Exact conservation laws for neural network integrators of dynamical systems

Eike Hermann Müller

Department of Mathematical Sciences

UKLFT meeting Plymouth, Mon 18th Mar 2024

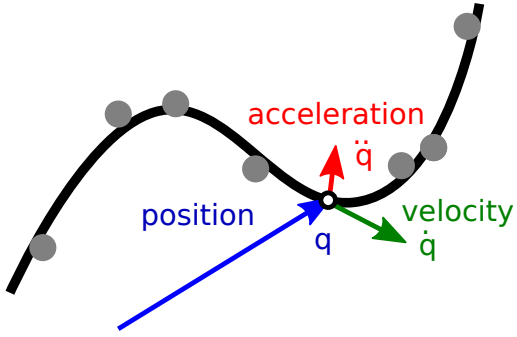
[JCP 488, p.112234 (2023), arxiv:2209.11661]

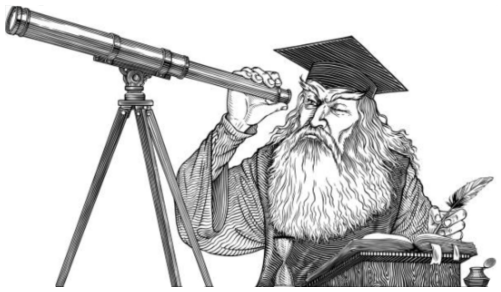
Dynamical systems

Challenge: learning dynamics from data

Predict acceleration from
position and velocity

$$\ddot{q} = \Phi(q, \dot{q})$$

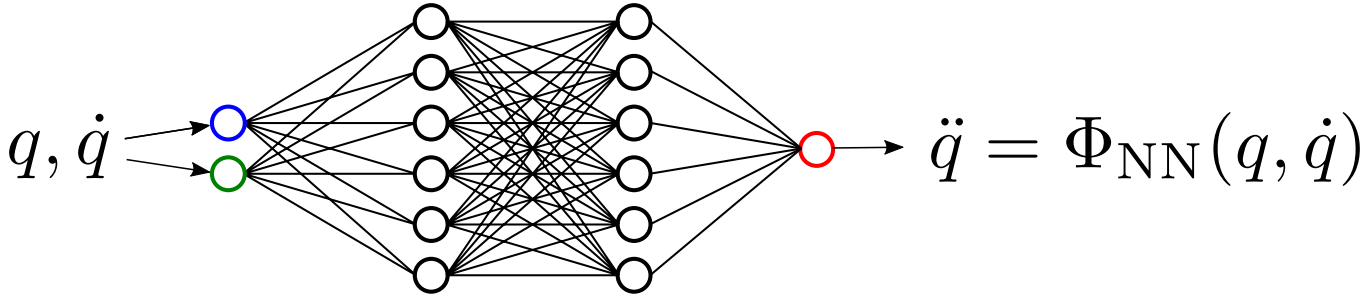




(noisy) training data

$$\left\{\ddot{q}^{(j)}, q^{(j)}, \dot{q}^{(j)}\right\}_{j=1}^{N}$$

$\Rightarrow$ learn function $\Phi \Rightarrow$ neural network Φ_{NN}



Conservation laws

Physical constraints = **conservation laws**
e.g. for:

- energy
- linear momentum
- angular momentum L

- ☹ Neural network has to **learn** constraints from data
- ☹ $\Rightarrow$ Conservation only **approximate**

Soft constraints in loss function?

$$\text{Loss}(\{\ddot{q}^{(j)}, q^{(j)}, \dot{q}^{(j)}\}) = \cdots + \frac{1}{N}\sum_{j=1}^{N} \left\| L(q^{(j)}, \dot{q}^{(j)}) - L(q_0, \dot{q}_0) \right\|$$

Can we build the conservation laws into the neural network?

Conservation laws: hard constraints

Can we build the conservation laws into the neural network?

Need two ingredients:

1. Lagrangian neural networks [Cranmer et al. (2020)]
2. Noether's Theorem [Noether (1918)]

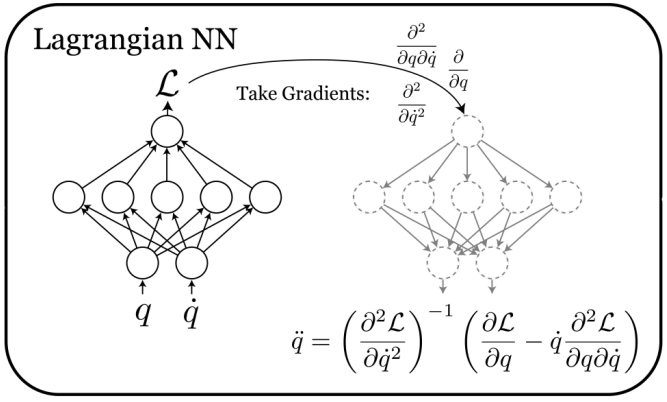
Lagrangian neural networks

Represent Lagrangian $\mathcal{L}$ by neural network [Cranmer (2020)]

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}} - \frac{\partial \mathcal{L}}{\partial q} = 0.$$

acceleration

$$\ddot{q} = \left(\frac{\partial^2 \mathcal{L}}{\partial \dot{q}^2}\right)^{-1}\left(\frac{\partial \mathcal{L}}{\partial q} - \dot{q}\frac{\partial^2 \mathcal{L}}{\partial q \partial \dot{q}}\right)$$



source: Cranmer et al. arxiv:2003.04630

NB: tensorflow will take care of derivatives

Noether's Theorem

Theorem [Noether (1918)]

Continuous symmetry of the Lagrangian $\Rightarrow$ conservation law

Symmetry transformation h^s:

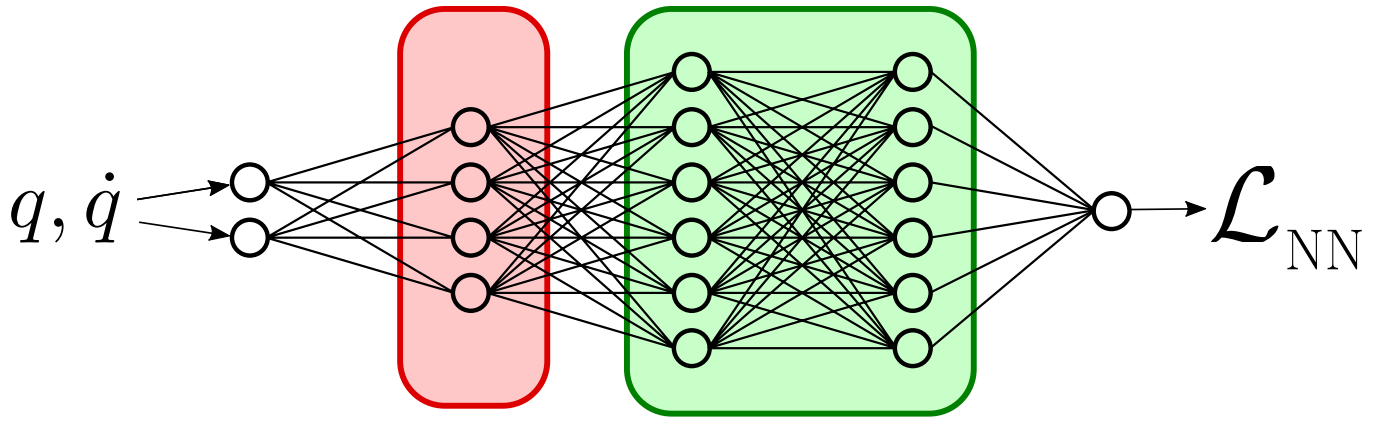
$$\mathcal{L}(h^s(q), h^s_{*,q}(\dot{q})) = \mathcal{L}(q, \dot{q})$$

Conserved quantity

$$I = \frac{\partial \mathcal{L}}{\partial \dot{q}} \frac{dh^s(q)}{ds}\Big|_{s=0} \qquad \frac{dI}{dt} = 0$$

Invariant NN Lagrangian

$\Rightarrow$ make neural network Lagrangian $\mathcal{L}_{\text{NN}}$ invariant under symmetry transformation(s)



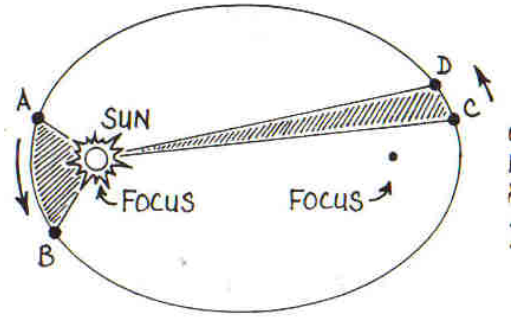
Output of **symmetry enforcing layer**: invariant scalars

$$S_j(h^s(q), h^s_{*,q}(\dot{q})) = S_j(q, \dot{q}) \quad j = 1, 2, \ldots, M \qquad \text{for all } q, \dot{q} \in TM,$$

Symmetry transformations

Examples

- Invariance under translations $\Rightarrow$ momentum conservation
- Rotational invariance $\Rightarrow$ angular momentum conservation



www.neam.co.uk/gcseAstronomy/kepler.html

Rotation group $SO(d)$

$$h^s(q) = \exp[s\Gamma]q \qquad \Gamma \in \mathfrak{so}(d)$$

$\Rightarrow$ (approximate) angular momentum:

$$L_{\text{NN}} = \frac{\partial \mathcal{L}}{\partial \dot{q}}\Gamma q \qquad \left(= \boldsymbol{x}\times\frac{\partial \mathcal{L}_{\text{NN}}}{\partial \dot{\boldsymbol{x}}} \approx \boldsymbol{x}\times\dot{\boldsymbol{x}} = L_{\text{true}} \qquad \text{in } d = 3 \text{ dimensions}\right)$$

Invariants (d=3)

$$S_1 = \boldsymbol{x}\cdot\boldsymbol{x}, \quad S_2 = \boldsymbol{x}\cdot\dot{\boldsymbol{x}}, \quad S_3 = \dot{\boldsymbol{x}}\cdot\dot{\boldsymbol{x}}$$

Setup

Network architecture: dense layers

- $2\times$ 128 nodes
- activation function = softplus $f(x) = \log\left(1 + e^x\right)$
- no attempts to optimise architecture or other hyperparameters
  (I only have a single NVIDIA Geforce GTX 1660 Super GPU...)

Tensorflow implementation

https://github.com/eikehmueller/mlconservation_code

All computations in single precision arithmetic

$\Rightarrow$ **relative errors** $\Delta Q/Q \sim 10^{-7}$

Training

Training data generation

- Fix initial condition $q(0), \dot{q}(0)$
- Integrate true trajectory with RK4, $\Delta t = 10^{-2}$
  (integration error $\sim 10^{-8}$)
- Sample $q(t^{(j)}), \dot{q}(t^{(j)}), \ddot{q}(t^{(j)})$ at times $t^{(j)}$
- Add i.i.d. normal random noise $\xi \sim \mathcal{N}(0, \sigma)$, $\sigma = 10^{-3}$
  $\Rightarrow q^{(j)}, \dot{q}^{(j)}, \ddot{q}^{(j)}$
- 2500 epochs, 100 steps/epoch, batchsize = 128

MSE loss (= acceleration mismatch)

$$\text{Loss}\left(\{q^{(j)}, \dot{q}^{(j)}, \ddot{q}^{(j)}\}_{j=1}^{N}\right) = \frac{1}{N}\sum_{j=1}^{N}\left(\Phi_{\text{NN}}(q^{(j)}, \dot{q}^{(j)}) - \ddot{q}^{(j)}\right)^2$$

$$\gtrsim \sigma^2 = 10^{-6}$$

(recall: NN predicts $\ddot{q} = \Phi_{\text{NN}}(q, \dot{q})$)

Model problem I

Motion of massive relativistic particle in **Schwarzschild metric**

$$\mathcal{L}_{\text{true}} = -\left(1 - \frac{r_s}{r}\right)\dot{\tau}^2 + \left(1 - \frac{r_s}{r}\right)^{-1}\dot{r}^2 + r^2\left(\dot{\theta}^2 + \sin^2(\theta)\dot{\varphi}^2\right)$$

$$\mathcal{L}_{\text{NN}} = \underbrace{\mathcal{L}_{\text{NN}}(\dot{\tau}, \boldsymbol{x}, \dot{\boldsymbol{x}})}_{\text{unconstrained}} \mapsto \underbrace{\mathcal{L}_{\text{NN}}(\dot{\tau}, \boldsymbol{x}^2, \dot{\boldsymbol{x}}^2, \boldsymbol{x}\cdot\dot{\boldsymbol{x}})}_{\text{rotationally invariant}}$$

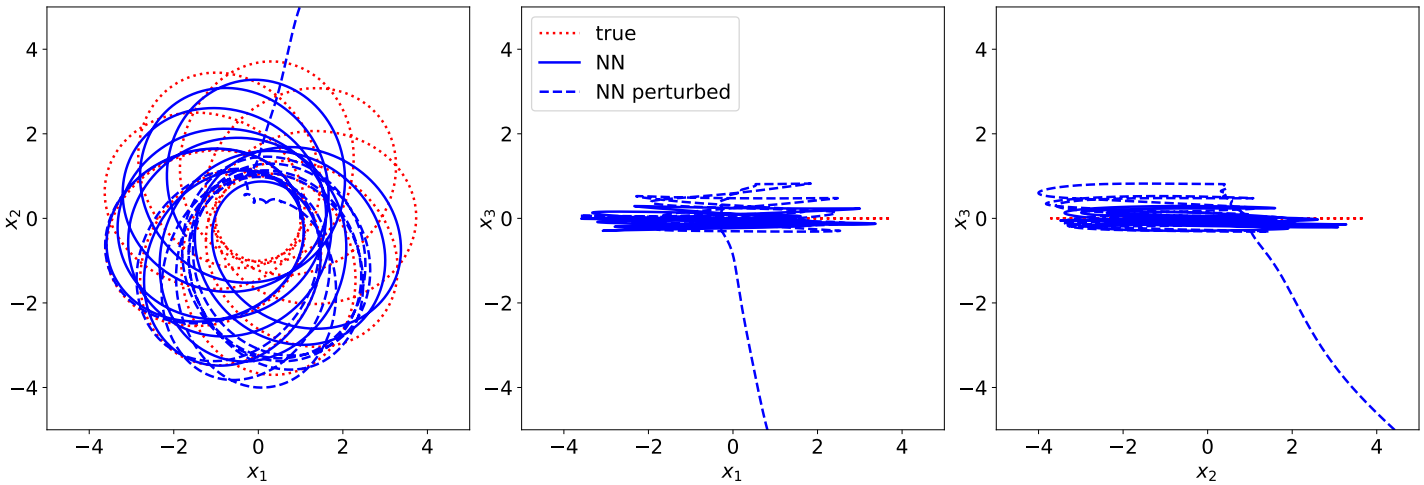
Rotational invariance $\Rightarrow$ conservation of angular momentum

$$\boldsymbol{L}_{\text{true}} = \boldsymbol{x}\times\dot{\boldsymbol{x}}, \quad \boldsymbol{L}_{\text{NN}} = \boldsymbol{x}\times\frac{\partial\mathcal{L}}{\partial\dot{\boldsymbol{x}}}$$

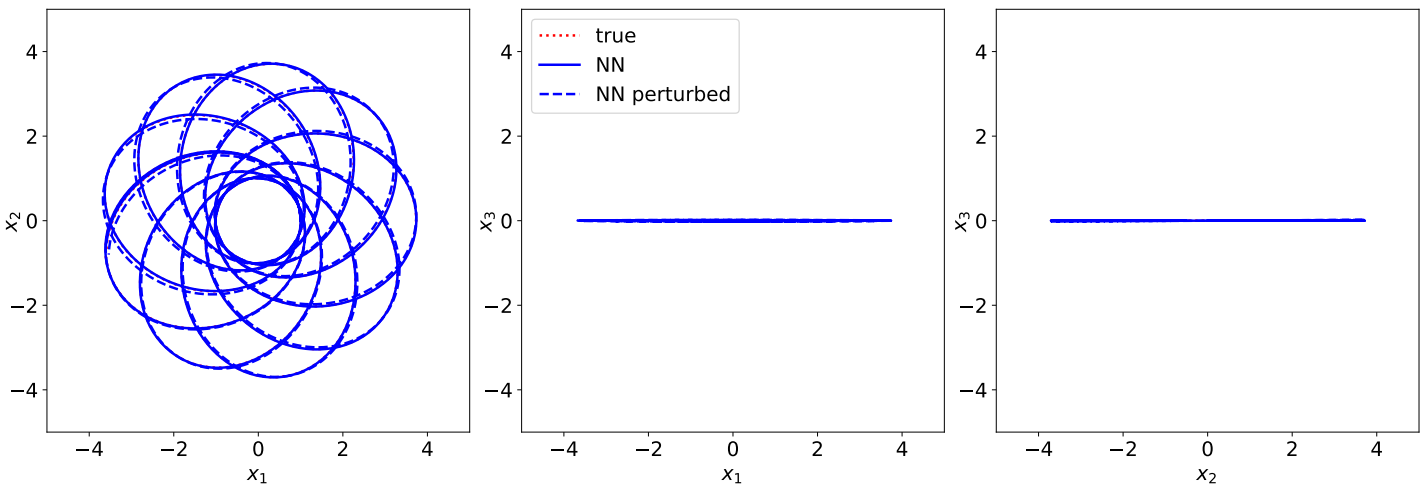
"Gargantua" from Interstellar (Paramount Pictures)

Results Ia - Trajectories

unconstrained NN

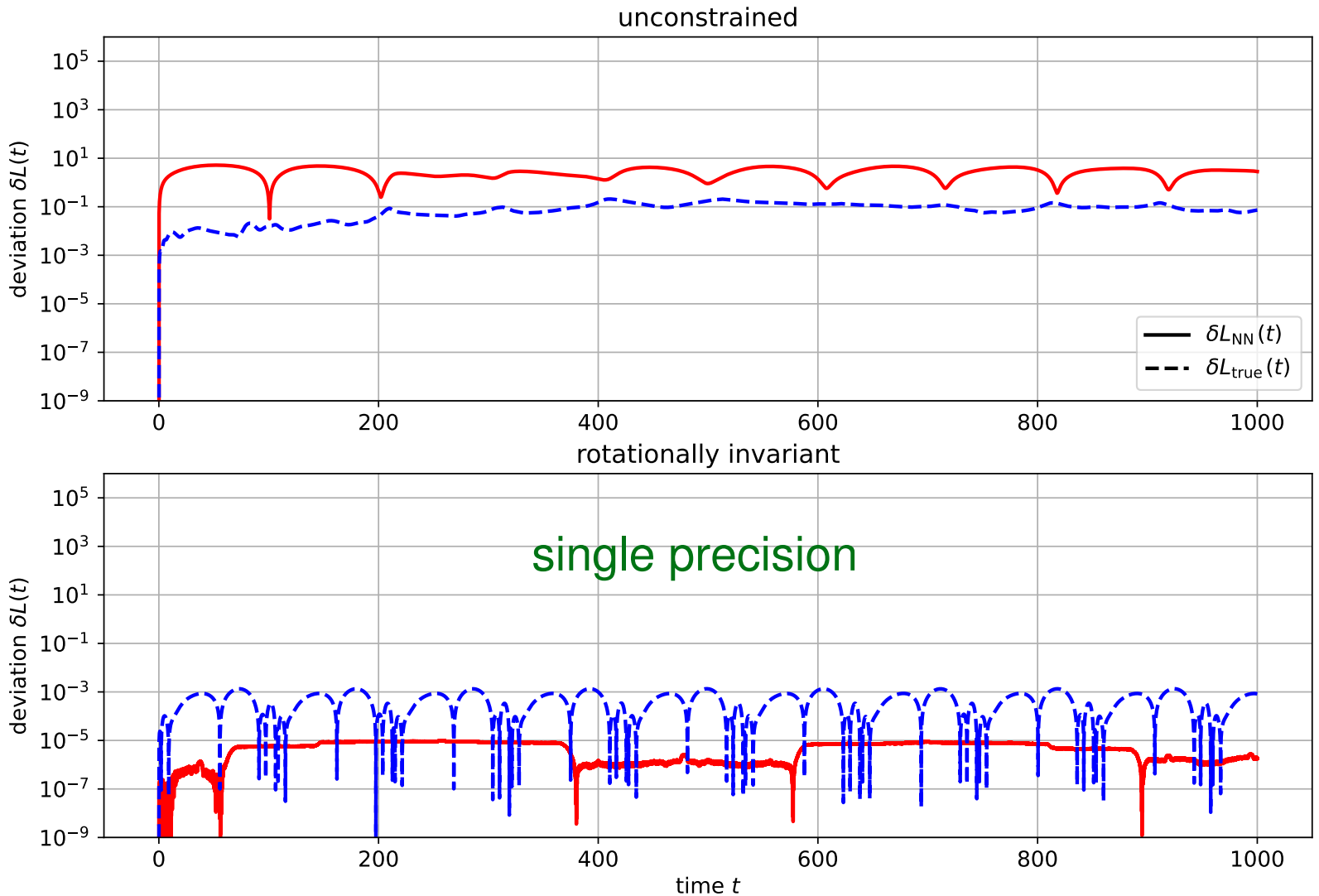


rotationally invariant NN



Results Ib - Conservation of angular momentum

$$\boldsymbol{L}_{\text{true}} = \boldsymbol{x} \times \dot{\boldsymbol{x}}, \quad \boldsymbol{L}_{\text{NN}} = \boldsymbol{x} \times \frac{\partial \mathcal{L}}{\partial \dot{\boldsymbol{x}}} \qquad \delta L(t) = \|\boldsymbol{L}(t) - \boldsymbol{L}(0)\|_2 / \|\boldsymbol{L}(0)\|_2$$



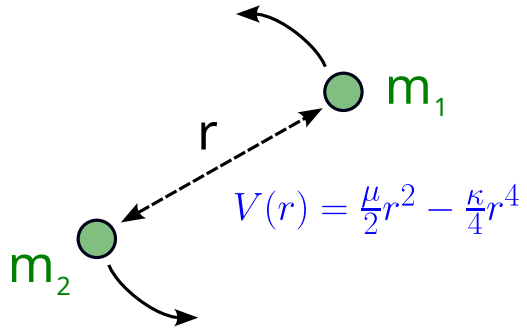
Model problem II: Two particles in D dimensions

Particle positions $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)} \in \mathbb{R}^D$

$$\mathcal{L}_{\text{true}}(q, \dot{q}) = \mathcal{L}_{\text{true}}(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \dot{\boldsymbol{x}}^{(1)}, \dot{\boldsymbol{x}}^{(2)})$$

$$= \frac{m_1}{2} \sum_{j=1}^{D} (\dot{x}_j^{(1)})^2 + \frac{m_2}{2} \sum_{j=1}^{D} (\dot{x}_j^{(2)})^2$$

$$- V(\|\boldsymbol{x}^{(1)} - \boldsymbol{x}^{(2)}\|)$$



linear momenta $$M^{(\alpha)} = \frac{\partial \mathcal{L}}{\partial \dot{x}_\alpha^{(1)}} + \frac{\partial \mathcal{L}}{\partial \dot{x}_\alpha^{(2)}}$$

angular momenta $$L^{(\rho,\sigma)} = \frac{\partial \mathcal{L}}{\partial \dot{x}_\rho^{(1)}} x_\sigma^{(1)} - \frac{\partial \mathcal{L}}{\partial \dot{x}_\sigma^{(1)}} x_\rho^{(1)} + \frac{\partial \mathcal{L}}{\partial \dot{x}_\rho^{(2)}} x_\sigma^{(2)} - \frac{\partial \mathcal{L}}{\partial \dot{x}_\sigma^{(2)}} x_\rho^{(2)}$$

Neural network Lagrangian (unconstrained)

$$\mathcal{L}_{\text{NN}} = \mathcal{L}_{\text{NN}}(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \dot{\boldsymbol{x}}^{(1)}, \dot{\boldsymbol{x}}^{(2)})$$

Invariants

Scalar invariants $S_1, S_2, \ldots$

Translational invariance

$$\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \dot{\boldsymbol{x}}^{(1)}, \dot{\boldsymbol{x}}^{(2)} \mapsto \boldsymbol{x}^{(1)} - \boldsymbol{x}^{(2)}, \dot{\boldsymbol{x}}^{(1)}, \dot{\boldsymbol{x}}^{(2)}$$

Rotational invariance

$$\boldsymbol{x}^{(1)} - \boldsymbol{x}^{(2)}, \dot{\boldsymbol{x}}^{(1)}, \dot{\boldsymbol{x}}^{(2)} \mapsto \{S_1, S_2, \ldots\} = \mathscr{R}\left(\boldsymbol{x}^{(1)} - \boldsymbol{x}^{(2)}, \dot{\boldsymbol{x}}^{(1)}, \dot{\boldsymbol{x}}^{(2)}\right)$$

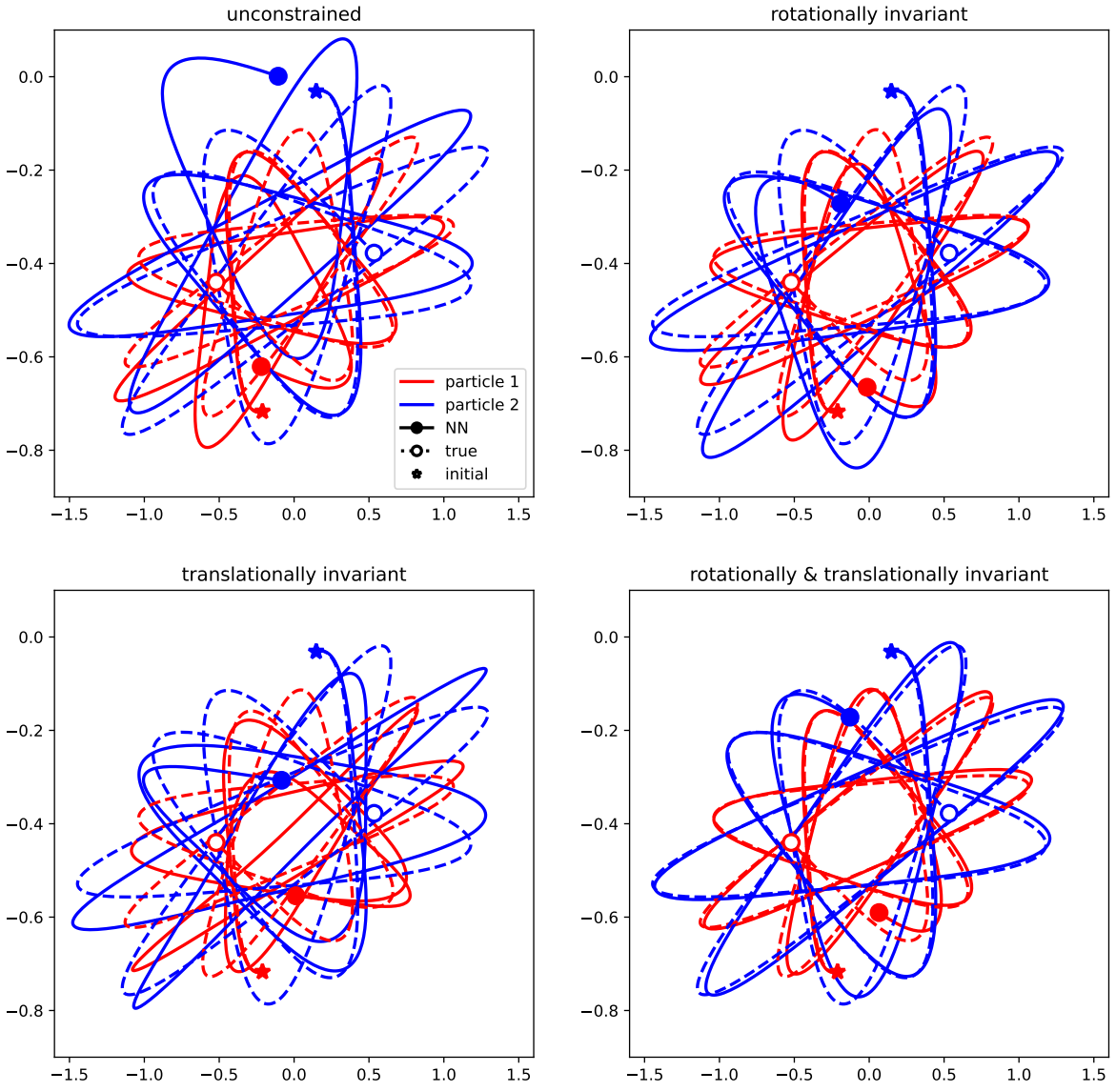
$$\mathscr{R}(a^{(1)}, a^{(2)}, \ldots, a^{(n)}) := \{\text{all possible contractions with } \eta, \varepsilon\}$$

$$= \left\{ \underbrace{\eta^{jk} a_j^{(r)} a_k^{(s)}}_{a^{(r)} \cdot a^{(s)}}, \varepsilon^{j_1 j_2 \ldots j_D} a_{j_1}^{(r_1)} a_{j_2}^{(r_2)} \ldots a_{j_D}^{(r_D)} \right\}$$

$\eta =$ metric tensor, $\varepsilon =$ Levi-Civita symbol

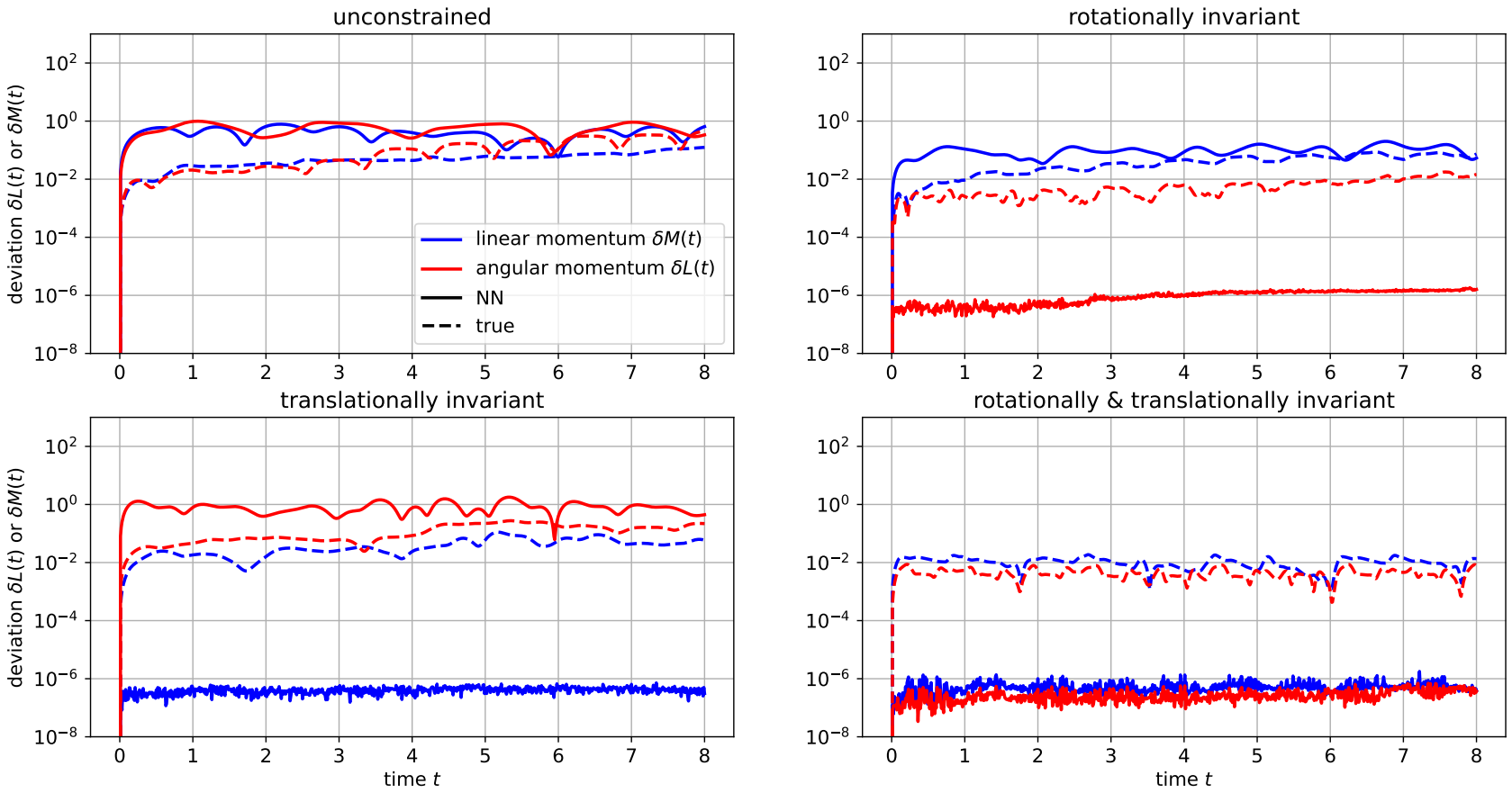
$$\underbrace{\mathcal{L}_{\text{NN}}\left(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \dot{\boldsymbol{x}}^{(1)}, \dot{\boldsymbol{x}}^{(2)}\right)}_{\text{unconstrained}} \mapsto \underbrace{\mathcal{L}_{\text{NN}}\left(\mathscr{R}\left(\boldsymbol{x}^{(1)} - \boldsymbol{x}^{(2)}, \dot{\boldsymbol{x}}^{(1)}, \dot{\boldsymbol{x}}^{(2)}\right)\right)}_{\text{constrained}}$$

Results IIa - Two-particle trajectories



Results IIb - Conservation of linear- & angular momentum

$$\delta M(t) = \|\boldsymbol{M}(t) - \boldsymbol{M}(0)\|_2 / \|\widetilde{\boldsymbol{M}}(0)\|_2 \qquad \delta L(t) = \|\boldsymbol{L}(t) - \boldsymbol{L}(0)\|_2 / \|\boldsymbol{L}(0)\|_2$$



Conclusion and Outlook

Summary

- Built **exact** conservation laws into neural network integrators for dynamical systems
- Strong inductive bias
- Improved prediction **accuracy** and **stability**

Future work

- Extend to more sophisticated problems
  point particles $\Rightarrow$ continuous systems
- Partially observed states in loss function
- Applications in other areas? Neural ODEs / ResNets, equivariance in Computer Vision?