implementations can be found in Appendix B.3.

4 Experiments

Dynamical Systems and Datasets. We conduct systematic evaluations over three different spring systems (Kipf et al., 2018) and a complex chaotic pendulum system. For spring systems, we consider three system settings: 1) conservative, i.e. no interactions with the environments, we call it *Simple Spring*; 2) non-conservative with frictions, we call it *Damped Spring*; 3) non-conservative with periodic external forces, we call it *Forced Spring*. The chaotic triple *Pendulum* system consists of sequentially connected sticks in a 2D plane, and is highly sensitive to initial conditions, leading to diverse trajectories (Shinbrot et al., 1992; Stachowiak & Okada, 2006; Awrejcewicz et al., 2008) that are chaotic and nonlinear. The dataset and task setup details can be found in Appendix D.1 D.2.

Main Results. Table 1 presents prediction performance across models measured by mean squared error (MSE). TANGO consistently surpasses other models across datasets, highlighting its generalizability and the efficacy of its reversal loss. Specifically, it reports an improvement in MSE ranging from 11.5% to 34.6% over the second-best baseline. The baseline and implementation details can be found in Appendix D.3 D.4. We observe that multi-agent approaches (LG-ODE, TRS-ODEN$_{GNN}$, TANGO) consistently outperform single-agent baselines (LatentODE, HODEN, TRS-ODEN), showing the importance of capturing inter-agent interactions via the message passing of Graph Neural Networks (GNN) encoding and ODE operations.

When examining individual datasets, HODEN excels among single-agent models on *Simple Spring*, indicating the benefit of incorporating appropriate physical bias. However, its performance is notably lower on *Force Spring* and *Damped Spring*. This suggests that wrongly applied inductive biases can degrade performance. Consequently, while HODEN is suited for strictly energy-conservative systems, TANGO offers versatility across diverse classical dynamics. Note that HODEN naively forces each agent to be energy-conservative, instead of the whole system. Therefore, it still performs worse than multi-agent baselines on energy-conservative systems.

The chaotic nature and high sensitivity to initial states [6] of the *Pendulum* system create challenges for dynamic modeling. This lead to unstable predictions for models like HODEN and TRS-ODEN, as these methods rely on linear spline interpolation (Endre Süli, 2003) to approximate missing initial states of agents, which can cause dramatic prediction errors. In contrast, latent models like LatentODE, LG-ODE, and TANGO that utilize advanced encoders derive latent states from observed data, yielding superior accuracy. TANGO maintains the most accurate predictions on the *Pendulum* system, further showing its robust generalization capabilities. We provide more experiment results and ablations in Appendix D.5.

---

[6]Visualization to show *Pendulum* is highly sensitive to different initial states can be found here

5 Limitations

Currently TANGO only incorporates inductive bias from the temporal aspect, while there are still many important properties in the spatial aspect such as translation and rotation equivariance Satorras et al. (2021). Future endeavors that combine biases from both temporal and spatial dimensions could unveil a new frontier in dynamical systems modeling.

References

Jan Awrejcewicz, Grzegorz Kudra, and Grzegorz Wasilewski. Chaotic zones in triple pendulum dynamics observed experimentally and numerically. *Applied Mechanics and Materials*, pp. 1–17, 2008.

Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and koray kavukcuoglu. Interaction networks for learning about objects, relations and physics. In *Advances in Neural Information Processing Systems*. 2016.

Ricky T. Q. Chen, Brandon Amos, and Maximilian Nickel. Learning neural event functions for ordinary differential equations. *International Conference on Learning Representations*, 2021.

Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, 2018.

David F. Mayers Endre Süli. *An Introduction to Numerical Analysis*. Cambridge University Press, 2003.

Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. *Advances in Neural Information Processing Systems*, 2019.

Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In *Proceedings of the 2020 World Wide Web Conference*, 2020.

Zijie Huang, Yizhou Sun, and Wei Wang. Learning continuous system dynamics from irregularly-sampled partial observations. In *Advances in Neural Information Processing Systems*, 2020.

Zijie Huang, Yizhou Sun, and Wei Wang. Coupled graph ode for learning interacting system dynamics. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2021.

In Huh, Eunho Yang, Sung Ju Hwang, and Jinwoo Shin. Time-reversal symmetric ode network. In *Advances in Neural Information Processing Systems*, 2020.

Sangtae Kim and Seppo J Karrila. *Microhydrodynamics: principles and selected applications*. Courier Corporation, 2013.

Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. *arXiv preprint arXiv:1802.04687*, 2018.

Jeroen SW Lamb and John AG Roberts. Time-reversal symmetry in dynamical systems: a survey. *Physica D: Nonlinear Phenomena*, pp. 1–39, 1998.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *The International Conference on Learning Representations*, 2019.

Xiao Luo, Jingyang Yuan, Zijie Huang, Huiyu Jiang, Yifang Qin, Wei Ju, Ming Zhang, and Yizhou Sun. HOPE: High-order graph ODE for modeling interacting dynamics. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

Emmy Noether. Invariant variation problems. *Transport theory and statistical physics*, 1(3):186–207, 1971.

Jill North. Formulations of classical mechanics. *Forthcoming in A companion to the philosophy of physics. Routledge*, 2021.

Michael Poli, Stefano Massaroli, Junyoung Park, Atsushi Yamashita, Hajime Asama, and Jinkyoo Park. Graph neural ordinary differential equations. *arXiv preprint arXiv:1911.07532*, 2019.

Costas Pozrikidis. Interfacial dynamics for stokes flow. *Journal of Computational Physics*, 169(2): 250–301, 2001.

John AG Roberts and GRW1173588 Quispel. Chaos and time-reversal symmetry. order and chaos in reversible dynamical systems. *Physics Reports*, 216(2-3):63–177, 1992.

Yulia Rubanova, Ricky TQ Chen, and David K Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. In *Advances in Neural Information Processing Systems*, 2019.

Alvaro Sanchez-Gonzalez, Victor Bapst, Kyle Cranmer, and Peter Battaglia. Hamiltonian graph networks with ode integrators. In *Advances in Neural Information Processing Systems*, 2019.

Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia. Learning to simulate complex physics with graph networks. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

Víctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E (n) equivariant graph neural networks. In *International conference on machine learning*, pp. 9323–9332. PMLR, 2021.

Michael Schober, Simo Särkkä, and Philipp Hennig. A probabilistic model for the numerical solution of initial value problems. In *Statistics and Computing*, pp. 99–122. 2019.

Hochreiter Sepp and Schmidhuber Jürgen. Long short-term memory. *Neural computation*, 1997.

Troy Shinbrot, Celso Grebogi, Jack Wisdom, and James A Yorke. Chaos in a double pendulum. *American Journal of Physics*, (6):491–499, 1992.

Tomasz Stachowiak and Toshio Okada. A numerical analysis of chaos in the double pendulum. *Chaos, Solitons & Fractals*, (2):417–422, 2006.

E. C. Tolman. The determiners of behavior at a choice point. *Psychological Review*, 45(1):1–41, 1938.

Riccardo Valperga, Kevin Webster, Dmitry Turaev, Victoria Klein, and Jeroen Lamb. Learning reversible symplectic dynamics. In *Proceedings of The 4th Annual Learning for Dynamics and Control Conference*, 2022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*. 2017.

Song Wen, Hao Wang, and Dimitris Metaxas. Social ode: Multi-agent trajectory forecasting with neural ordinary differential equations. In *European Conference on Computer Vision*, 2022.

Chengxi Zang and Fei Wang. Neural dynamics on complex networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2020.

A Model Details

We represent a multi-agent dynamical system as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes the node set of N agents[7] and $\mathcal{E}$ denotes the set of edges representing their physical interactions, which for simplicity we assumed to be static over time. We denote $\boldsymbol{X}(t) \in \mathbb{R}^{N \times d}$ as the feature matrix at timestamp t for all agents, with d as the dimension of features. Model input consists of trajectories of such feature matrices over K timestamps $X(t_{1:K}) = \{\boldsymbol{X}(t_1), \boldsymbol{X}(t_2), \ldots, \boldsymbol{X}(t_K)\}$. Note that the timestamps $t_1, t_2 \cdots t_K$ can have non-uniform intervals and be of any continuous values. Our goal is to learn a neural simulator $f_\theta(\cdot) : X(t_{1:K}) \to Y(t_{K+1:T})$, which predicts node dynamics $\boldsymbol{Y}(t)$ in the future based on observations. We use $\boldsymbol{y}_i(t)$ to denote the targeted dynamic vector of agent i at time t. In some cases when we are only predicting system feature trajectories, $\boldsymbol{Y}(\cdot) \equiv \boldsymbol{X}(\cdot)$.

A.1 GraphODE for Multi-agent Dynamical Systems

Graph Neural Networks (GNNs) have been widely used to model multi-agent dynamical systems which approximate pair-wise node (agent) interaction through message passing to impose strong inductive bias. The majority of them are discrete models, which learn a fixed-step state transition function such as to predict trajectories from timestamp t to timestamp $t + 1$. However, discrete models (Battaglia et al., 2016; Kipf et al., 2018; Sanchez-Gonzalez et al., 2020) have two main limitations: (1) they are not able to adequately model systems that are continuous in nature such as n-body spring systems. (2) they cannot deal with irregular-observed systems, where the observations for different agents are not temporally aligned and can happen at arbitrary time intervals.

Recently, researchers propose GraphODE models (Poli et al., 2019; Huang et al., 2020; Zang & Wang, 2020; Luo et al., 2023; Wen et al., 2022) which describe multi-agent dynamical systems by a series of ODEs in a continuous manner. Specifically, they employ GNN as the ODE function and learn it in a data-driven way, making GraphODEs flexible to model a wide range of real-worldsystems (Chen et al., 2018; Rubanova et al., 2019). The state evolution can be described as: $\dot{\boldsymbol{z}}_i(t) := \frac{d\boldsymbol{z}_i(t)}{dt} = g\left(\boldsymbol{z}_1(t), \boldsymbol{z}_2(t) \cdots \boldsymbol{z}_N(t)\right)$, where $\boldsymbol{z}_i(t) \in \mathbb{R}^d$ denotes the latent state variable for agent i at timestamp t and g denotes the message passing function that drives the system to move forward. GraphODEs have been shown to achieve superior performance, especially in long-range predictions and can handle data irregularity issues. They usually follow an encoder-processor-decoder architecture, where an encoder first computes the latent initial states $\boldsymbol{z}_1(0), \cdots \boldsymbol{z}_N(0)$ for all agents simultaneously based on their historical observations as in Eqn 8.

$$\boldsymbol{z}_1(0), \boldsymbol{z}_2(0), ..., \boldsymbol{z}_N(0) = f_{\text{ENC}}\left(X(t_{1:K}), \mathcal{G}\right) \tag{8}$$

Then the GNN-based ODE predicts the latent trajectories starting from the learned initial states. Given the initial states $\boldsymbol{z}_1(0), \cdots \boldsymbol{z}_N(0)$ for all agents and the ODE, we can compute the latent state $\boldsymbol{z}_i(T)$ at any desired time using a numerical ODE solver such as Runge-Kuttais (Schober et al., 2019) as shown in Eqn 9.

$$\boldsymbol{z}_i(T) = \text{ODE-Solver}\left(g, [\boldsymbol{z}_1(0), \boldsymbol{z}_2(0)...\boldsymbol{z}_N(0)], t = T\right) = \boldsymbol{z}_i(0) + \int_{t=0}^{T} g\left(\boldsymbol{z}_1(t), \boldsymbol{z}_2(t) \cdots \boldsymbol{z}_N(t)\right) dt \tag{9}$$

Finally, a decoder extracts the predicted dynamics $\hat{\boldsymbol{y}}_i(1), \hat{\boldsymbol{y}}_i(2), ..., \hat{\boldsymbol{y}}_i(T)$ based on the trajectory of latent states $\boldsymbol{z}_i(1), \boldsymbol{z}_i(2), ..., \boldsymbol{z}_i(T)$.

$$\hat{\boldsymbol{y}}_i(t) = f_{\text{DEC}}(\boldsymbol{z}_i(t)) \tag{10}$$

In the following we introduce in details how we implement our model and each baseline.

A.2 Initial State Encoder

The initial state encoder computes the latent node initial states $\boldsymbol{z}_i(t)$ for all agents simultaneously considering their mutual interaction. Specifically, it first fuses all observations into a temporal

---

[7]Following (Kipf et al., 2018), we use "agents" to denote "objects" in dynamical systems, which is different from "intelligent agent" in AI.

graph and conducts dynamic node representation through a spatial-temporal GNN as in Huang et al. (2020):

$$\boldsymbol{h}_{j(t)}^{l+1} = \boldsymbol{h}_{j(t)}^{l} + \sigma \left(\sum_{i(t') \in \mathcal{N}_{j(t)}} \alpha_{i(t') \to j(t)}^{l} \times \boldsymbol{W}_v \hat{\boldsymbol{h}}_{i(t')}^{l-1} \right)$$

$$\alpha_{i(t') \to j(t)}^{l} = \left(\boldsymbol{W}_k \hat{\boldsymbol{h}}_{i(t')}^{l-1} \right)^T \left(\boldsymbol{W}_q \boldsymbol{h}_{j(t)}^{l-1} \right) \cdot \frac{1}{\sqrt{d}}, \quad \hat{\boldsymbol{h}}_{i(t')}^{l-1} = \boldsymbol{h}_{i(t')}^{l-1} + \text{TE}(t' - t) \tag{11}$$

$$\text{TE}(\Delta t)_{2i} = \sin \left(\frac{\Delta t}{10000^{2i/d}} \right), \quad \text{TE}(\Delta t)_{2i+1} = \cos \left(\frac{\Delta t}{10000^{2i/d}} \right),$$

where $||$ denotes concatenation; $\sigma(\cdot)$ is a non-linear activation function; d is the dimension of node embeddings. The node representation is computed as a weighted summation over its neighbors plus residual connection where the attention score is a transformer-based Vaswani et al. (2017) dot-product of node representations by the use of value, key, query projection matrices $\boldsymbol{W}_v, \boldsymbol{W}_k, \boldsymbol{W}_q$. Here $\boldsymbol{h}_{j(t)}^{l}$ is the representation of agent j at time t in the l-th layer. $i(t')$ is the general index for neighbors connected by temporal edges (where $t' \neq t$) and spatial edges (where $t = t'$ and $i \neq j$). The temporal encoding Hu et al. (2020) is added to a neighborhood node representation in order to distinguish its message delivered via spatial and temporal edges. Then, we stack L layers to get the final representation for each observation node: $\boldsymbol{h}_i^t = \boldsymbol{h}_{i(t)}^{L}$. Finally, we employ a self-attention mechanism to generate the sequence representation $\boldsymbol{u}_i$ for each agent as their latent initial states:

$$\boldsymbol{u}_i = \frac{1}{K} \sum_t \sigma \left(\boldsymbol{a}_i^T \hat{\boldsymbol{h}}_i^t \hat{\boldsymbol{h}}_i^t \right), \quad \boldsymbol{a}_i = \tanh \left(\left(\frac{1}{K} \sum_t \hat{\boldsymbol{h}}_i^t \right) \boldsymbol{W}_a \right), \tag{12}$$

where $\boldsymbol{a}_i$ is the average of observation representations with a nonlinear transformation $\boldsymbol{W}_a$ and $\hat{\boldsymbol{h}}_i^t = \boldsymbol{h}_i^t + \text{TE}(t)$. K is the number of observations for each trajectory. Compared with recurrent models such as RNN, LSTM Sepp & Jürgen (1997), it offers better parallelization for accelerating training speed and in the meanwhile alleviates the vanishing/exploding gradient problem brought by long sequences.

Given the latent initial states, the dynamics of the whole system are determined by the ODE function g which we parametrize as a GNN as in Huang et al. (2020) to capture the continuous interaction among agents. We then employ Multilayer Perceptron (MLP) as a decoder to predict the trajectories $\hat{\boldsymbol{y}}_i(t)$ from the latent states $\boldsymbol{z}_i(t)$.

$$\boldsymbol{z}_i(t), \boldsymbol{z}_i(1), \boldsymbol{z}_i(2) \cdots \boldsymbol{z}_i(T) = \text{ODEsolver}(g, [\boldsymbol{z}_1(0), \boldsymbol{z}_2(0) \cdots \boldsymbol{z}_N(0)], (t_0, t_1 \cdots t_T))$$
$$\hat{\boldsymbol{y}}_i(t) = f_{dec}(\boldsymbol{z}_i(t)) \tag{13}$$

B   Theoretical Analysis

B.1   Lemma 1

Lemma 1. *Eqn 2 is equivalent to $R \circ \phi_t \circ R \circ \phi_t = I$, where I denotes identity mapping.*

Proof. The definition of time-reversal symmetry is given by:

$$R \circ \phi_t = \phi_{-t} \circ R = \phi_t^{-1} \circ R \tag{14}$$

Here, R is an involution operator, which means $R \circ R = \text{id}$.

First, we apply the time evolution operator ϕ_t to both sides of Eqn. equation 14:

$$\phi_t \circ R \circ \phi_t = \phi_t \circ \phi_t^{-1} \circ R \tag{15}$$

Simplifying, we obtain:

$$\phi_t \circ R \circ \phi_t = R \tag{16}$$

Next, we apply the involution operator R to both sides of the equation:

$$R \circ \phi_t \circ R \circ \phi_t = R \circ R \tag{17}$$

Since $R \circ R = \mathrm{I}$, we finally arrive at:

$$R \circ \phi_t \circ R \circ \phi_t = \mathrm{I} \tag{18}$$

which means the trajectories can overlap when evolving backward from the final state. $\qquad\square$

B.2   Proof of Theorem 1

Let Δt denote the integration step size in an ODE solver and T be the prediction length. The time stamps of the ODE solver are $\{t_j\}_{j=0}^T$, where $t_{j+1} - t_j = \Delta t$ for $j = 0, \cdots, T(T > 1)$. Next suppose during the forward evolution, the updates go through states $z^{\mathrm{fwd}}(t_j) = (q^{\mathrm{fwd}}(t_j), p^{\mathrm{fwd}}(t_j))$ for $j = 0, \cdots, T$, where $q^{\mathrm{fwd}}(t_j)$ is position, $p^{\mathrm{fwd}}(t_j)$ is momentum, while during the reverse evolution they go through states $z^{\mathrm{rev}}(t_j) = (q^{\mathrm{rev}}(t_j), p^{\mathrm{rev}}(t_j))$ for $j = 0, \cdots, T$, in reverse order. The ground truth trajectory is $z^{\mathrm{gt}}(t_j) = (q^{\mathrm{gt}}(t_j), p^{\mathrm{gt}}(t_j))$ for $j = 0, \cdots, T$.

For the sake of brevity in the ensuing proof, we denote $z^{\mathrm{gt}}(t_j)$ by z_j^{gt}, $z^{\mathrm{fwd}}(t_j)$ by z_j^{fwd} and $z^{\mathrm{rev}}(t_j)$ by z_j^{rev}, and we will use Mathematical Induction to prove the theorem.

B.2.1   Reconstruction Loss ($\mathcal{L}_{pred}$) Analysis.

First, we bound the forward loss $\sum_{j=0}^{T} \|z_j^{\mathrm{fwd}} - z_j^{\mathrm{gt}}\|_2^2$. Since our method models the momentum and position of the system, we can write the following Taylor expansion of the forward process, where for any $0 \le j < T$:

$$\begin{cases} q_{j+1}^{\mathrm{fwd}} = q_j^{\mathrm{fwd}} + (p_j^{\mathrm{fwd}}/m)\Delta t + (\dot{p}_j^{\mathrm{fwd}}/2m)\Delta t^2 + \mathcal{O}(\Delta t^3), & \text{(19a)} \\ p_{j+1}^{\mathrm{fwd}} = p_j^{\mathrm{fwd}} + \dot{p}_j^{\mathrm{fwd}}\Delta t + \mathcal{O}(\Delta t^2), & \text{(19b)} \\ \dot{p}_{j+1}^{\mathrm{fwd}} = \dot{p}_j^{\mathrm{fwd}} + \mathcal{O}(\Delta t), & \text{(19c)} \end{cases}$$

and for the ground truth process, we also have from Taylor expansion that

$$\begin{cases} q_{j+1}^{\mathrm{gt}} = q_j^{\mathrm{gt}} + (p_j^{\mathrm{gt}}/m)\Delta t + (\dot{p}_j^{\mathrm{gt}}/2m)\Delta t^2 + \mathcal{O}(\Delta t^3), & \text{(20a)} \\ p_{j+1}^{\mathrm{gt}} = p_j^{\mathrm{gt}} + \dot{p}_j^{\mathrm{gt}}\Delta t + \mathcal{O}(\Delta t^2), & \text{(20b)} \\ \dot{p}_{j+1}^{\mathrm{gt}} = \dot{p}_j^{\mathrm{gt}} + \mathcal{O}(\Delta t). & \text{(20c)} \end{cases}$$

With these, we aim to prove that for any $k = 0, 1, \cdots, T$, the following hold :

$$\begin{cases} \|q_k^{\mathrm{fwd}} - q_k^{\mathrm{gt}}\|_2 \le C_2^{\mathrm{fwd}} k^2 \Delta t^2, & \text{(21a)} \\ \|p_k^{\mathrm{fwd}} - p_k^{\mathrm{gt}}\|_2 \le C_1^{\mathrm{fwd}} k \Delta t, & \text{(21b)} \end{cases}$$

where C_1^{fwd} and C_2^{fwd} are constants.

Base Case $k = 0$: Based on the initialization rules, it is obvious that $\|q_0^{\mathrm{fwd}} - q_0^{\mathrm{gt}}\|_2 = 0$ and $\|p_0^{\mathrm{fwd}} - p_0^{\mathrm{gt}}\|_2 = 0$, thus (21a) and (21b) both hold for $k = 0$.

Inductive Hypothesis: Assume (21a) and (21b) hold for $k = j$, which means:

$$\begin{cases} \|q_j^{\mathrm{fwd}} - q_j^{\mathrm{gt}}\|_2 \le C_2^{\mathrm{fwd}} j^2 \Delta t^2, & \text{(22a)} \\ \|p_j^{\mathrm{fwd}} - p_j^{\mathrm{gt}}\|_2 \le C_1^{\mathrm{fwd}} j \Delta t, & \text{(22b)} \end{cases}$$

Inductive Proof: We need to prove (21a) and (21b) hold for $k = j + 1$.

First, using (19c) and (20c), we have

$$\left\|\dot{p}_{j+1}^{\mathrm{fwd}} - \dot{p}_{j+1}^{\mathrm{gt}}\right\|_2 = \left\|\dot{p}_j^{\mathrm{fwd}} - \dot{p}_j^{\mathrm{gt}}\right\|_2 + \mathcal{O}(\Delta t) = \left\|\dot{p}_0^{\mathrm{fwd}} - \dot{p}_0^{\mathrm{gt}}\right\|_2 + \mathcal{O}\big((j+1)\Delta t\big) = \mathcal{O}(1), \tag{23}$$

9

where we iterate through $j, j-1, \cdots, 0$ in the second equality. Then using (20b) and (19b), we get for $j+1$ that

$$
\begin{aligned}
\left\|\boldsymbol{p}_{j+1}^{\mathrm{fwd}} - \boldsymbol{p}_{j+1}^{\mathrm{gt}}\right\|_2 &= \left\|(\boldsymbol{p}_j^{\mathrm{fwd}} + \dot{\boldsymbol{p}}_j^{\mathrm{fwd}}\Delta t) - (\boldsymbol{p}_j^{\mathrm{gt}} + \dot{\boldsymbol{p}}_j^{\mathrm{gt}}\Delta t) + \mathcal{O}(\Delta t^2)\right\|_2 \\
&\leq \left\|\boldsymbol{p}_j^{\mathrm{fwd}} - \boldsymbol{p}_j^{\mathrm{gt}}\right\|_2 + \left\|\dot{\boldsymbol{p}}_j^{\mathrm{fwd}} - \dot{\boldsymbol{p}}_j^{\mathrm{gt}}\right\|_2\Delta t + \mathcal{O}(\Delta t^2) \\
&\leq \left[C_1^{\mathrm{fwd}} j + \mathcal{O}(1)\right]\Delta t,
\end{aligned}
$$

where the first inequality uses the triangle inequality, and in the second inequality we use (22b) as well as (23). We can see there exists C_1^{fwd} such that the final expression above is upper bounded by $C_1^{\mathrm{fwd}}(j+1)\Delta t$, with which the claim holds for $j+1$.

Next for (21a), using (20a) and (19a), we get for any j that

$$
\begin{aligned}
\left\|\boldsymbol{q}_{j+1}^{\mathrm{fwd}} - \boldsymbol{q}_{j+1}^{\mathrm{gt}}\right\|_2 &= \left\|(\boldsymbol{q}_j^{\mathrm{fwd}} + (\boldsymbol{p}_j^{\mathrm{fwd}}/m)\Delta t + (\dot{\boldsymbol{p}}_j^{\mathrm{fwd}}/2m)\Delta t^2)\right. \\
&\quad \left. - (\boldsymbol{q}_j^{\mathrm{gt}} + (\boldsymbol{p}_j^{\mathrm{gt}}/m)\Delta t + (\dot{\boldsymbol{p}}_j^{\mathrm{gt}}/2m)\Delta t^2) + \mathcal{O}(\Delta t^3)\right\|_2 \\
&\leq \left\|\boldsymbol{q}_j^{\mathrm{fwd}} - \boldsymbol{q}_j^{\mathrm{gt}}\right\|_2 + \frac{1}{m}\left\|\boldsymbol{p}_j^{\mathrm{fwd}} - \boldsymbol{p}_j^{\mathrm{gt}}\right\|_2\Delta t + \frac{1}{2m}\left\|\dot{\boldsymbol{p}}_j^{\mathrm{fwd}} - \dot{\boldsymbol{p}}_j^{\mathrm{gt}}\right\|_2\Delta t^2 + \mathcal{O}(\Delta t^3) \\
&\leq \left[C_2^{\mathrm{fwd}} j^2 + \frac{C_1^{\mathrm{fwd}}}{m}j + \mathcal{O}(1)\right]\Delta t^2,
\end{aligned}
$$

where the first inequality uses the triangle inequality, and in the second inequality we use (22a) and (22b) as well as (23). Thus with an appropriate C_2^{fwd}, we have the final expression above is upper bounded by $C_2^{\mathrm{fwd}}(j+1)^2\Delta t^2$, and so the claim holds for $j+1$.

Since both the base case and the inductive step have been proven, by the principle of mathematical induction, (21a) and (21b) holds for all $k = 0, 1, \cdots, T$.

With this, we finish the forward proof by plugging (21a) and (21b) into the loss function:

$$
\begin{aligned}
\sum_{j=0}^{T}\|\boldsymbol{z}_j^{\mathrm{fwd}} - \boldsymbol{z}_j^{\mathrm{gt}}\|_2^2 &= \sum_{j=0}^{T}\|\boldsymbol{p}_j^{\mathrm{fwd}} - \boldsymbol{p}_j^{\mathrm{gt}}\|_2^2 + \sum_{j=0}^{T}\|\boldsymbol{q}_j^{\mathrm{fwd}} - \boldsymbol{q}_j^{\mathrm{gt}}\|_2^2 \\
&\leq \left(C_1^{\mathrm{fwd}}\right)^2\sum_{j=0}^{T} j^2\Delta t^2 + \left(C_2^{\mathrm{fwd}}\right)^2\sum_{j=0}^{T} j^4\Delta t^4 \\
&= \mathcal{O}(T^3\Delta t^2).
\end{aligned}
$$

B.2.2 Reversal Loss ($\mathcal{L}_{reverse}$) Analysis.

Next we analyze the reversal loss $\sum_{j=0}^{T}\|R(\boldsymbol{z}_j^{\mathrm{rev}}) - \boldsymbol{z}_j^{\mathrm{fwd}}\|_2^2$. For this, we need to refine the Taylor expansion residual terms for a more in-depth analysis.

First reconsider the forward process. Since the process is generated from the learned network, we may assume that for some constants c_1, c_2, and c_3, the states satisfy the following for any $0 \leq j < T$:

$$
\begin{cases}
\boldsymbol{q}_j^{\mathrm{fwd}} = \boldsymbol{q}_{j+1}^{\mathrm{fwd}} - (\boldsymbol{p}_{j+1}^{\mathrm{fwd}}/m)\Delta t + (\dot{\boldsymbol{p}}_{j+1}^{\mathrm{fwd}}/2m)\Delta t^2 + \mathbf{rem}_j^{\mathrm{fwd},3}, & \text{(24a)} \\
\boldsymbol{p}_j^{\mathrm{fwd}} = \boldsymbol{p}_{j+1}^{\mathrm{fwd}} - \dot{\boldsymbol{p}}_{j+1}^{\mathrm{fwd}}\Delta t + \mathbf{rem}_j^{\mathrm{fwd},2}, & \text{(24b)} \\
\dot{\boldsymbol{p}}_j^{\mathrm{fwd}} = \dot{\boldsymbol{p}}_{j+1}^{\mathrm{fwd}} + \mathbf{rem}_j^{\mathrm{fwd},1}, & \text{(24c)}
\end{cases}
$$

where the remaining terms $\left\|\mathbf{rem}_j^{\mathrm{fwd},i}\right\|_2 \leq c_i\Delta t^i$ for $i = 1, 2, 3$. Similarly, we have approximate Taylor expansions for the reverse process:

$$
\begin{cases}
\boldsymbol{q}_j^{\mathrm{rev}} = \boldsymbol{q}_{j+1}^{\mathrm{rev}} + (\boldsymbol{p}_{j+1}^{\mathrm{rev}}/m)\Delta t + (\dot{\boldsymbol{p}}_{j+1}^{\mathrm{rev}}/2m)\Delta t^2 + \mathbf{rem}_j^{\mathrm{rev},3}, & \text{(25a)} \\
\boldsymbol{p}_j^{\mathrm{rev}} = \boldsymbol{p}_{j+1}^{\mathrm{rev}} + \dot{\boldsymbol{p}}_{j+1}^{\mathrm{rev}}\Delta t + \mathbf{rem}_j^{\mathrm{rev},2}, & \text{(25b)} \\
\dot{\boldsymbol{p}}_j^{\mathrm{rev}} = \dot{\boldsymbol{p}}_{j+1}^{\mathrm{rev}} + \mathbf{rem}_j^{\mathrm{rev},1}, & \text{(25c)}
\end{cases}
$$

where $\left\|\mathbf{rem}_j^{\mathrm{rev},i}\right\|_2 \leq c_i\Delta t^i$ for $i = 1, 2, 3$.

We will prove via induction that for $k = T, T-1, \cdots, 0$,

$$\begin{cases} \|R(\boldsymbol{q}_k^{\text{rev}}) - \boldsymbol{q}_k^{\text{fwd}}\|_2 \leq C_3^{\text{rev}}(T-k)^3 \Delta t^3, & \text{(26a)} \\[4pt] \|R(\boldsymbol{p}_k^{\text{rev}}) - \boldsymbol{p}_k^{\text{fwd}}\|_2 \leq C_2^{\text{rev}}(T-k)^2 \Delta t^2, & \text{(26b)} \\[4pt] \|R(\dot{\boldsymbol{p}}_k^{\text{rev}}) - \dot{\boldsymbol{p}}_k^{\text{fwd}}\|_2 \leq C_1^{\text{rev}}(T-k) \Delta t, & \text{(26c)} \end{cases}$$

where C_1^{rev}, C_2^{rev} and C_3^{rev} are constants.

The entire proof process is analogous to the previous analysis of Reconstruction Loss.

Base Case $k = T$: Since the reverse process is initialized by the forward process variables at $k = T$, it is obvious that $\|\boldsymbol{q}_T^{\text{fwd}} - \boldsymbol{q}_T^{ev}\|_2 = \|\boldsymbol{p}_T^{\text{fwd}} - \boldsymbol{p}_T^{\text{rev}}\|_2 = \|\dot{\boldsymbol{p}}_T^{\text{fwd}} - \dot{\boldsymbol{p}}_T^{\text{rev}}\|_2 = 0$. Thus (26a), (26b) and (26c) all hold for $k = 0$.

Inductive Hypothesis: Assume the inequalities (26b), (26a) and (26c) hold for $k = j + 1$, which means:

$$\begin{cases} \|R(\boldsymbol{q}_{j+1}^{\text{rev}}) - \boldsymbol{q}_{j+1}^{\text{fwd}}\|_2 \leq C_3^{\text{rev}}(T-(j+1))^3 \Delta t^3, & \text{(27a)} \\[4pt] \|R(\boldsymbol{p}_{j+1}^{\text{rev}}) - \boldsymbol{p}_{j+1}^{\text{fwd}}\|_2 \leq C_2^{\text{rev}}(T-(j+1))^2 \Delta t^2, & \text{(27b)} \\[4pt] \|R(\dot{\boldsymbol{p}}_{j+1}^{\text{rev}}) - \dot{\boldsymbol{p}}_{j+1}^{\text{fwd}}\|_2 \leq C_1^{\text{rev}}(T-(j+1)) \Delta t, & \text{(27c)} \end{cases}$$

Inductive Proof: We need to prove (26b) (26a) and (26c) holds for $k = j$.

First, for (26c), using (24c) and (25c), we get for any j that

$$\begin{aligned} \|R(\dot{\boldsymbol{p}}_j^{\text{rev}}) - \dot{\boldsymbol{p}}_j^{\text{fwd}}\|_2 &= \|(\dot{\boldsymbol{p}}_{j+1}^{\text{rev}} + \mathbf{rem}_j^{\text{rev},1}) - (\dot{\boldsymbol{p}}_{j+1}^{\text{fwd}} + \mathbf{rem}_j^{\text{fwd},1})\|_2 \\ &\leq \|R(\dot{\boldsymbol{p}}_{j+1}^{\text{rev}}) - \dot{\boldsymbol{p}}_{j+1}^{\text{fwd}}\|_2 + \|\mathbf{rem}_j^{\text{rev},1}\|_2 + \|\mathbf{rem}_j^{\text{fwd},1}\|_2 \\ &\leq C_1^{\text{rev}}(T-j-1)\Delta t + 2c_1 \Delta t, \end{aligned}$$

where the first inequality uses the triangle inequality, and the second inequality plugs in (27c). Thus taking $C_1^{\text{rev}} = 2c_1$, the above is upped bounded by $C_1^{\text{rev}}(T-j)\Delta t$, and (26b) holds for j.

Second, for (27b), using (24b) and (25b), we get

$$\begin{aligned} \|R(\boldsymbol{p}_j^{\text{rev}}) - \boldsymbol{p}_j^{\text{fwd}}\|_2 &= \| -(\boldsymbol{p}_{j+1}^{\text{rev}} + \dot{\boldsymbol{p}}_{j+1}^{\text{rev}}\Delta t + \mathbf{rem}_j^{\text{rev},2}) - (\boldsymbol{p}_{j+1}^{\text{fwd}} - \dot{\boldsymbol{p}}_{j+1}^{\text{fwd}}\Delta t + \mathbf{rem}_j^{\text{fwd},2})\|_2 \\ &\leq \|R(\boldsymbol{p}_{j+1}^{\text{rev}}) - \boldsymbol{p}_{j+1}^{\text{fwd}}\|_2 + \|R(\dot{\boldsymbol{p}}_{j+1}^{\text{rev}}) - \dot{\boldsymbol{p}}_{j+1}^{\text{fwd}}\|_2 \Delta t \\ &\quad + \|\mathbf{rem}_j^{\text{rev},2}\|_2 + \|\mathbf{rem}_j^{\text{fwd},2}\|_2 \\ &\leq \left[C_2^{\text{rev}}(T-j-1)^2 + C_1^{\text{rev}}(T-j-1) + 2c_2 \right] \Delta t^2, \end{aligned}$$

where the first inequality uses the triangle inequality, and in the second inequality we use (27a) and (27b). Thus taking $C_2^{\text{rev}} = \max\{C_1^{\text{rev}}/2, 2c_2\}$, we have the final expression above is upper bounded by $C_2^{\text{rev}}(T-j)^2 \Delta t^2$, and so the claim holds for j.

Finally, for (27a), we use (24a) and (25a) to get

$$\begin{aligned} \|R(\boldsymbol{q}_j^{\text{rev}}) - \boldsymbol{q}_j^{\text{fwd}}\|_2 &= \|(\boldsymbol{q}_{j+1}^{\text{rev}} + (\boldsymbol{p}_{j+1}^{\text{rev}}/m)\Delta t + (\dot{\boldsymbol{p}}_{j+1}^{\text{rev}}/2m)\Delta t^2 + \mathbf{rem}_j^{\text{rev},3}) \\ &\quad - (\boldsymbol{q}_{j+1}^{\text{fwd}} - (\boldsymbol{p}_{j+1}^{\text{fwd}}/m)\Delta t + (\dot{\boldsymbol{p}}_{j+1}^{\text{fwd}}/2m)\Delta t^2 + \mathbf{rem}_j^{\text{fwd},3})\|_2 \\ &\leq \|R(\boldsymbol{q}_{j+1}^{\text{rev}}) - \boldsymbol{q}_{j+1}^{\text{fwd}}\|_2 + \frac{1}{m}\|R(\boldsymbol{p}_{j+1}^{\text{rev}}) - \boldsymbol{p}_{j+1}^{\text{fwd}}\|_2 \Delta t \\ &\quad + \frac{1}{2m}\|R(\dot{\boldsymbol{p}}_{j+1}^{\text{rev}}) - \dot{\boldsymbol{p}}_{j+1}^{\text{fwd}}\|_2 \Delta t^2 + \|\mathbf{rem}_j^{\text{rev},3}\|_2 + \|\mathbf{rem}_j^{\text{fwd},3}\|_2 \\ &\leq \left[C_3^{\text{rev}}(T-j-1)^3 + \frac{C_2^{\text{rev}}}{m}(T-j-1)^2 + \frac{C_1^{\text{rev}}}{2m}(T-j-1) + 2c_3 \right] \Delta t^3, \end{aligned}$$

where the first inequality uses the triangle inequality, and in the second inequality we use (27a), (27b) and (27c). Thus taking $C_3^{\text{rev}} = \max\{C_2^{\text{rev}}/3m, C_1^{\text{rev}}/6m, 2c_3\}$, we have the final expression above is upper bounded by $C_3^{\text{rev}}(T-j)^3 \Delta t^3$, and so the claim holds for j.

Since both the base case and the inductive step have been proven, by the principle of mathematical induction, (26b), (26a) and (26c) hold for all $k = T, T-1, \cdots, 0$.

With this we finish the proof by plugging (26b) and (26a) into the loss function:

$$\sum_{j=0}^{T} \|R(\boldsymbol{z}_j^{\text{rev}}) - \boldsymbol{z}_j^{\text{fwd}}\|_2^2 = \sum_{j=0}^{T} \|R(\boldsymbol{p}_j^{\text{rev}}) - \boldsymbol{p}_j^{\text{fwd}}\|_2^2 + \sum_{j=0}^{T} \|R(\boldsymbol{q}_j^{\text{rev}}) - \boldsymbol{q}_j^{\text{fwd}}\|_2^2$$

$$\leq \left(C_2^{\text{rev}}\right)^2 \sum_{j=0}^{T} (T-j)^4 \Delta t^4 + \left(C_3^{\text{rev}}\right)^2 \sum_{j=0}^{T} (T-j)^6 \Delta t^6$$

$$= \mathcal{O}(T^5 \Delta t^4).$$

B.3 Analysis on Implementations of Reversal Loss

Remark 1. We define the reversal loss $\mathcal{L}_{reverse}$ as the distance between model forward trajectories and backward trajectories. There are other implementation choices. The first is to minimize the distances between model backward trajectories and ground truth trajectories. When both forward and backward trajectories are close to ground-truth, they are implicitly symmetric. The major drawback is that at the early stage of learning when the forward is far away from ground truth, such implicit regularization does not force time-reversal symmetry, but introduce more noise.

Remark 2. The computational time of the reversal loss $\mathcal{L}_{reverse}$ is of the same scale as the forward reconstruction loss $\mathcal{L}_{pred}$. As the computation process of the reversal loss is to first use the ODE solver to generate the reverse trajectories, which has the same computational overhead as computing the forward trajectories, and then compute the L2 distances. The space complexity is only $\mathcal{O}(1)$ as we are using the adjoint methods (Chen et al., 2018) to compute the reverse trajectories.

Remark 3. Comparing TANGO against the implementation following Eqn.(2), when the reconstruction loss defined in Eqn.(4) and the time-reversal loss defined in Eqn. (6) of both methods are equal, the maximum error between the reversal and ground truth trajectory, i.e. $MaxError_{gt\_rev} = \max_{j \in [T]} \|\boldsymbol{y}(j) - \hat{\boldsymbol{y}}^{\text{rev}}(T-j)\|_2$ made by TANGO is smaller than that of following Eqn.(2).We show the empirical comparison of them in Appendix D.5.
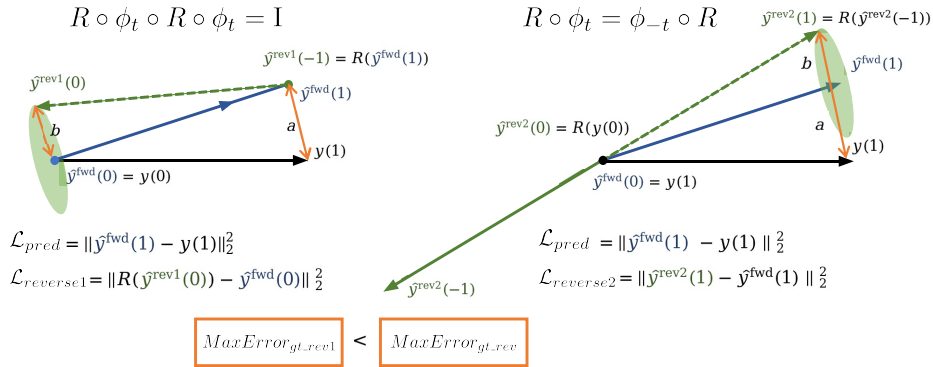

Figure 3: Comparison between two reversal loss implementation

This remark suggests that our implementation of time-reversal symmetry is numerically better than the implementation used in (Huh et al., 2020). We give an illustration of the remark below.

We expect an ideal model to align both the predicted forward and reverse trajectories with the ground truth. As shown in Figure 3, we integrate one step from the initial state $\hat{\boldsymbol{y}}^{\text{fwd}}(0)$ (which is the same as $\boldsymbol{y}(0)$) and reach the state $\hat{\boldsymbol{y}}^{\text{fwd}}(1)$.

The first reverse loss implementation (ours) follows lemma (1) as $R \circ \Phi_t \circ R \circ \Phi_t = id$, which means when we evolve forward and reach the state $\hat{\boldsymbol{y}}^{\text{fwd}}(1)$ we reverse it into $\hat{\boldsymbol{y}}^{\text{rev1}}(-1)$ and go back to reach $\hat{\boldsymbol{y}}^{\text{rev1}}(0)$, then reverse it to get $R(\hat{\boldsymbol{y}}^{\text{rev1}}(0))$, which ideally should be the same as $\hat{\boldsymbol{y}}^{\text{fwd}}(0)$.

The second reverse loss implementation follows Eqn.(2) as $R \circ \Phi_t = \Phi_{-t} \circ R$, which means we first reverse the initial state as $\hat{\boldsymbol{y}}^{\text{rev2}}(0) = R(\boldsymbol{y}(0))$, then evolve the reverse trajectory in the opposite di-

rection to reach $\hat{\boldsymbol{y}}^{\mathrm{rev2}}(-1)$, and then perform a time-symmetric operation to reach $\hat{\boldsymbol{y}}^{\mathrm{rev2}}(1)$, aligning it with the forward trajectory.

We assume the two reconstruction losses $\mathcal{L}_{pred} = \|\hat{\boldsymbol{y}}^{\mathrm{fwd}}(1) - \boldsymbol{y}(1)\|_2^2 := a$ are the same. For the time-reversal losses:

$$\mathcal{L}_{reverse1} = \|R(\hat{\boldsymbol{y}}^{\mathrm{rev1}}(0)) - \hat{\boldsymbol{y}}^{\mathrm{fwd}}(0)\|_2^2 + \|R(\hat{\boldsymbol{y}}^{\mathrm{rev1}}(-1)) - \hat{\boldsymbol{y}}^{\mathrm{fwd}}(1)\|_2^2 = \|R(\hat{\boldsymbol{y}}^{\mathrm{rev1}}(0)) - \hat{\boldsymbol{y}}^{\mathrm{fwd}}(0)\|_2^2 := b,$$

$$\mathcal{L}_{reverse2} = \|\hat{\boldsymbol{y}}^{\mathrm{rev2}}(0) - \hat{\boldsymbol{y}}^{\mathrm{fwd}}(0)\|_2^2 + \|\hat{\boldsymbol{y}}^{\mathrm{rev2}}(1) - \hat{\boldsymbol{y}}^{\mathrm{fwd}}(1)\|_2^2 = \|\hat{\boldsymbol{y}}^{\mathrm{rev2}}(1) - \hat{\boldsymbol{y}}^{\mathrm{fwd}}(1)\|_2^2 := b,$$

we also assume they have reached the same value b.

As shown in Figure.3 where we illustrate the worse case scenario $MaxError_{gt\_rev} = \max_{j \in [T]} \|\boldsymbol{y}(j) - \hat{\boldsymbol{y}}^{\mathrm{rev}}(T-j)\|_2$ of the two loss, we can see that in our implementation the worst error is the maximum of two loss, while the TRS-ODEN's implementation has the risk of accumulating the error together, making the worst error being the sum of both:

$$MaxError_{gt\_rev1} = \max\left\{\left\|R(\hat{\boldsymbol{y}}^{\mathrm{rev1}}(0)) - \boldsymbol{y}(0)\right\|_2, \left\|R(\hat{\boldsymbol{y}}^{\mathrm{rev1}}(-1)) - \boldsymbol{y}(1)\right\|_2\right\} = max\{a, b\},$$

$$MaxError_{gt\_rev2} = \left\|\hat{\boldsymbol{y}}^{\mathrm{rev2}}(1) - \hat{\boldsymbol{y}}^{\mathrm{fwd}}(1)\right\|_2 + \left\|\hat{\boldsymbol{y}}^{\mathrm{fwd}}(1) - \boldsymbol{y}(1)\right\|_2 = a + b,$$

This means our model achieves a smaller error of the maximum distance between the reversal and ground truth trajectory.

C   Example of varying dynamical systems

We illustrate the energy conservation and time reversal of the three n-body spring systems in Figure 1(a). We use the Hamiltonian formalism of systems under classical mechanics to describe their dynamics and verify their energy conservation and time-reversibility characteristics.

The scalar function that describes a system's motion is called the Hamiltonian, $\mathcal{H}$, and is typically equal to the total energy of the system, that is, the potential energy plus the kinetic energy (North, 2021). It describes the phase space equations of motion by following two first-order ODEs called Hamilton's equations:

$$\frac{d\mathbf{q}}{dt} = \frac{\partial\mathcal{H}(\mathbf{q}, \mathbf{p})}{\partial\mathbf{p}}, \frac{d\mathbf{p}}{dt} = -\frac{\partial\mathcal{H}(\mathbf{q}, \mathbf{p})}{\partial\mathbf{q}}, \tag{28}$$

where $\mathbf{q} \in \mathbb{R}^n, \mathbf{p} \in \mathbb{R}^n$, and $\mathcal{H} : \mathbb{R}^{2n} \mapsto \mathbb{R}$ are positions, momenta, and Hamiltonian of the system.

Under this formalism, energy conservative is defined by $d\mathcal{H}/dt = 0$, and the time-reversal symmetry is defined by $\mathcal{H}(q, p, t) = \mathcal{H}(q, -p, -t)$ (Lamb & Roberts, 1998).

C.1   Conservative and reversible systems.

A simple example is the isolated n-body spring system, which can be described by :

$$\begin{aligned} \frac{d\mathbf{q_i}}{dt} &= \frac{\mathbf{p_i}}{m} \\ \frac{d\mathbf{p_i}}{dt} &= \sum_{j \in N_i} -k(\mathbf{q_i} - \mathbf{q_j}), \end{aligned} \tag{29}$$

where $\mathbf{q} = (\mathbf{q_1}, \mathbf{q_2}, \cdots, \mathbf{q_N})$ is a set of positions of each object , $\mathbf{p} = (\mathbf{p_1}, \mathbf{p_2}, \cdots, \mathbf{p_N})$ is a set of momenta of each object, m_i is mass of each object, k is spring constant.

The Hamilton's equations are:

$$\begin{aligned} \frac{\partial\mathcal{H}(\mathbf{q}, \mathbf{p})}{\partial\mathbf{p_i}} &= \frac{d\mathbf{q_i}}{dt} = \frac{\mathbf{p_i}}{m} \\ \frac{\partial\mathcal{H}(\mathbf{q}, \mathbf{p})}{\partial\mathbf{q_i}} &= -\frac{d\mathbf{p_i}}{dt} = \sum_{j \in N_i} k(\mathbf{q_i} - \mathbf{q_j}), \end{aligned} \tag{30}$$

Hence, we can obtain the Hamiltonian through the integration of the above equation.

$$\mathcal{H}(\mathbf{q}, \mathbf{p}) = \sum_{i=1}^{N} \frac{\mathbf{p_i}^2}{2m_i} + \frac{1}{2} \sum_{i=1}^{N} \sum_{j \in N_i}^{N} \frac{1}{2}k(\mathbf{q_i} - \mathbf{q_j})^2, \tag{31}$$

Verify the systems' energy conservation

$$\frac{d\mathcal{H}(\mathbf{q},\mathbf{p})}{dt} = \frac{1}{dt}\left(\sum_{i=1}^{N}\frac{\mathbf{p_i}^2}{2m_i}\right) + \frac{1}{dt}\left(\frac{1}{2}\sum_{i=1}^{N}\sum_{j\in N_i}^{N}\frac{1}{2}k(\mathbf{q_i}-\mathbf{q_j})^2\right) = 0, \tag{32}$$

So it is conservative.

Verify the systems' time-reversal symmetry We do the transformation $R : (\mathbf{q},\mathbf{p},t) \mapsto (\mathbf{q},-\mathbf{p},-t)$.

$$\mathcal{H}(\mathbf{q},\mathbf{p}) = \sum_{i=1}^{N}\frac{\mathbf{p_i}^2}{2m_i} + \frac{1}{2}\sum_{i=1}^{N}\sum_{j\in N_i}^{N}\frac{1}{2}k(\mathbf{q_i}-\mathbf{q_j})^2,$$

$$\mathcal{H}(\mathbf{q},-\mathbf{p}) = \sum_{i=1}^{N}\frac{(-\mathbf{p_i})^2}{2m_i} + \frac{1}{2}\sum_{i=1}^{N}\sum_{j\in N_i}^{N}\frac{1}{2}k(\mathbf{q_i}-\mathbf{q_j})^2, \tag{33}$$

It is obvious $\mathcal{H}(\mathbf{q},\mathbf{p}) = \mathcal{H}(\mathbf{q},-\mathbf{p})$, so it is reversible

C.2 Non-conservative and reversible systems.

A simple example is a n-body spring system with periodical external force, which can be described by:

$$\frac{d\mathbf{q_i}}{dt} = \frac{\mathbf{p_i}}{m}$$

$$\frac{d\mathbf{p_i}}{dt} = \sum_{j\in N_i}^{N}-k(\mathbf{q_i}-\mathbf{q_j}) - k_1\cos\omega t, \tag{34}$$

The Hamilton's equations are:

$$\frac{\partial\mathcal{H}(\mathbf{q},\mathbf{p})}{\partial\mathbf{p_i}} = \frac{d\mathbf{q_i}}{dt} = \frac{\mathbf{p_i}}{m}$$

$$\frac{\partial\mathcal{H}(\mathbf{q},\mathbf{p})}{\partial\mathbf{q_i}} = -\frac{d\mathbf{p_i}}{dt} = \sum_{j\in N_i}k(\mathbf{q_i}-\mathbf{q_j}) + k_1\cos\omega t, \tag{35}$$

Hence, we can obtain the Hamiltonian through the integration of the above equation:

$$\mathcal{H}(\mathbf{q},\mathbf{p}) = \sum_{i=1}^{N}\frac{\mathbf{p_i}^2}{2m_i} + \frac{1}{2}\sum_{i=1}^{N}\sum_{j\in N_i}^{N}\frac{1}{2}k(\mathbf{q_i}-\mathbf{q_j})^2 + \sum_{i=1}^{N}q_i * k_1\cos\omega t, \tag{36}$$

Verify the systems' energy conservation

$$\frac{d\mathcal{H}(\mathbf{q},\mathbf{p})}{dt} = \frac{1}{dt}\left(\sum_{i=1}^{N}\frac{\mathbf{p_i}^2}{2m_i}\right) + \frac{1}{dt}\left(\frac{1}{2}\sum_{i=1}^{N}\sum_{j\in N_i}^{N}\frac{1}{2}k(\mathbf{q_i}-\mathbf{q_j})^2\right) + \frac{1}{dt}\left(\sum_{i=1}^{N}q_i * k_1\cos\omega t\right)$$

$$= 0 + \frac{1}{dt}\left(\sum_{i=1}^{N}q_ik_1\cos\omega t\right) \tag{37}$$

$$= \left(\sum_{i=1}^{N}-\omega q_ik_1\sin\omega t\right) \neq 0$$

So it is non-conservative.

Verify the systems' time-reversal symmetry We do the transformation $R : (\mathbf{q},\mathbf{p},t) \mapsto (\mathbf{q},-\mathbf{p},-t)$.

$$\mathcal{H}(\mathbf{q},\mathbf{p}) = \sum_{i=1}^{N}\frac{\mathbf{p_i}^2}{2m_i} + \frac{1}{2}\sum_{i=1}^{N}\sum_{j\in N_i}^{N}\frac{1}{2}k(\mathbf{q_i}-\mathbf{q_j})^2 + \sum_{i=1}^{N}q_i * k_1\cos\omega t,$$

$$\mathcal{H}(\mathbf{q},-\mathbf{p}) = \sum_{i=1}^{N}\frac{(-\mathbf{p_i})^2}{2m_i} + \frac{1}{2}\sum_{i=1}^{N}\sum_{j\in N_i}^{N}\frac{1}{2}k(\mathbf{q_i}-\mathbf{q_j})^2 + \sum_{i=1}^{N}q_i * k_1\cos\omega(-t), \tag{38}$$

It is obvious $\mathcal{H}(\mathbf{q},\mathbf{p},t) = \mathcal{H}(\mathbf{q},-\mathbf{p},t)$, so it is reversible

C.3 Non-conservative and irreversible systems.

A simple example is an n-body spring system with frictions proportional to its velocity,γ is the coefficient of friction, which can be described by:

$$
\begin{aligned}
\frac{d\mathbf{q}_i}{dt} &= \frac{\mathbf{p}_i}{m} \\
\frac{d\mathbf{p}_i}{dt} &= -k_0\mathbf{q}_i - \gamma\frac{\mathbf{p}_i}{m}
\end{aligned} \tag{39}
$$

The Hamilton's equations are:

$$
\begin{aligned}
\frac{\partial\mathcal{H}(\mathbf{q},\mathbf{p})}{\partial\mathbf{p_i}} &= \frac{d\mathbf{q_i}}{dt} = \frac{\mathbf{p_i}}{m} \\
\frac{\partial\mathcal{H}(\mathbf{q},\mathbf{p})}{\partial\mathbf{q_i}} &= -\frac{d\mathbf{p_i}}{dt} = \sum_{j\in N_i} k(\mathbf{q_i}-\mathbf{q_j}) + \gamma\frac{\mathbf{p_i}}{m}
\end{aligned} \tag{40}
$$

Hence, we can obtain the Hamiltonian through the integration of the above equation:

$$
\mathcal{H}(\mathbf{q},\mathbf{p}) = \sum_{i=1}^{N}\frac{\mathbf{p_i}^2}{2m_i} + \frac{1}{2}\sum_{i=1}^{N}\sum_{j\in N_i}^{N}\frac{1}{2}k(\mathbf{q_i}-\mathbf{q_j})^2 + \sum_{i=1}^{N}\frac{\gamma}{m}\int_0^t\frac{\mathbf{p_i}^2}{m}dt, \tag{41}
$$

Verify the systems' energy conservation

$$
\begin{aligned}
&\frac{d\mathcal{H}(\mathbf{q},\mathbf{p})}{dt} \\
=&\frac{1}{dt}\Big(\sum_{i=1}^{N}\frac{\mathbf{p_i}^2}{2m_i}\Big) + \frac{1}{dt}\Big(\frac{1}{2}\sum_{i=1}^{N}\sum_{j\in N_i}^{N}\frac{1}{2}k(\mathbf{q_i}-\mathbf{q_j})^2\Big) + \frac{1}{dt}\Big(\sum_{i=1}^{N}\frac{\gamma}{m}\int_0^t\frac{\mathbf{p_i}^2}{m}dt\Big) \\
=&0 + \frac{1}{dt}\Big(\sum_{i=1}^{N}\frac{\gamma}{m}\int_0^t\frac{\mathbf{p_i}^2}{m}dt\Big) \\
=&\Big(\sum_{i=1}^{N}\frac{\gamma}{m}\frac{\mathbf{p_i}^2}{m}\Big) \neq 0
\end{aligned} \tag{42}
$$

So it is non-conservative.

Verify the systems' time-reversal symmetry We do the transformation $R : (\mathbf{q},\mathbf{p},t) \mapsto (\mathbf{q},-\mathbf{p},-t)$.

$$
\begin{aligned}
\mathcal{H}(\mathbf{q},\mathbf{p}) &= \sum_{i=1}^{N}\frac{\mathbf{p_i}^2}{2m_i} + \frac{1}{2}\sum_{i=1}^{N}\sum_{j\in N_i}^{N}\frac{1}{2}k(\mathbf{q_i}-\mathbf{q_j})^2 + \sum_{i=1}^{N}\frac{\gamma}{m}\int_0^t\frac{\mathbf{p_i}^2}{m}dt, \\
\mathcal{H}(\mathbf{q},-\mathbf{p}) &= \sum_{i=1}^{N}\frac{(-\mathbf{p_i})^2}{2m_i} + \frac{1}{2}\sum_{i=1}^{N}\sum_{j\in N_i}^{N}\frac{1}{2}k(\mathbf{q_i}-\mathbf{q_j})^2 + \sum_{i=1}^{N}\frac{\gamma}{m}\int_0^{(-t)}\frac{\mathbf{p_i}^2}{m}d(-t),
\end{aligned} \tag{43}
$$

It is obvious $\mathcal{H}(\mathbf{q},\mathbf{p},t) \neq \mathcal{H}(\mathbf{q},-\mathbf{p},t)$, so it is irreversible

D Experiments

D.1 Dataset

In our experiments, all datasets are synthesized from ground-truth physical law via sumulation. We generate four simulated datasets: three n-body spring systems under damping, periodic, or no external force, and one chaotic tripe pendulum dataset with three sequentially connected stiff sticks that form. We name the first three as *Sipmle Spring*, *Forced Spring*, and *Damped Spring* respectively. All n-body spring systems contain 5 interacting balls, with varying connectivities. Each *Pendulum* system contains 3 connected stiff sticks.

For the n-body spring system, we randomly sample whether a pair of objects are connected, and model their interaction via forces defined by Hooke's law. In the *Damped spring*, the objects have an additional friction force that is opposite to their moving direction and whose magnitude is proportional to their speed. In the *Forced spring*, all objects have the same external force that changes direction periodically. We show in Figure 1(a), the energy variation in both of the *Damped spring* and *Forced spring* is significant.

For the chaotic triple *Pendulum* , the equations governing the motion are inherently nonlinear. Although this system is deterministic, it is also highly sensitive to the initial condition and numerical errors (Shinbrot et al., 1992; Awrejcewicz et al., 2008; Stachowiak & Okada, 2006). This property is often referred to as the "butterfly effect", as depicted in Fig 4. Unlike for n-body spring systems, where the forces and equations of motion can be easily articulated, for the *Pendulum*, the explicit forces cannot be directly defined, and the motion of objects can only be described through Lagrangian formulations North (2021), making the modeling highly complex and raising challenges for accurate learning.
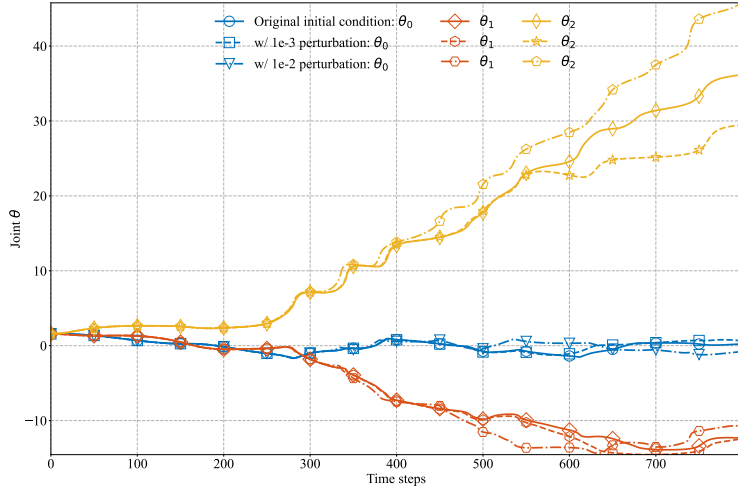

Figure 4: Illustration to show the pendulum is highly-sensitive to initial states

We simulate the trajectories by using Euler's method for n-body spring systems and using the 4th order Runge-Kutta (RK4) method for the *Pendulum*. We integrate with a fixed step size and subsample every 100 steps. For training, we use a total of 6000 forward steps. To generate irregularly sampled partial observations, we follow Huang et al. (2020) and sample the number of observations n from a uniform distribution U(40, 52) and draw the n observations uniformly for each object. For testing, we additionally sample 40 observations following the same procedure from PDE steps [6000, 12000], besides generating observations from steps [1, 6000]. The above sampling procedure is conducted independently for each object. We generate 20k training samples and 5k testing samples for each dataset. The features (position/velocity) are normalized to the maximum absolute value of 1 across training and testing datasets.

In the following subsections, we show the dynamical equations of each dataset in detail.

D.1.1 spring

Simple spring. The dynamical equations of *simple spring* are as follows:

$$\frac{d\mathbf{q_i}}{dt} = \frac{\mathbf{p_i}}{m}$$
$$\frac{d\mathbf{p_i}}{dt} = \sum_{j \in N_i}^{N} -k(\mathbf{q_i} - \mathbf{q_j}) \tag{44}$$

where where $\mathbf{q} = (\mathbf{q_1}, \mathbf{q_2}, \cdots, \mathbf{q_N})$ is a set of positions of each object , $\mathbf{p} = (\mathbf{p_1}, \mathbf{p_2}, \cdots, \mathbf{p_N})$ is a set of momenta of each object. We set the mass of each object $m = 1$, the spring constant $k = 0.1$.

16

Damped spring. The dynamical equations of *damped spring* are as follows:

$$\frac{d\mathbf{q_i}}{dt} = \frac{\mathbf{p_i}}{m}$$
$$\frac{d\mathbf{p_i}}{dt} = \sum_{j \in N_i} -k(\mathbf{q_i} - \mathbf{q_j}) - \gamma\frac{\mathbf{p}_i}{m} \tag{45}$$

where where $\mathbf{q} = (\mathbf{q_1}, \mathbf{q_2}, \cdots, \mathbf{q_N})$ is a set of positions of each object, $\mathbf{p} = (\mathbf{p_1}, \mathbf{p_2}, \cdots, \mathbf{p_N})$ is a set of momenta of each object, We set the mass of each object $m = 1$, the spring constant$k = 0.1$, the coefficient of friction $\gamma = 10$.

Forced spring. The dynamical equations of *forced spring* system are as follows:

$$\frac{d\mathbf{q_i}}{dt} = \frac{\mathbf{p_i}}{m}$$
$$\frac{d\mathbf{p_i}}{dt} = \sum_{j \in N_i}^{N} -k(\mathbf{q_i} - \mathbf{q_j}) - k_1 \cos\omega t, \tag{46}$$

where where $\mathbf{q} = (\mathbf{q_1}, \mathbf{q_2}, \cdots, \mathbf{q_N})$ is a set of positions of each object , $\mathbf{p} = (\mathbf{p_1}, \mathbf{p_2}, \cdots, \mathbf{p_N})$ is a set of momenta of each object. We set the mass of each object $m = 1$, the spring constant$k = 0.1$, the external strength $k_1 = 10$ and the frequency of variation $\omega = 1$

We simulate the positions and momentums of three spring systems by using Euler methods as follows:

$$\mathbf{q_i}(t + 1) = \mathbf{q_i}(t) + \frac{d\mathbf{q_i}}{dt}\Delta t$$
$$\mathbf{p_i}(t + 1) = \mathbf{p_i}(t) + \frac{d\mathbf{p_i}}{dt}\Delta t \tag{47}$$

where $\frac{d\mathbf{q_i}}{dt}$ and $\frac{d\mathbf{p_i}}{dt}$ were defined as above for each datasets, and $\Delta t = 0.001$ is the integration steps.

D.1.2 chaotic pendulum

In this section, we demonstrate how to derive the dynamics equations for a chaotic triple pendulum using the Lagrangian formalism.

The moment of inertia of each stick about the centroid is

$$I = \frac{1}{12}ml^2 \tag{48}$$

The position of the center of gravity of each stick is as follows:

$$x_1 = \frac{l}{2}\sin\theta_1, \quad y_1 = -\frac{l}{2}\cos\theta_1$$
$$x_2 = l(\sin\theta_1 + \frac{1}{2}\sin\theta_2), \quad y_2 = -l(\cos\theta_1 + \frac{1}{2}\cos\theta_2) \tag{49}$$
$$x_3 = l(\sin\theta_1 + \sin\theta_2 + \frac{1}{2}\sin\theta_3), \quad y_3 = -l(\cos\theta_1 + \cos\theta_2 + \frac{1}{2}\cos\theta_3)$$

The change in the center of gravity of each stick is:

$$\dot{x}_1 = \frac{l}{2}\cos\theta_1 \cdot \dot{\theta}_1, \quad \dot{y}_1 = \frac{l}{2}\sin\theta_1 \cdot \dot{\theta}_1$$
$$\dot{x}_2 = l(\cos\theta_1 \cdot \dot{\theta}_1 + \frac{1}{2}\cos\theta_2 \cdot \dot{\theta}_2), \quad \dot{y}_2 = l(\sin\theta_1 \cdot \dot{\theta}_1 + \frac{1}{2}\sin\theta_2 \cdot \dot{\theta}_2) \tag{50}$$
$$\dot{x}_3 = l(\cos\theta_1 \cdot \dot{\theta}_1 + \cos\theta_2 \cdot \dot{\theta}_2 + \frac{1}{2}\cos\theta_3 \cdot \dot{\theta}_3), \quad \dot{y}_3 = l(\sin\theta_1 \cdot \dot{\theta}_1 + \sin\theta_2 \cdot \dot{\theta}_2 + \frac{1}{2}\sin\theta_3 \cdot \dot{\theta}_3)$$

The Lagrangian L of this triple pendulum system is:

$$\begin{aligned}
\mathcal{L} =& T - V \\
=& \frac{1}{2}m(\dot{x_1}^2 + \dot{x_2}^2 + \dot{x_3}^2 + \dot{y_1}^2 + \dot{y_2}^2 + \dot{y_3}^2) + \frac{1}{2}I(\dot{\theta_1}^2 + \dot{\theta_2}^2 + \dot{\theta_3}^2) - mg(y_1 + y_2 + y_3) \\
=& \frac{1}{6}ml(9\dot{\theta}_2\dot{\theta}_1 l\cos(\theta_1 - \theta_2) + 3\dot{\theta}_3\dot{\theta}_1 l\cos(\theta_1 - \theta_3) + 3\dot{\theta}_2\dot{\theta}_3 l\cos(\theta_2 - \theta_3) + 7\dot{\theta}_1^2 l + 4\dot{\theta}_2^2 l + \dot{\theta}_3^2 l \\
& + 15g\cos(\theta_1) + 9g\cos(\theta_2) + 3g\cos(\theta_3))
\end{aligned} \tag{51}$$

The Lagrangian equation is defined as follows:

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\theta}} - \frac{\partial \mathcal{L}}{\partial \theta} = \mathbf{0} \tag{52}$$

and we also have:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \dot{\theta}} &= \frac{\partial T}{\partial \dot{\theta}} = p \\
\dot{p} &= \frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\theta}} = \frac{\partial \mathcal{L}}{\partial \theta}
\end{aligned} \tag{53}$$

where p is the Angular Momentum.
We can list the equations for each of the three sticks separately:

$$\begin{aligned}
p_1 &= \frac{\partial \mathcal{L}}{\partial \dot{\theta}_{\mathbf{1}}} & \dot{p}_1 &= \frac{\partial \mathcal{L}}{\partial \theta_{\mathbf{1}}} \\
p_2 &= \frac{\partial \mathcal{L}}{\partial \dot{\theta}_{\mathbf{2}}} & \dot{p}_2 &= \frac{\partial \mathcal{L}}{\partial \theta_{\mathbf{2}}} \\
p_3 &= \frac{\partial \mathcal{L}}{\partial \dot{\theta}_{\mathbf{3}}} & \dot{p}_3 &= \frac{\partial \mathcal{L}}{\partial \theta_{\mathbf{3}}}
\end{aligned} \tag{54}$$

Finally, we have :

$$\begin{cases}
\dot{\theta}_1 = \frac{6(9p_1\cos(2(\theta_2-\theta_3))+27p_2\cos(\theta_1-\theta_2)-9p_2\cos(\theta_1+\theta_2-2\theta_3)+21p_3\cos(\theta_1-\theta_3)-27p_3\cos(\theta_1-2\theta_2+\theta_3)-23p_1)}{ml^2(81\cos(2(\theta_1-\theta_2))-9\cos(2(\theta_1-\theta_3))+45\cos(2(\theta_2-\theta_3))-169)} \\
\dot{\theta}_2 = \frac{6(27p_1\cos(\theta_1-\theta_2)-9p_1\cos(\theta_1+\theta_2-2\theta_3)+9p_2\cos(2(\theta_1-\theta_3))-27p_3\cos(2\theta_1-\theta_2-\theta_3)+57p_3\cos(\theta_2-\theta_3)-47p_2)}{ml^2(81\cos(2(\theta_1-\theta_2))-9\cos(2(\theta_1-\theta_3))+45\cos(2(\theta_2-\theta_3))-169)} \\
\dot{\theta}_3 = \frac{6(21p_1\cos(\theta_1-\theta_3)-27p_1\cos(\theta_1-2\theta_2+\theta_3)-27p_2\cos(2\theta_1-\theta_2-\theta_3)+57p_2\cos(\theta_2-\theta_3)+81p_3\cos(2(\theta_1-\theta_2))-143p_3)}{ml^2(81\cos(2(\theta_1-\theta_2))-9\cos(2(\theta_1-\theta_3))+45\cos(2(\theta_2-\theta_3))-169)} \\
\dot{p}_1 = -\frac{1}{2}ml\left(3\dot{\theta}_2\dot{\theta}_1 l\sin(\theta_1-\theta_2) + \dot{\theta}_1\dot{\theta}_3 l\sin(\theta_1-\theta_3) + 5g\sin(\theta_1)\right) \\
\dot{p}_1 = -\frac{1}{2}ml\left(-3\dot{\theta}_1\dot{\theta}_2 l\sin(\theta_1-\theta_2) + \dot{\theta}_2\dot{\theta}_3 l\sin(\theta_2-\theta_3) + 3g\sin(\theta_2)\right) \\
\dot{p}_1 = -\frac{1}{2}ml\left(\dot{\theta}_1\dot{\theta}_3 l\sin(\theta_1-\theta_3) + \dot{\theta}_2\dot{\theta}_3 l\sin(\theta_2-\theta_3) - g\sin(\theta_3)\right)
\end{cases} \tag{55}$$

We simulate the angular of the three sticks by using the Runge-Kutta 4th Order Method as follows:

$$\begin{aligned}
\Delta\boldsymbol{\theta}^1(t) &= \dot{\boldsymbol{\theta}}(t, \boldsymbol{\theta}(t)) \cdot \Delta t \\
\Delta\boldsymbol{\theta}^2(t) &= \dot{\boldsymbol{\theta}}(t + \frac{\Delta t}{2}, \boldsymbol{\theta}(t) + \frac{\Delta\boldsymbol{\theta}^1(t)}{2}) \cdot \Delta t \\
\Delta\boldsymbol{\theta}^3(t) &= \dot{\boldsymbol{\theta}}(t + \frac{\Delta t}{2}, \boldsymbol{\theta}(t) + \frac{\Delta\boldsymbol{\theta}^2(t)}{2}) \cdot \Delta t \\
\Delta\boldsymbol{\theta}^4(t) &= \dot{\boldsymbol{\theta}}(t + \Delta t, \boldsymbol{\theta}(t) + \Delta\boldsymbol{\theta}^3(t)) \cdot \Delta t \\
\Delta\boldsymbol{\theta}(t) &= \frac{1}{6}(\Delta\boldsymbol{\theta}^1(t) + \Delta\boldsymbol{\theta}^2(t) + \Delta\boldsymbol{\theta}^3(t) + \Delta\boldsymbol{\theta}^4(t)) \\
\boldsymbol{\theta}(t+1) &= \boldsymbol{\theta}(t) + \Delta\boldsymbol{\theta}(t)
\end{aligned} \tag{56}$$

where $\dot{\boldsymbol{\theta}}$ was defined as above , and $\Delta t = 0.0001$ is the integration steps.

D.2 Task Setup

We evaluate model performance by splitting the trajectories into two halves: $[t_1, t_K]$, $[t_{K+1}, t_T]$ where timestamps can be irregular. We condition the first half of observations to make predictions for the second half as in Rubanova et al. (2019). For training, we condition from $[t_1, t_2]$ to predict trajectories in $[t_2, t_3]$, while in testing, we condition from $[t_1, t_3]$ to predict longer trajectories in $[t_3, t_4]$ to test model's extrapolation ability. We generate irregular-sampled trajectories for all datasets and set the number of training samples to be 20,000 and testing samples to be 5,000 respectively. The maximum trajectory prediction length is 60.

D.3 Baselines

We compare TANGO against three baseline types: 1) pure data-driven approaches including LG-ODE (Huang et al., 2020) and LatentODE (Rubanova et al., 2019), where the first one is a multi-agent approach considering the interaction graph, and the second one is a single-agent approach that predicts each trajectory independently; 2) energy-preserving HODEN (Greydanus et al., 2019); and 3) time-reversal TRS-ODEN (Huh et al., 2020).

The latter two are single-agent approaches and require initial states as given input. To handle missing initial states in our dataset, we approximate the initial states for the two methods via linear spline interpolation (Endre Süli, 2003). In addition, we substitute the ODE network in TRS-ODEN with a GNN (Kipf et al., 2018) as TRS-ODEN$_{GNN}$, which serves as a new multi-agent approach, for fair comparison. HODEN cannot be easily extended to the multi-agent setting as replacing the ODE function with a GNN can violate the energy conservation property of the original HODEN.

Note that we compared with baselines that are continuous, as discrete methods such as RNN (Sepp & Jürgen, 1997), NRI (Kipf et al., 2018) are not able to handle data irregularity (Huang et al., 2020).

Finally, we conduct two ablation by changing the implementation of $\mathcal{L}_{reverse}$: 1) TANGO$_{\mathcal{L}_{rev}=\text{gt-rev}}$, which computes the reversal loss as the L2 distance between ground truth trajectories to model backward trajectories as discussed in Remark 1 of Appendix B.3; 2) TANGO$_{\mathcal{L}_{rev}=\text{rev2}}$, which implements the reversal loss based on Eqn 2, similar as TRS-ODEN but calculate over latent z.

For all the baseline methods, we adopt hyper-parameter search over hidden dimension and learning rate, and report the best results. Implementation details is elaborated in Appendix D.4.

D.4 Implementation Details

TANGO

Our implementation of TANGO follows GraphODE pipeline. We implement the initial state encoder using a 2-layer GNN with a hidden dimension of 64 across all datasets. We use ReLU for nonlinear activation. For the sequence self-attention module, we set the output dimension to 128. The encoder's output dimension is set to 16, and we add 64 additional dimensions initialized with all zeros to the latent states $z_i(t)$ to stabilize the training processes as in Huang et al. (2021). The GNN ODE function is implemented with a single-layer GNN from Kipf et al. (2018) with hidden dimension 128. To compute trajectories, we use the Runge-Kutta method from torchdiffeq python package s(Chen et al., 2021) as the ODE solver and a one-layer MLP as the decoder.

We implement our model in pytorch. Encoder, generative model, and the decoder parameters are jointly optimized with AdamW optimizer (Loshchilov & Hutter, 2019) using a learning rate of 0.0001 for spring datasets and 0.00001 for *Pendulum*. The batch size for all datasets is set to 512.

TANGO$_{\text{gt-rev}}$ and TANGO$_{\text{rev2}}$ share the same architecture and hyparameters as TANGO, with different implementations of the loss function. In TANGO$_{\text{gt-rev}}$, instead of comparing forward and reverse trajectories, we look at the L2 distance between the ground truth and reverse trajectories when computing the reversal loss.

For TANGO$_{\text{rev2}}$, we implement the reversal loss following Huh et al. (2020) with one difference: we do not apply the reverse operation to the momentum portion of the initial state to the ODE function. This is because the initial hidden state is an output of the encoder that mixes position and momentum information. Note that we also remove the additional dimensions to the latent state that TANGO has.

LatentODE

We implement the Latent ODE sequence to sequence model as specified in (Rubanova et al., 2019). We use a 4-layer ODE function in the recognition ODE, and a 2-layer ODE function in the generative ODE. The recognition and generative ODEs use Euler and Dopri5 as solvers (Chen et al., 2021), respectively. The number of units per layer is 1000 in the ODE functions and 50 in GRU update networks. The dimension of the recognition model is set to 100. The model is trained with a learning rate of 0.001 with an exponential decay rate of 0.999 across different experiments. Note that since latentODE is a single-agent model, we compute the trajectory of each object independently when applying it to multi-agent systems.

HODEN

Table 2: Results of varying observation ratios on MSE (10^{-2}).

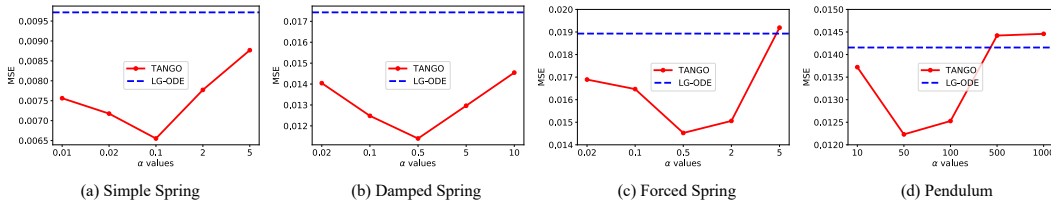| Dataset	Simple Spring		Forced Spring		Damped Spring		Pendulum	
Observation Ratios	0.8	0.4	0.8	0.4	0.8	0.4	0.8	0.4
LG-ODE	1.7054	1.6889	1.7554	2.0370	0.9305	1.0217	1.4314	1.7469
TANGO	**1.1176**	**1.1429**	**1.3611**	**1.5109**	**0.6920**	**0.6964**	**1.2309**	**1.2110**

(a) Simple Spring   (b) Damped Spring   (c) Forced Spring   (d) Pendulum

Figure 5: Varying α values across datasets.

To adapt HODEN, which requires full initial states of all objects, to systems with partial observations, we compute each object's initial state via linear spline interpolation if it is missing. Following the setup in Huh et al. (2020), we have two 2-layer linear networks with Tanh activation in between as ODE functions, in order to model both positions and momenta. Each network has a 1000-unit layer followed by a single-unit layer. The model is trained with a learning rate of 0.00001 using a cosine scheduler.

TRS-ODEN

Similar to HODEN, we compute each object's initial state via linear spline interpolation if it is missing. As in Huh et al. (2020), we use a 2-layer linear network with Tanh activation in between as the ODE functions, and the Leapfrog method for solving ODEs. The network has 1000 hidden units and is trained with a learning rate of 0.00001 using a cosine scheduler.

TRS-ODEN$_{\text{GNN}}$

For TRSODEN$_{\text{GNN}}$, we substitute the ODE function in TRS-ODEN with a GraphODE network. The GraphODE generative model is implemented with a single-layer GNN with hidden dimension 128. As in HODEN and TRS-ODEN, we compute each object's missing initial state via linear spline interpolation and the Leapfrog method for solving ODE. For all datasets, we use 0.5 as the coefficient for the reversal loss in Huh et al. (2020), and 0.0002 as the learning rate under cosine scheduling.

LGODE

Our implementation follows Huang et al. (2020) except we remove the Variational Autoencoder (VAE) from the initial state encoder. Instead of using the output from the encoder GNN as the mean and std of the VAE, we directly use it as the latent initial state. We use the same architecture as in TANGO and train the model using an AdamW optimizer with a learning rate of 0.0001 across all datasets.

D.5 Ablation and Sensitivity Analysis

Ablation on implementation of $\mathcal{L}_{reverse}$. From the last block of Table 1, we can clearly see that our implementation of reversal loss $\mathcal{L}_{reverse}$ achieves the best performance compared with gt-rev and TRS-ODEN's implementation (rev2). We also analytically show that our reversal loss implementation is expected to achieve a smaller error than the one in TRS-ODEN in Appendix B.3.

Evaluation across prediction lengths. We vary the maximum prediction lengths from 20 to 60 across models as shown in Figure 6. As the number of prediction step increases, TANGO keeps the best prediction performance, while other baselines have significant error accumulations. Notably for the chaotic pendulum dataset, which is highly nonlinear and has much more complex inter-agent interactions, single-agent baselines (HODEN, TRS-ODEN) perform poorly even on short prediction lengths. The performance gap between TANGO and baselines increases when making long-range predictions, showing the superior ability of TANGO.

Evaluation across observation ratios. For LG-ODE and TANGO, the encoder computes the initial states from observed trajectories. We randomly masked out 40% and 80% observations to study models' sensitivity towards data sparsity. As shown in Table 2, when changing the ratios from 80%
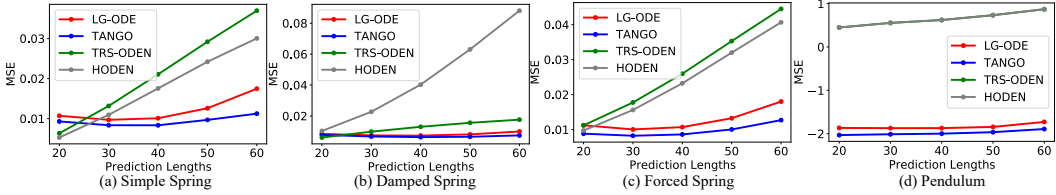
Figure 6: Varying prediction lengths across datasets (Pendulum MSE is in log values.).
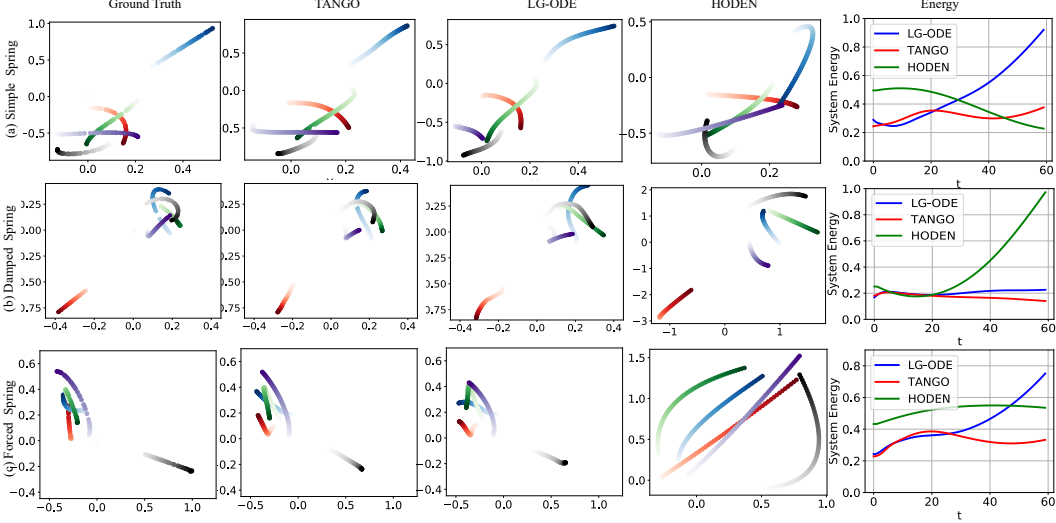

Figure 7: Trajectory and energy visualization (trajectory starts from light colors to dark colors.)
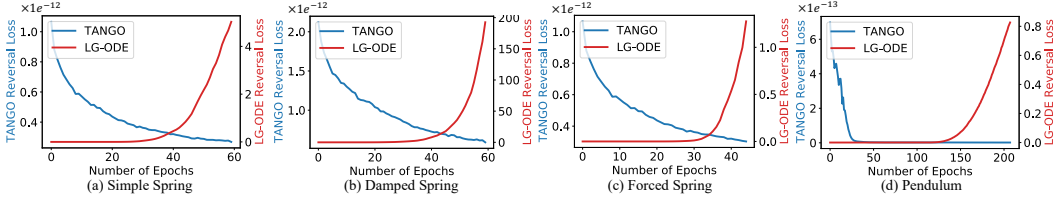

Figure 8: Time-Reversal symmetry loss visualization across datasets.

to 40%, we observe that TANGO has a smaller performance drop compared with LG-ODE, especially on the more complex Pendulum dataset (LG-ODE decreases 22.04% while TANGO decreases 1.62%). This indicates that TANGO is less sensitive toward data sparsity.

Evaluation across different α. We then vary the values of α defined in Eqn 7, which balances the prediction loss and the reversal loss. While prediction loss aims to match true trajectories, reversal loss ensures time-reversal property with better numerical accuracy Figure 5 demonstrates optimal α values being neither too high nor too low. When the weight α is too small, the model tends to neglect the physical bias, reducing test performance; Conversely, very high weights can emphasize reversibility at the cost of accuracy Nonetheless, across various α values, TANGO consistently surpasses LG-ODE, showcasing its flexibility in modeling diverse dynamical systems.

D.6 Visualizations

Trajectory Visualizations. Model predictions and ground truth are visualized in Figure 7. As HODEN is a single-agent baseline that individually forces every agent's energy to be constant over time which is not valid, the predicted trajectories is having the largest errors and systems' total energy is not conserved for all datasets. The purely data-driven LG-ODE exhibits unrealistic energy patterns, as seen in the energy spikes in *Simple Spring* and *Force Spring*. In contrast, TANGO, incorporating reversal loss, generates realistic energy trends, and consistently produces trajectories closest to the ground truth, showing its superior performance.

21

Reversal Loss Visualizations. To further illustrate the issue of energy explosion from LG-ODE that is purely data-driven, we visualize the reversal loss over training epochs from LG-ODE[8] and TANGO in Figure 8. As results suggest, LG-ODE is having increased reversal loss over training epochs, meaning it is violating the time-reversal symmetry sharply, as contrast to TANGO which has decrease reversal loss over epochs.

---

[8]There is no reversal loss backpropagation in LG-ODE, we just compute its value along training.