# Symplectic networks: Intrinsic structure-preserving networks for identifying Hamiltonian systems

Pengzhan Jin, Zhen Zhang

joint with Aiqing Zhu, George Em Karniadakis, Yifa Tang

Apr. 3rd, 2020

# Outline

# Introduction

Denote the $d$-by-$d$ identity matrix by $I_d$, let

$$J := \begin{pmatrix} 0 & I_d \\ -I_d & 0 \end{pmatrix}.$$

In consideration of the Hamiltonian system

$$\begin{cases} \dot{y} = J^{-1}\nabla H(y) \\ y(t_0) = y_0 \end{cases}, \tag{1}$$

where $y(t) \in \mathbb{R}^{2d}$, and $H$ is the Hamiltonian function, usually representing the energy of the system. Let $\phi_h(y)$ be the phase transformation of above system, i.e., $\phi_h(y_0) = y(t_0 + h)$. $\phi_h(y)$ in fact gives the next phase point after time step $h$ with respect to the phase point $y$.

# Introduction

**Definition**

A matrix $H \in \mathbb{R}^{2d \times 2d}$ is called symplectic if $H^T J H = J$.

**Definition**

A differentiable map $\Phi : U \to \mathbb{R}^{2d}$ (where $U \subset \mathbb{R}^{2d}$ is an open set) is called symplectic if the Jacobian matrix $\frac{\partial \Phi}{\partial x}$ is everywhere symplectic, i.e.,

$$\left(\frac{\partial \Phi}{\partial x}\right)^T J \left(\frac{\partial \Phi}{\partial x}\right) = J.$$

# Introduction

The connection between Hamiltonian system and symplectic map:

In 1899, Poincare pointed out that the phase transformation of the Hamiltonian system is a symplectic map, i.e.,

$$\left(\frac{\partial \phi_h}{\partial x}\right)^T J \left(\frac{\partial \phi_h}{\partial x}\right) = J.$$

"It is natural to look forward to those discrete systems which preserve as much as possible the intrinsic properties of the continuous system." (Kang Feng, 1984)

# Introduction

- Forward problem: solve the Hamiltonian system by **symplectic integrators**, such as the mid-point rule:

$$\bar{y} = y + hJ^{-1}\nabla H(\frac{\bar{y}+y}{2}).$$

(It is straightforward to verify that $\left(\frac{\partial \bar{y}}{\partial y}\right)^T J \left(\frac{\partial \bar{y}}{\partial y}\right) = J$)

- Inverse problem: how to discover the Hamiltonian system, given the data of phase points from a unknown Hamiltonian system?

  We know less about the inverse problem, while the forward problem is well developed.

# Introduction

Problem setup:

Given the dataset $\{(x_i, y_i)\}$ satisfying $\phi_h(x_i) = y_i$, to predict the phase flow $\phi_h(x), \phi_h^2(x), \phi_h^3(x) \cdots$, where $\phi_h^2(x)$ means $\phi_h \circ \phi_h(x)$.

Mostly, the dataset $\{(x_i, y_i)\}$ can be a series of phase points depending on time, i.e., $\{(x_i, x_{i+1})\}$ where $\phi_h(x_i) = x_{i+1}$.

# Introduction

**Hamiltonian neural networks (NIPS 2019)**
by Samuel Greydanus, Misko Dzamba, Jason Yosinski.

Using a neural network to approximate the Hamiltonian $H$ with loss

$$\|\frac{dy}{dt} - J^{-1}\nabla\widetilde{H}(y)\|,$$

and the time derivative is discretized by a numerical integrator (in fact should be a **symplectic integrator**, they did not mention this point, but it is indeed needed, proved by other later works). For example, with mid-point rule,

$$\|\frac{x_{i+1} - x_i}{h} - J^{-1}\nabla\widetilde{H}(\frac{x_i + x_{i+1}}{2})\|.$$

Finally computing the predicted flow with the trained network $\widetilde{H}$.

# Introduction

Another strategy: directly learn the phase transformation $\phi_{\boldsymbol{h}}$.

To this end, we expect to construct the neural networks preserving the symplecticity. Our architecture design philosophy is based on the fact that the composition of symplectic maps is again symplectic.

# Purpose

To construct the symplectic networks, we have to search for satisfactory unconstrained parametrization of symplectic matrices. We first show some existing modern factorizations.

Denote the collection of symplectic matrices by

$$SP = \{H \in \mathbb{R}^{2d \times 2d} | H^T J H = J\}.$$

# The modern factorizations

**Theorem (QR-like factorization)**

*The set of 2d-by-2d symplectic matrices is*

$$SP = \left\{ Q \begin{pmatrix} R & RS \\ 0 & R^{-T} \end{pmatrix} \middle| \begin{array}{l} R \in \mathbb{R}^{d\times d} \text{ upper triangular} \\ Q \text{ symplectic orthogonal},\ S \text{ symmetric} \end{array} \right\}.$$

**Theorem (Polar factorization)**

*The set of 2d-by-2d symplectic matrices is*

$$SP = \left\{ QP \middle| \begin{array}{l} P = P^T \text{ symplectic positive definite} \\ Q \text{ symplectic orthogonal} \end{array} \right\}.$$

**Theorem (SVD-like factorization)**

*The set of 2d-by-2d symplectic matrices is*

$$SP = \left\{ U \begin{pmatrix} \Omega & 0 \\ 0 & \Omega^{-1} \end{pmatrix} V^T \middle| \begin{array}{l} U, V \text{ symplectic orthogonal} \\ \Omega = diag(\omega_1, \cdots, \omega_d) \\ \omega_1 \geq \cdots \geq \omega_d \geq 1 \end{array} \right\}.$$

# The modern factorizations

**Transvections factorization.** The symplectic group is generated by so-called symplectic transvections.

## Definition

For $0 \neq u \in \mathbb{R}^{2d}$ and $0 \neq \beta \in \mathbb{R}$, the matrix

$$G = I + \beta u u^T J \in \mathbb{R}^{2d \times 2d}$$

is symplectic, and $G$ is called a symplectic transvection.

## Theorem

*The set of $2d$-by-$2d$ symplectic matrices is*

$$SP = \left\{ G_1 G_2 \cdots G_m \middle| \begin{array}{l} G_i \text{ a symplectic transvection} \\ m \leq 4d \end{array} \right\}.$$

# The modern factorizations

- Most of the factorizations require cells like symplectic-orthogonal matrices, which are not elementary enough hence hard to be freely parameterized.
- Transvections factorization needs $\mathcal{O}(d)$ generators.
- We need a more elementary unconstrained factorization, since the deep learning techniques focus on the unconstrained optimization.

# Unit triangular factorization

**Unit triangular symplectic matrices**

$$\begin{pmatrix} I & S \\ 0 & I \end{pmatrix}, \quad \begin{pmatrix} I & 0 \\ S & I \end{pmatrix}, \quad S^T = S$$

*We can rewrite the symmetric $S$ as $(W^T + W)$ in practice, then the $W$ is unconstrained.* Denote

$$L_n = \left\{ \begin{pmatrix} I & 0/S_n \\ S_n/0 & I \end{pmatrix} \cdots \begin{pmatrix} I & S_3 \\ 0 & I \end{pmatrix} \begin{pmatrix} I & 0 \\ S_2 & I \end{pmatrix} \begin{pmatrix} I & S_1 \\ 0 & I \end{pmatrix} \middle| \right.$$
$$\left. S_i \in \mathbb{R}^{d\times d}, S_i^T = S_i, i = 1, 2, \cdots, n \right\},$$

where the unit upper triangular symplectic matrices and the unit lower triangular symplectic matrices appear alternately. And it is clear that $L_m \subset L_n \subset SP$ for all integers $1 \leq m \leq n$.

# Unit triangular factorization

**Theorem (Unit Triangular Factorization)**

$SP = L_9$.

Thus any symplectic matrix can be factored into no more than 9 unit triangular symplectic matrices. **Only $\mathcal{O}(1)$ generators.**

**Unit triangular factorization of the matrix symplectic group** (2019, by Pengzhan Jin, Yifa Tang, Aiqing Zhu.)

# SympNets

- **Linear Modules**

$$\mathcal{L}_n(x) = \begin{pmatrix} I & 0/S_n \\ S_n/0 & I \end{pmatrix} \cdots \begin{pmatrix} I & 0 \\ S_2 & I \end{pmatrix} \begin{pmatrix} I & S_1 \\ 0 & I \end{pmatrix} x + b, \quad x, b \in \mathbb{R}^{2d}, S_i^T = S_i.$$

Here $S_i = W_i^T + W_i \in \mathbb{R}^{d\times d}$, and $W_i$ are the network parameters to be optimized.

- **Activation Modules**

$$\mathcal{N}_{up}\begin{pmatrix} p \\ q \end{pmatrix} = \begin{bmatrix} I & \tilde{\sigma}_a \\ 0 & I \end{bmatrix} \begin{pmatrix} p \\ q \end{pmatrix} := \begin{pmatrix} p + diag(a)\sigma(q) \\ q \end{pmatrix}, \quad \mathcal{N}_{low}\begin{pmatrix} p \\ q \end{pmatrix} = \begin{bmatrix} I & 0 \\ \tilde{\sigma}_a & I \end{bmatrix} \begin{pmatrix} p \\ q \end{pmatrix}.$$

Here $a \in \mathbb{R}^d$ is the network parameter to be optimized.

- **Gradient Modules**

$$\mathcal{G}_{up}\begin{pmatrix} p \\ q \end{pmatrix} = \begin{bmatrix} I & \hat{\sigma}_{K,a,b} \\ 0 & I \end{bmatrix} \begin{pmatrix} p \\ q \end{pmatrix} := \begin{pmatrix} p + K^T diag(a)\sigma(Kq + b) \\ q \end{pmatrix},$$

$$\mathcal{G}_{low}\begin{pmatrix} p \\ q \end{pmatrix} = \begin{bmatrix} I & 0 \\ \hat{\sigma}_{K,a,b} & I \end{bmatrix} \begin{pmatrix} p \\ q \end{pmatrix}.$$

Here $K \in \mathbb{R}^{m\times d}, a, b \in \mathbb{R}^m$ are the network parameters to be optimized.

# SympNets

**Remark**

*The term "gradient module": we proved that $\hat{\sigma}_{K,a,b}$ can approximate any $\nabla f$ for $f : \mathbb{R}^d \to \mathbb{R}$.*

$$\mathcal{M}_L = \{\psi | \psi \text{ is a linear module}\}.$$
$$\mathcal{M}_A = \{\psi | \psi \text{ is an activation module}\}.$$
$$\mathcal{M}_G = \{\psi | \psi \text{ is a gradient module}\}.$$

**Definition**

Consider $\{v_i\}_1^{k+1} \subset \mathcal{M}_L$, $\{w_i\}_1^k \subset \mathcal{M}_A$. Let

$$\psi = v_{k+1} \circ w_k \circ v_k \circ \cdots \circ w_1 \circ v_1.$$

$\psi$ is called the **LA-SympNet**. We define the collection of all the LA-SympNets as

$$\Psi_{LA} = \{\psi | \psi \text{ is a LA-SympNet}\}.$$

# SympNets

**Definition**

Consider $\{u_i\}_1^k \subset \mathcal{M}_G$. Let

$$\psi = u_k \circ u_{k-1} \circ \cdots \circ u_1.$$

$\psi$ is called the **G-SympNet**. We define the collection of all the G-SympNets as

$$\Psi_G = \{\psi | \psi \textit{ is a G-SympNet}\}.$$

**Definition**

Consider $\{v_i\}_1^k \subset \mathcal{M}_L \cup \mathcal{M}_A \cup \mathcal{M}_G$, where $\mathcal{M}_L$, $\mathcal{M}_A$ and $\mathcal{M}_G$ are the set of linear, activation and gradient modules respectively. Let

$$\psi = v_k \circ v_{k-1} \circ \cdots \circ v_1.$$

$\psi$ is called the **symplectic network (SympNet)**. Futhermore, we define the collection of all the SympNets as

$$\Psi = \{\psi | \psi \textit{ is a symplectic network}\}.$$

# SympNets

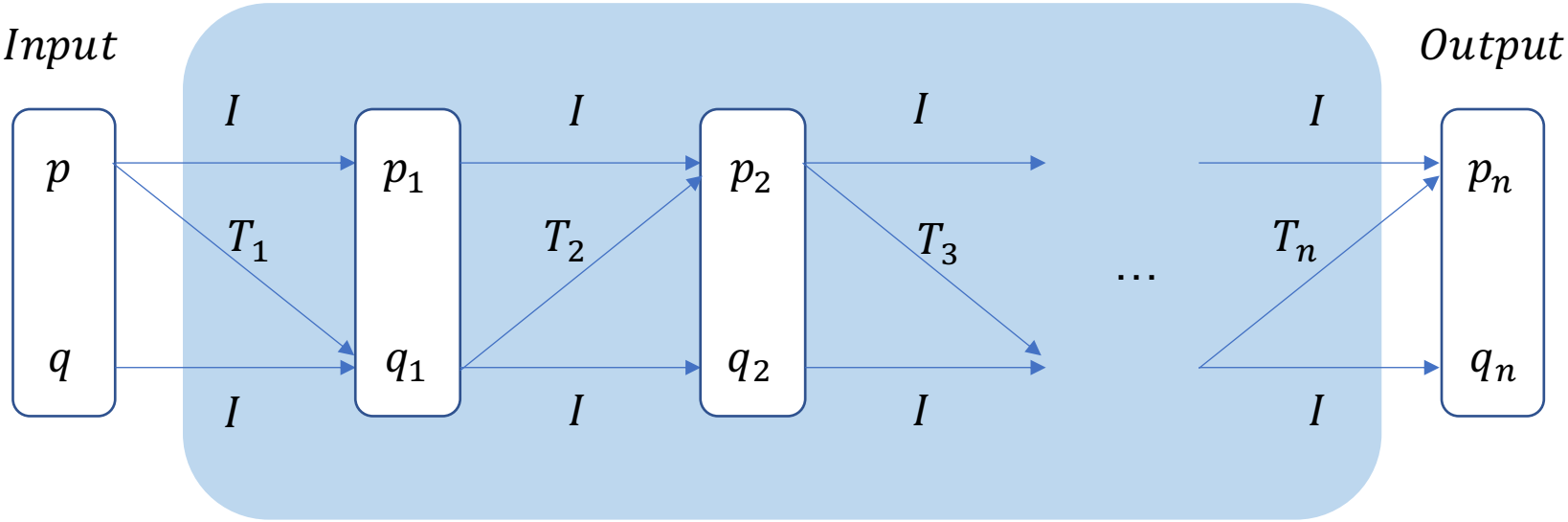



Figure 1: **Architecture of SympNet.** The SympNet can be seen as the neural network with a specific connection pattern, which gaurantees symplecticity. Here $T_i$ can be chosen as $S$, $\tilde{\sigma}$ or $\hat{\sigma}$, depending on which type of module it belongs to.

# Algebraic structure of SympNets

We show the main theorems.

**Theorem (Group structure)**

*The collection of all the symplectic networks $\Psi$ is a group.*

**Theorem (Group structure)**

*The collection of all the LA(G)-SympNets $\Psi_{LA}(\Psi_{GS})$ is a group.*

**Proof.**

$$\begin{pmatrix} I & S \\ 0 & I \end{pmatrix}^{-1} = \begin{pmatrix} I & -S \\ 0 & I \end{pmatrix}$$

$$\begin{bmatrix} I & \tilde{\sigma}_a \\ 0 & I \end{bmatrix}^{-1} = \begin{bmatrix} I & -\tilde{\sigma}_a \\ 0 & I \end{bmatrix}, \quad \begin{bmatrix} I & \hat{\sigma}_{K,a,b} \\ 0 & I \end{bmatrix}^{-1} = \begin{bmatrix} I & -\hat{\sigma}_{K,a,b} \\ 0 & I \end{bmatrix}.$$

□

# Algebraic structure of SympNets

It is noteworthy that we can easily obtain the inverse network of a SympNet! For example:

$$\phi(x) = \begin{pmatrix} I & 0 \\ S_3 & I \end{pmatrix} \begin{bmatrix} I & \tilde{\sigma}_a \\ 0 & I \end{bmatrix} \begin{pmatrix} I & 0 \\ S_2 & I \end{pmatrix} \begin{pmatrix} I & S_1 \\ 0 & I \end{pmatrix} x,$$

then

$$\phi^{-1}(x) = \begin{pmatrix} I & -S_1 \\ 0 & I \end{pmatrix} \begin{pmatrix} I & 0 \\ -S_2 & I \end{pmatrix} \begin{bmatrix} I & -\tilde{\sigma}_a \\ 0 & I \end{bmatrix} \begin{pmatrix} I & 0 \\ -S_3 & I \end{pmatrix} x.$$

# Approximation theorems

$$\mathcal{SP}^r(U) = \left\{ \Phi \in C^r(U; \mathbb{R}^{2d}) \middle| \left(\frac{\partial \Phi}{\partial x}\right)^T J \left(\frac{\partial \Phi}{\partial x}\right) = J \right\}.$$

**Definition**

Let $r \in \{0\} \cup \mathbb{N}$ be given. $\sigma$ is $r$-finite if $\sigma \in C^r(\mathbb{R})$ and $0 < \int |D^r(\sigma)| d\lambda < \infty$, where $\lambda$ is the Lebesgue measure on $\mathbb{R}$.

**Theorem (Approximation theorem for G-SympNet)**

*If the activation function $\sigma$ is $r$-finite, then the set of G-SympNet $\Psi_G$ is $r$-uniformly dense on compacta in $\mathcal{SP}^r(U)$.*

**Theorem (Approximation theorem for LA-SympNet)**

*If the activation function $\sigma$ is $r$-finite, then the set of LA-SympNet $\Psi_{LA}$ is $r$-uniformly dense on compacta in $\mathcal{SP}^r(U)$.*

# Approximation theorems

In practice we use the sigmoid as the activation function. Sigmoid is $r - \textit{finite}$ for arbitrary $r > 0$.

# Numerical results

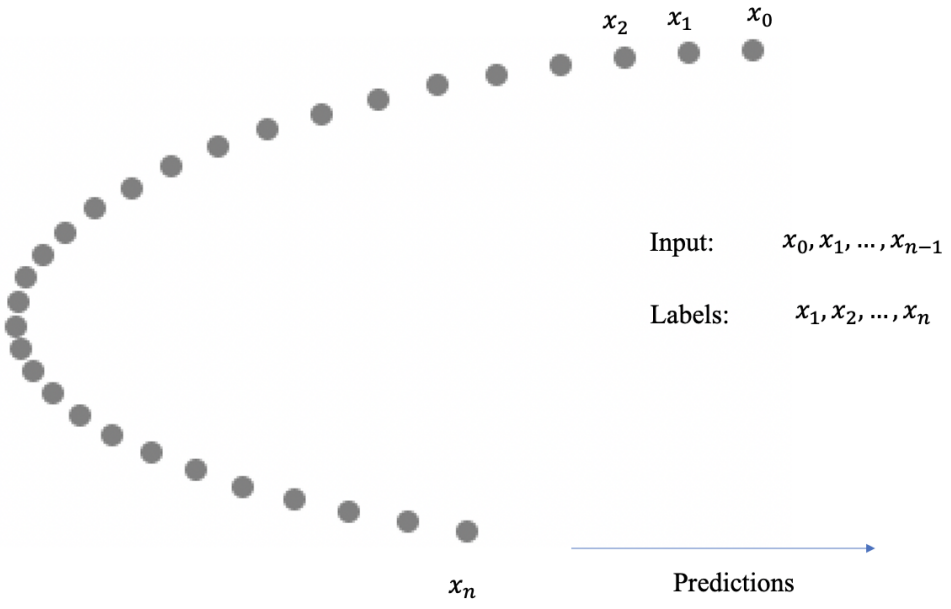



Figure 2: Example of Training Dataset

- Only one trajectory is simulated, want to make one-shot prediction.

# Comparison with HNN

Advantages of SympNet:

- Relatively smaller parameter size (LA-SympNet).
- Can handle sparsely-sampled data.
- Can scale better to higher dimensions.
- Will converge to the ground truth when the network size increases, while HNN converging to the modified system due to the discretization.
- Don't need to manually choose numerical integrator, can handle nonseparable Hamiltonian much more easily.
- Faster in training and predicting.

Advantages of HNN:

- Can handle irregularly sampled data.

# Pendulum

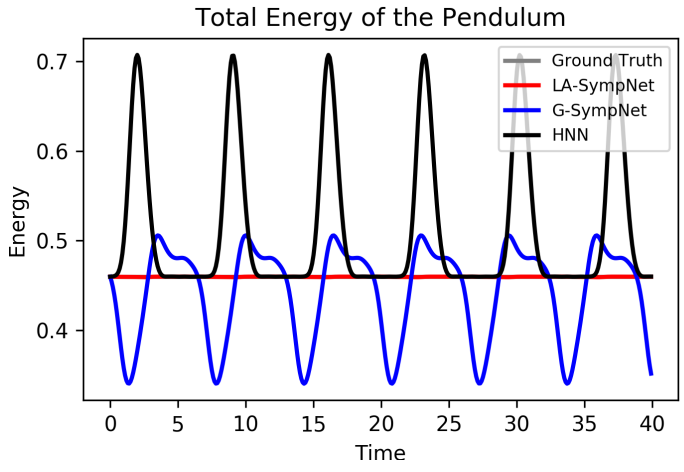
Mathematical pendulum (mass $m = 1$, massless rod of length $l = 1$, gravitational acceleration $g = 1$) which is a system with one degree of freedom having the Hamiltonian

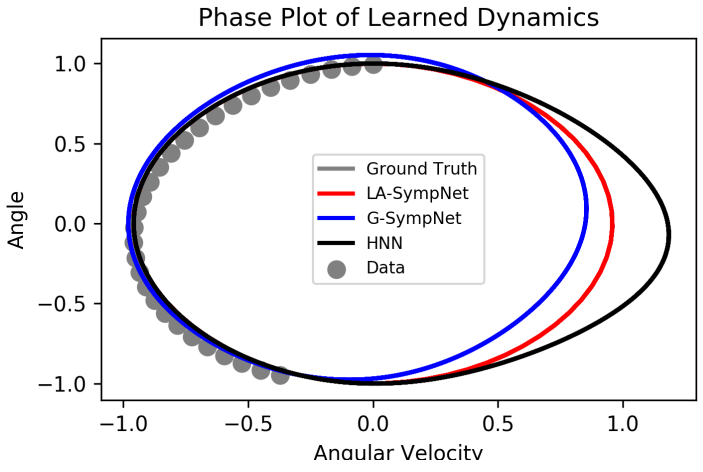
$$H(p, q) = \frac{1}{2}p^2 - \cos(q).$$

We obtain the flow starting from $(0, 1.0)$ with 30 points and time step 0.1 as the training data, i.e., $\mathcal{T} = \{(x_{i-1}, x_i)\}_1^n$ where $n = 30$, $x_0 = (0, 1.0)$.

- LA-SympNet can learn the dynamics of this pendulum perfectly with only 14 parameters.

# Pendulum

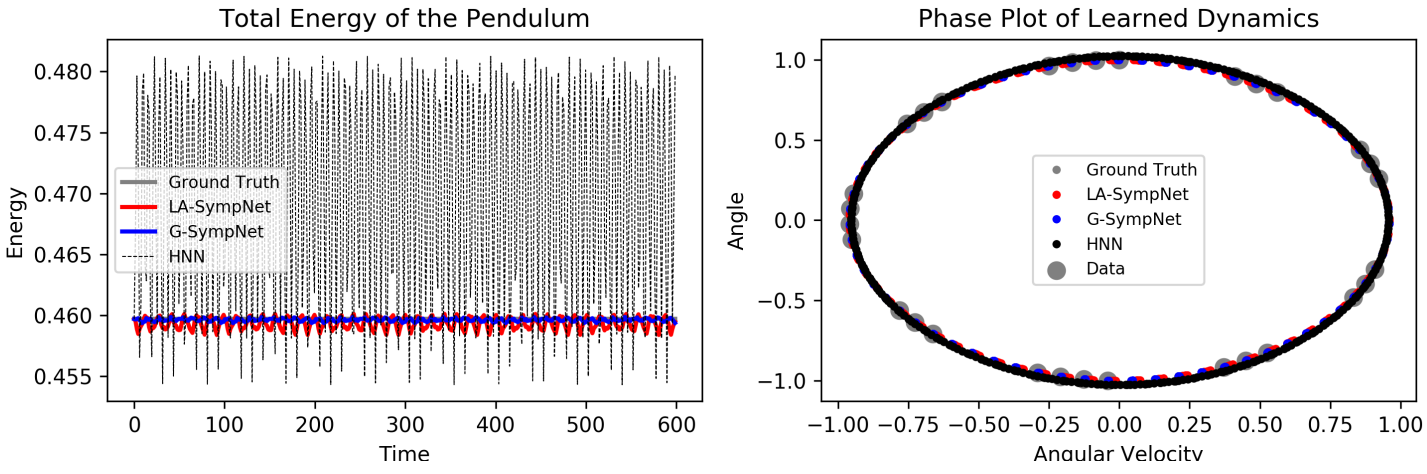



- Energy is conserved perfectly by LA-SympNet.
- Even if we increase the parameter size of HNN, it is still not comparable with LA-SympNet.

# Pendulum V2

Now we sample data sparsely: choose time step $dt = 1.5$.




HNN fails since its loss function involves discretization in time.

# Two Body Problem

We try to model the 2-body problem with Hamiltonian

$$H = \frac{|p_{CM}^2|}{m_1 + m_2} + \frac{|p_1|^2 + |p_2|^2}{2\mu} + g\frac{m_1 m_2}{|q_1 - q_2|^2}$$

For simplicity, assume $m_1 = m_2 = g = 1$. We simulate a trajectory, and use one period as training dataset. We initialize the trajectory with center of mass zero, total momentum 0, and radius $|q_1 - q_2|$ in the range $[0.5, 1.5]$.

# Two Body Problem

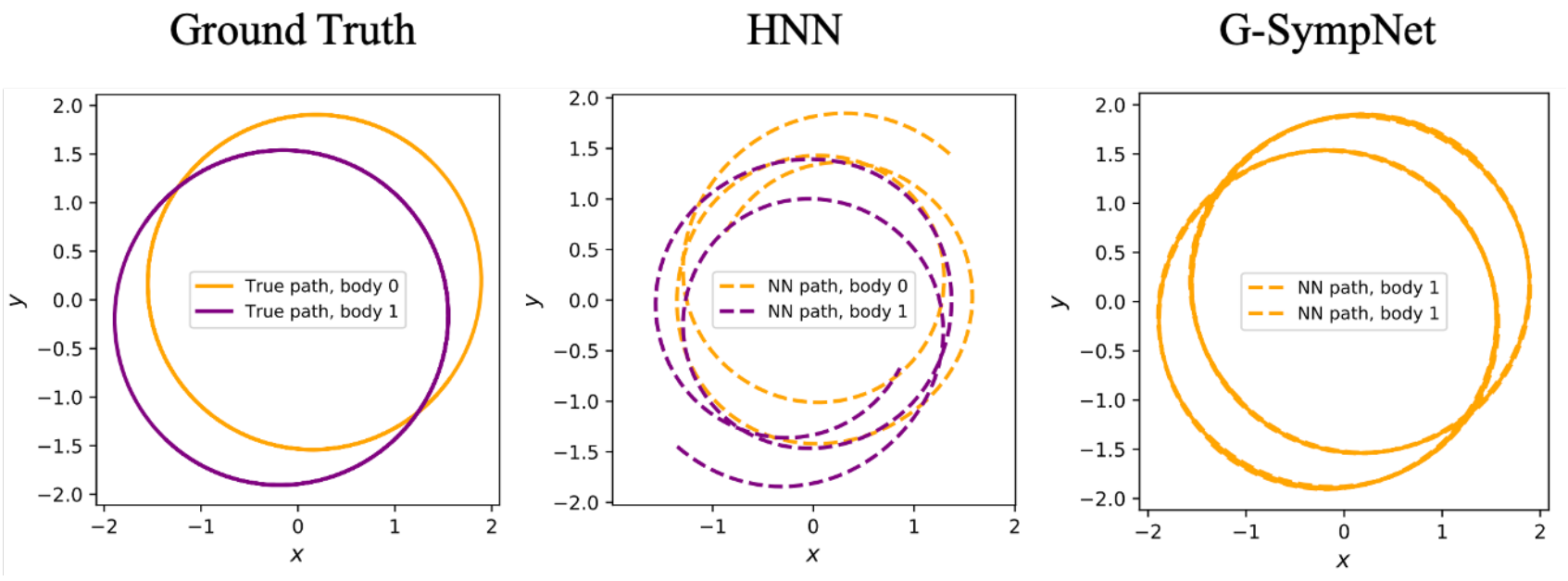



Figure 3: Result for 2-body problem

Still tuning hyperparameters for LA-SympNet.

# Possible Future Work

- Handle irregularly sampled data?
- More difficult dataset, PDEs.
- Applications to normalizing flows.
- Symplectic RNN.
- ...

# Thanks!