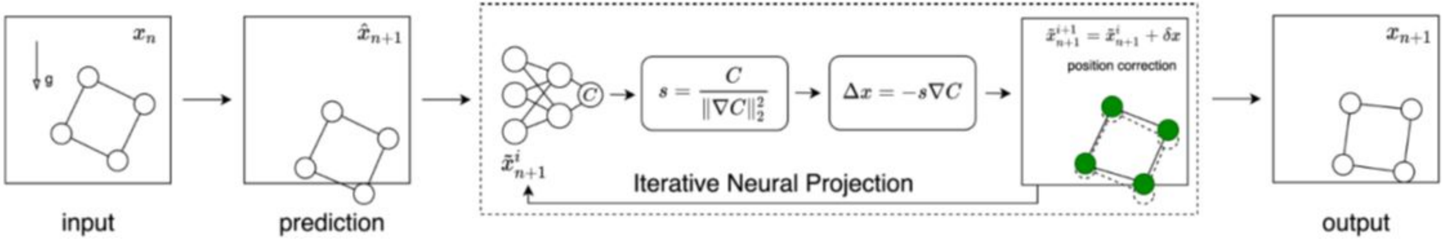
DARTMOUTH

Learning Physical Constraints with Neural Projections

Shuqi Yang, Xingzhe He, and Bo Zhu

Computer Science Department, Dartmouth College

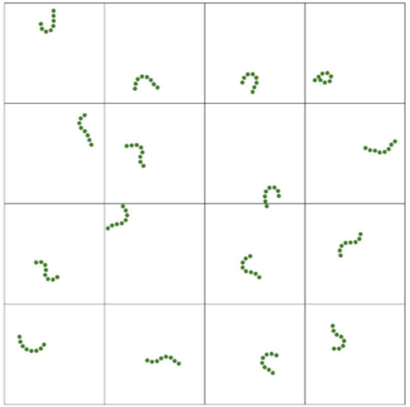


https://y-sq.github.io/proj/neural_proj/

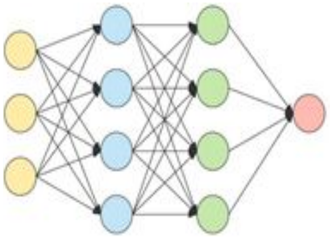
Background: Data-Driven Simulators

- Learning and predicting an unknown physical system

Observed data



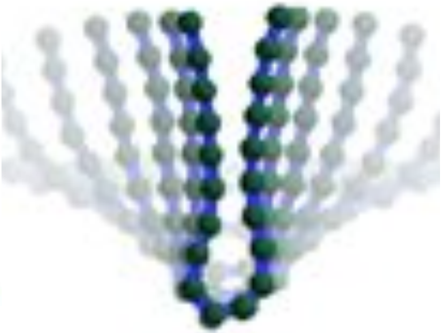
Learn the model



An ML Model:

Input: state_i
Output: state_i+1

Predict the dynamics



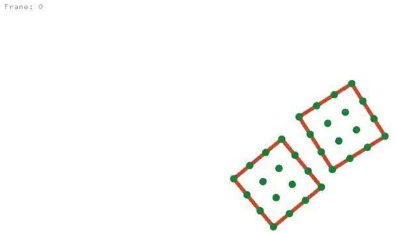
State_0

Future States

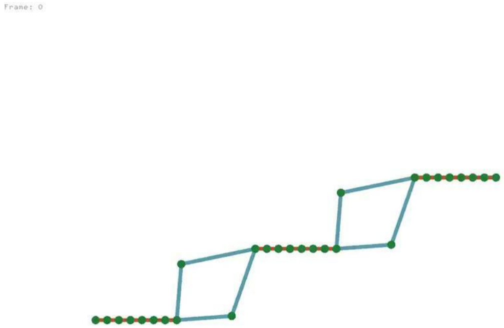
Background: Data-Driven Simulators

- How to describe a dynamic physical system?

System - 1



System - 2

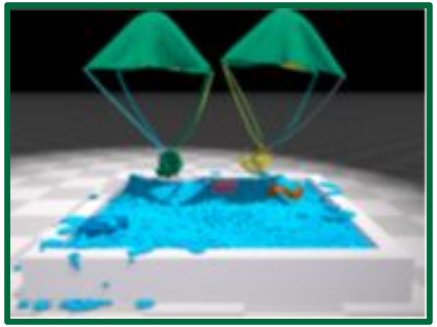


Simulation and Learning

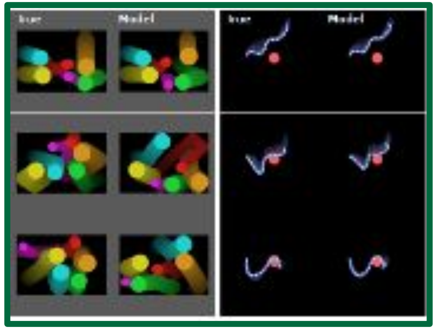
- Unified physics simulators in CG can inspire the design of learning algorithms to perceive physical systems

Physics simulation:
Systems with mathematical models
Predicting dynamics with unified models

Physics learning:
Systems with observation data
Predicting dynamics without known models



Use the priors from physical simulations to guide the design of network architectures

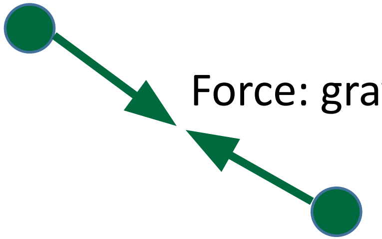


Figures from Maclin et al. Unified Particle Physics for Real-Time Applications. ACM TOG 33.
Battaglia et al. Interaction networks for learning about objects, relations and physics. NeurIPS 2016.

Simulation and Learning

- Mass-spring models
  - Compute the forces among each particles in the system;
  - Explicit time integration or implicit time integration.

- Interaction networks*

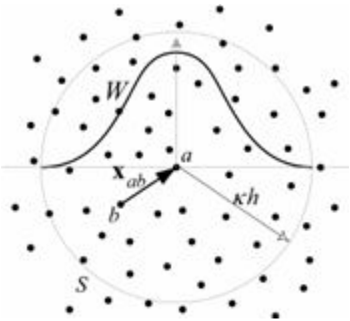


Force: gravity, spring, friction, …

* Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. In Advances in neural information processing systems, pages3194502–4510, 2016.

Simulation and Learning

- Smoothed particle hydrodynamics
  - A Lagrangian viewpoint to simulate fluids
  - Physical property approximated by weighted sum of the kernel



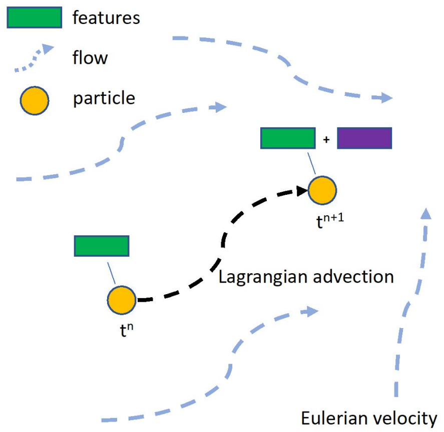
- Lagrangian fluid simulation*

* Ummenhofer, Benjamin, Lukas Prantl, Nils Thuerey, and Vladlen Koltun. "Lagrangian fluid simulation with continuous convolutions." In International Conference on Learning Representations. 2019.

Simulation and Learning

- PIC/FLIP
  - Eulerian-Lagrangian representation
  - Particle to grid; grid to particle

- AdvectiveNet*

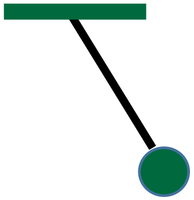


* Xingzhe He, Helen L. Cao, Bo Zhu. AdvectiveNet: An Eulerian-Lagrangian Fluidic Reservoir for Point Cloud Processing. International Conference on Learning Representations (ICLR), 2020.

Simulation and Learning

- Hamiltonian systems
  - Describe the system’s state using the position and momentum of the objects, whose evolution equation is given by the Hamilton's equations

- Hamiltonian networks*



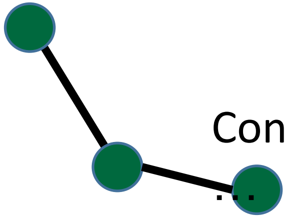
Energy: kinematics, potential, …

* Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. In Advances in Neural Information Processing Systems, pages 15353–15363, 2019.

Simulation and Learning

- Position-based dynamics
  - Model the system using the constraints(C(·)) that the position(x) should satisfy
  - Use a projection algorithm to correct the predicted positions such that they satisfy all the constraints: C(x) = 0

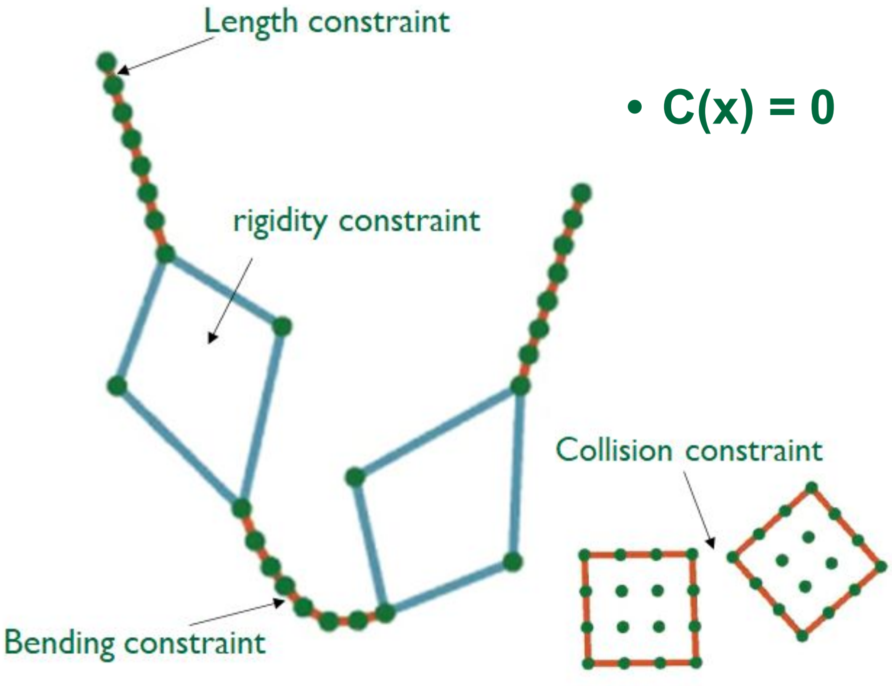
- Constraints (Ours)



Constraints: length, angle, position, …

Examples of Constraints

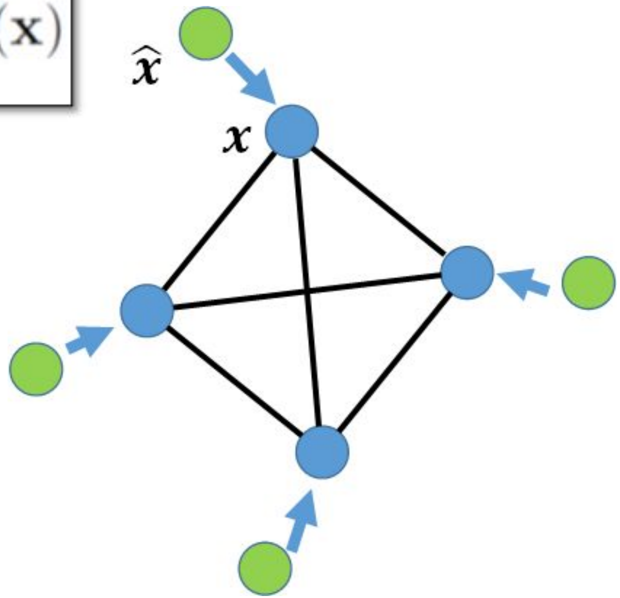
- Distance (length)
  - Distance(i, j) - constant = 0
- Angle (bending)
  - Angle(i, j, k) - constant = 0
- Shape (rigidity)
  - Shape - Initial_Shape = 0
- Non-penetration (collision)
  - Distance(i, other_objects) >= 0



- **C(x) = 0**

Position-based Dynamics: Variational Perspective

- $$\min_{\mathbf{x}} g(\mathbf{x}) = \frac{1}{\Delta t^2}(\mathbf{x}-\hat{\mathbf{x}})^T\mathbf{M}(\mathbf{x}-\hat{\mathbf{x}}) + \lambda^T\mathcal{C}(\mathbf{x})$$



- Prediction with forces
- Correction with constraints
- The correction step amounts to an energy minimization problem

Position-based Dynamics: Algorithm Overview

Algorithm 1 Position-based dynamics

for all vertices i **do**
  initialize $\mathbf{x}_i = \mathbf{x}_i^0, \mathbf{v}_i = \mathbf{v}_i^0, w_i = 1/m_i$
end for
loop
  for all vertices i **do** $\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t\, w_i \mathbf{f}_{\text{ext}}(\mathbf{x}_i)$
  for all vertices i **do** $\mathbf{p}_i \leftarrow \mathbf{x}_i + \Delta t \mathbf{v}_i$
  for all vertices i **do** genCollConstraints($\mathbf{x}_i \rightarrow \mathbf{p}_i$)
  loop solverIteration **times**
    projectConstraints($C_1, \ldots, C_{M+M_{\text{Coll}}}, \mathbf{p}_1, \ldots, \mathbf{p}_N$)
  end loop
  for all vertices i **do**
    $\mathbf{v}_i \leftarrow (\mathbf{p}_i - \mathbf{x}_i)/\Delta t$
    $\mathbf{x}_i \leftarrow \mathbf{p}_i$
  end for
  velocityUpdate($\mathbf{v}_1, \ldots, \mathbf{v}_N$)
end loop

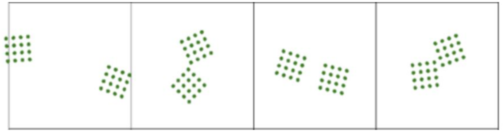
Enforcing **hard-coded** constraints

A Survey on Position Based Dynamics, 2017. Jan Bender, Matthias Müller and Miles Macklin.

Using Constraints to describe the physical systems

- Model the physical systems using its constraints: C(x) = 0

- Distance(i, j) - constant = 0
- Angle(i, j, k) - constant = 0



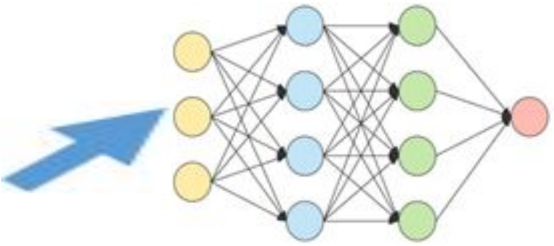
- Shape - Initial_Shape = 0
- Distance(i, other_objects) >= 0

- Directly related to human's perception
- A unified representation of physics
- Directly manipulate on the positions
- Inherently implicit scheme for stable prediction

Our Approach: Learning Constraints by Neural Projection

Algorithm 1 Position-based dynamics

for all vertices i **do**
  initialize $\mathbf{x}_i = \mathbf{x}_i^0, \mathbf{v}_i = \mathbf{v}_i^0, w_i = 1/m_i$
end for
loop
  for all vertices i **do** $\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t\, w_i \mathbf{f}_{\text{ext}}(\mathbf{x}_i)$
  for all vertices i **do** $\mathbf{p}_i \leftarrow \mathbf{x}_i + \Delta t\, \mathbf{v}_i$
  for all vertices i **do** genCollConstraints($\mathbf{x}_i \rightarrow \mathbf{p}_i$)
  loop solverIteration **times**
    projectConstraints($C_1, \ldots, C_{M+M_{\text{Coll}}}, \mathbf{p}_1, \ldots, \mathbf{p}_N$)
  end loop
  for all vertices i **do**
    $\mathbf{v}_i \leftarrow (\mathbf{p}_i - \mathbf{x}_i)/\Delta t$
    $\mathbf{x}_i \leftarrow \mathbf{p}_i$
  end for
  velocityUpdate($\mathbf{v}_1, \ldots, \mathbf{v}_N$)
end loop



Unknown constraints:

Replace this module with an NN

A Survey on Position Based Dynamics, 2017. Jan Bender, Matthias Müller and Miles Macklin.

Our Approach: Learning Constraints by Neural Projection

- Using a lightweight neural network to represent the constraints C(·)
- Iteratively solve the projection that moves the input positions($\hat{\mathbf{x}}$) to a state where C(x)=0 holds.

Algorithm 1: Projection Unit

Input: Constraint $C_{net}(\cdot)$, positions $\hat{\mathbf{x}}$.

$\tilde{\mathbf{x}}^1 = \hat{\mathbf{x}}$;

for $i = 1 \to N$ **do**

  $\lambda = C_{net}(\tilde{\mathbf{x}}^i)/|\nabla C_{net}(\tilde{\mathbf{x}}^i)|^2$;

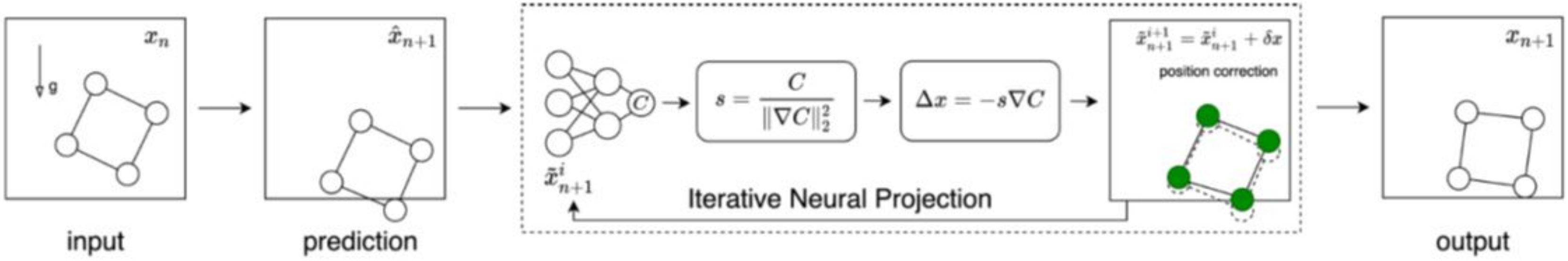
  $\delta\tilde{\mathbf{x}} = -\lambda \nabla C_{net}(\tilde{\mathbf{x}}^i)$;

  $\tilde{\mathbf{x}}^{i+1} = \tilde{\mathbf{x}}^i + \delta\tilde{\mathbf{x}}$;

end

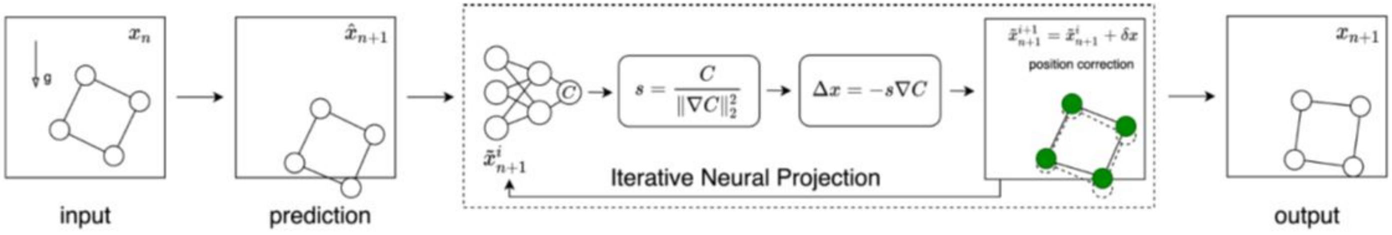
Output: Projected positions $\mathbf{x}$

Gradients calculated by auto differentiation



Our Approach: Learning Constraints by Neural Projection

- Prediction:
  - Advance the position of each particle with given external forces
- Correction:
  - A black-box NN module to enforce constraints
  - Multiple loops of projection iterations
  - Update the particle positions to the places where the black-box constraints are satisfied
- Velocity update:
  - Based on the positions in time n and n+1

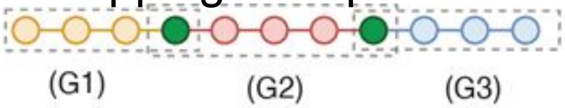


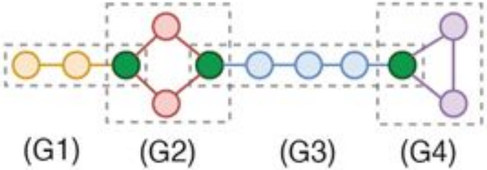
Loss is computed using the difference between the predicted and the groundtruth

（groundtruth data generated using physical simulation)

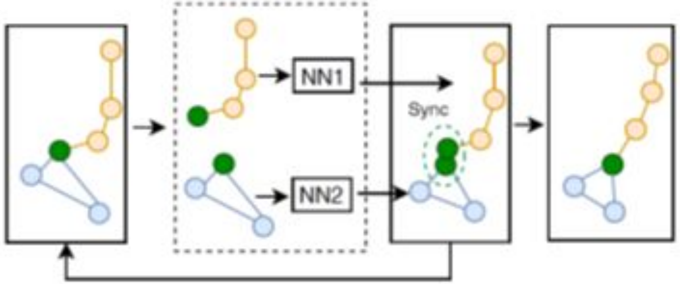
Our Approach: Multi-Group Representation

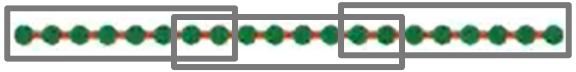
- Overlapping Groups:





- Sub-Networks:





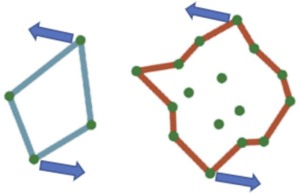
Algorithm 2: Multi-Group Projection

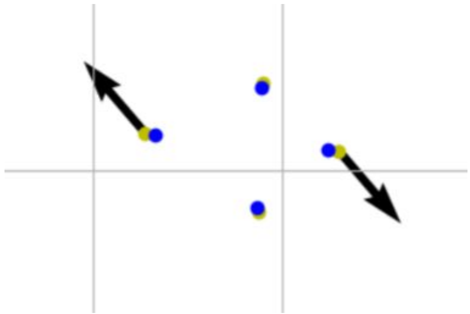
Input: NNs $C_{net_1}(\cdot), \cdots, C_{net_M}(\cdot)$,
Group of positions $\hat{\mathbf{x}}_1, \cdots, \hat{\mathbf{x}}_M$.

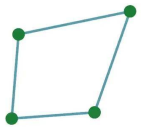
$\tilde{\mathbf{x}}^1 = \hat{\mathbf{x}}$;
for $i = 1 \rightarrow N$ **do**
  for $j = 1 \rightarrow M$ **do**
    $\tilde{\mathbf{x}}_j^{i+1} = Project(C_{net_j}, \tilde{\mathbf{x}}_j^i)$;
  end
  Synchronizing $\tilde{\mathbf{x}}^{i+1}$ among groups;
end
Output: Projected positions $\mathbf{x}$

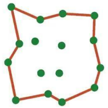
Animations of the predicted results

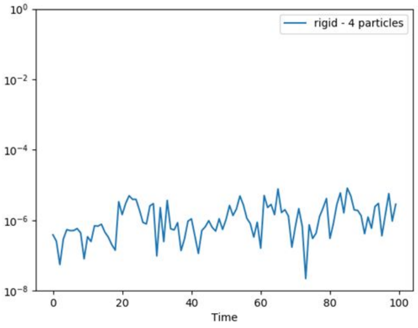
- A rigid body rotating with the external forces that sum to 0.

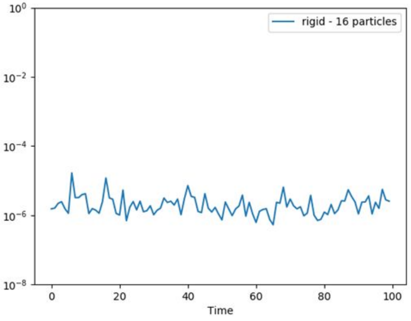






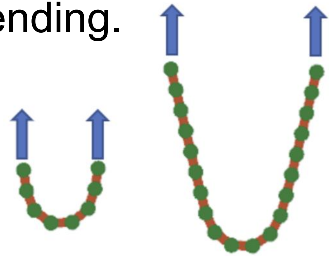


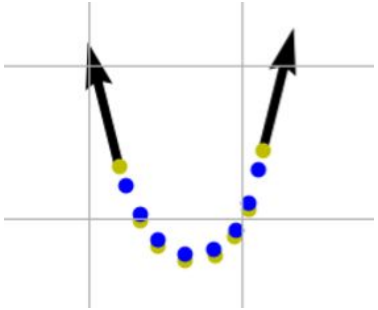


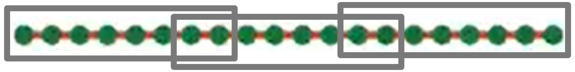


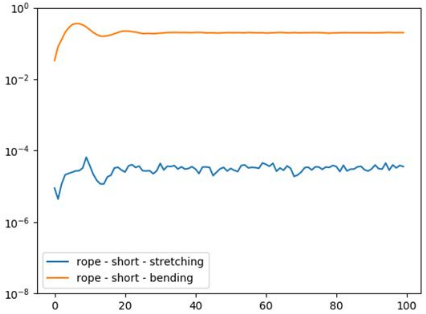
Animations of the predicted results

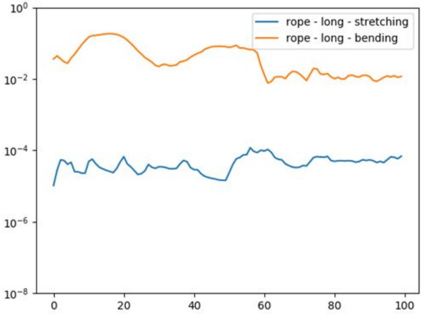
- A rope with stretching and bending.





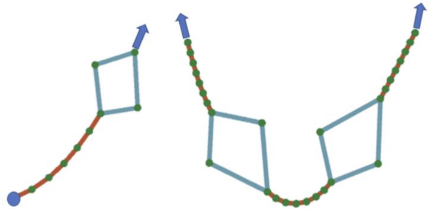


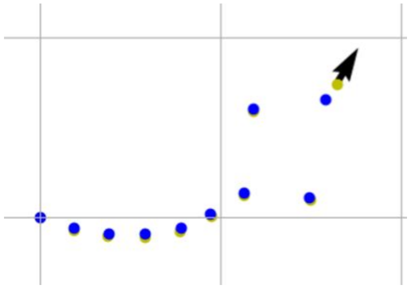


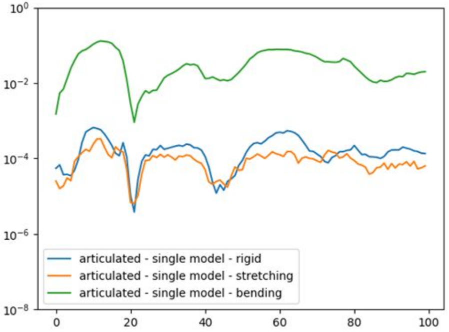


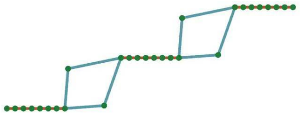
Animations of the predicted results

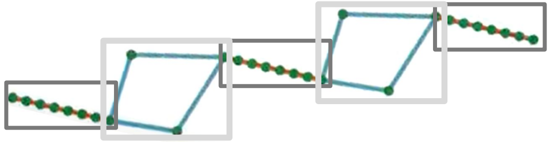
- The articulated body

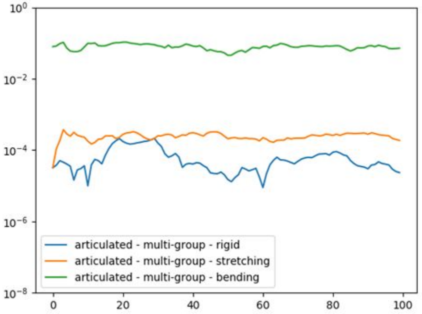






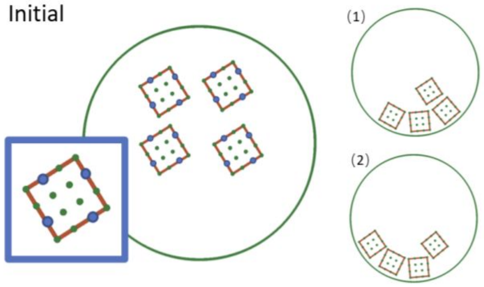


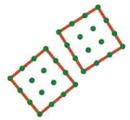


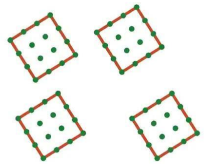


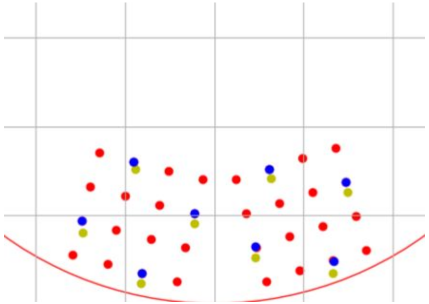
Animations of the predicted results

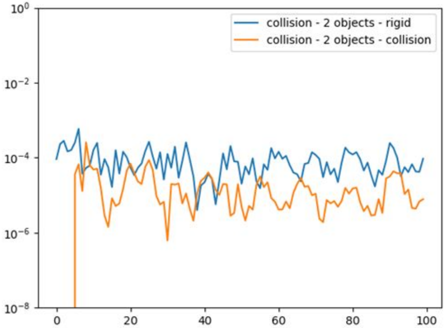
- Rigid body collisions

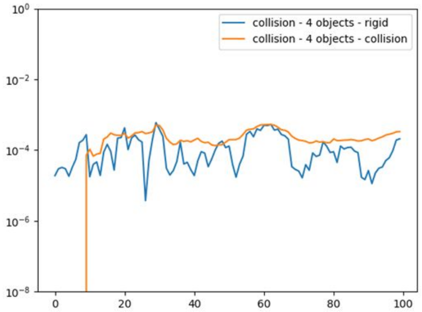






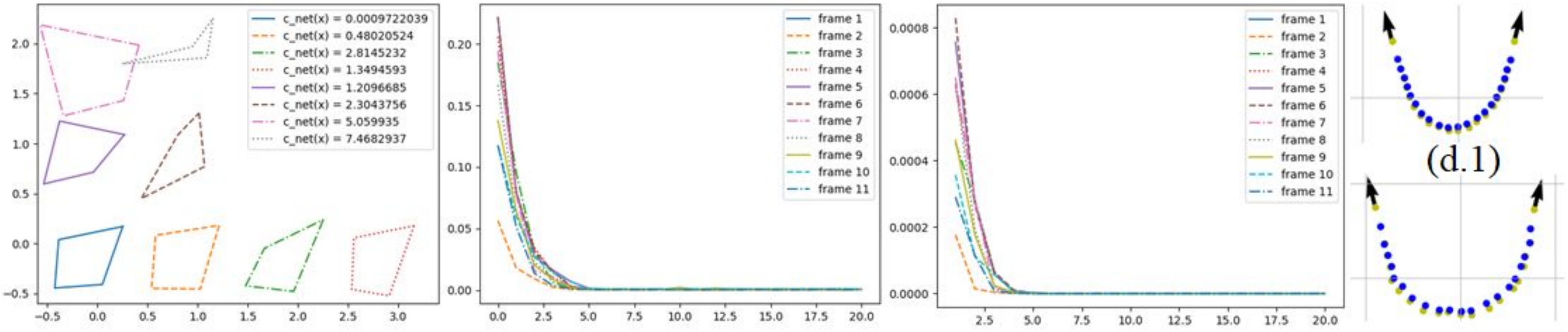






Discussions

- The learned constraints:
  - The value has a physical meanings; Future work to better separate each constraint.



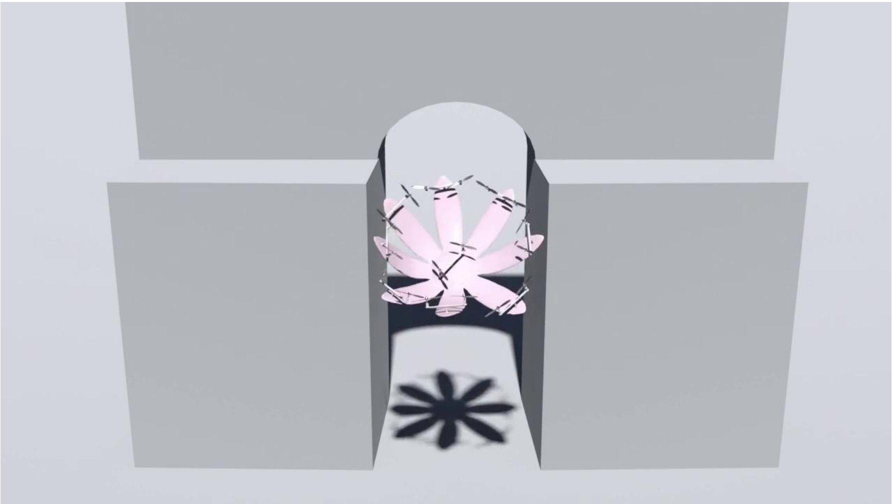
- The iterative projection:
  - A projection process is also used in other applications.
  - Fixed point problems.
- Network architectures for more types of systems.

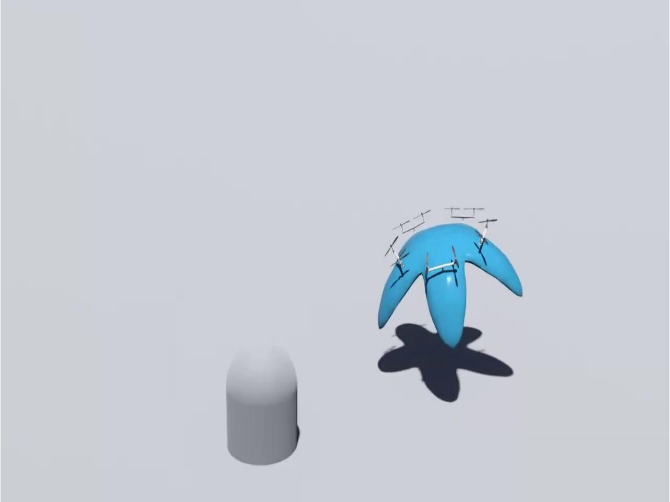
Take-home message

- Intersection between physics simulation and physics learning
- Use the priors from physical simulations to guide the design of network architectures
- Specific network architectures target at embedding specific types of priors
    - Two additional examples

More physics learning

- Soft Multicopter Control using Neural Dynamics Identification

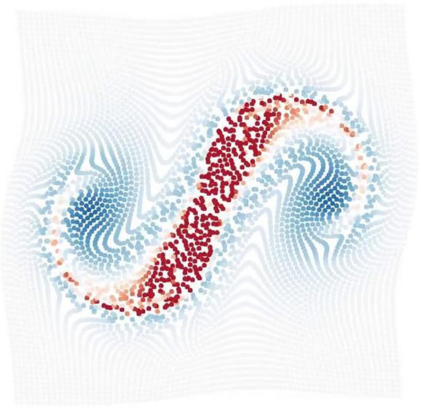


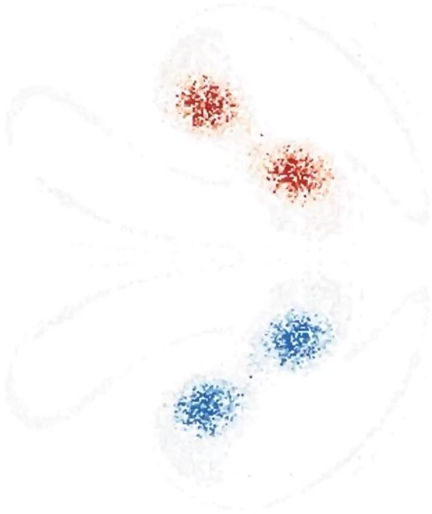


* Yitong Deng, Yaorui Zhang, Xingzhe He, Shuqi Yang, Yunjin Tong, Michael Zhang, Daniel M. DiPietro, Bo Zhu. Soft Multicopter Control using Neural Dynamics Identification. Conference on Robot Learning (CoRL 2020)
https://corlconf.github.io/corl2020/paper_396/

More physics learning

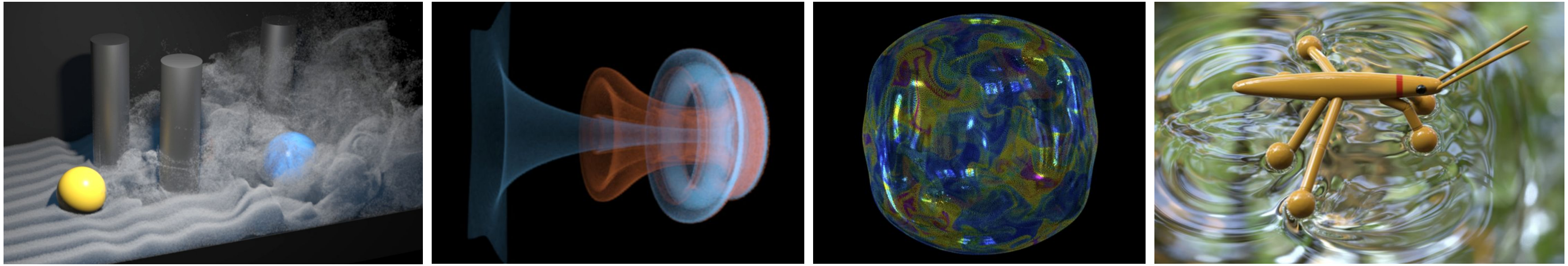
- Nonseparable Symplectic Neural Networks



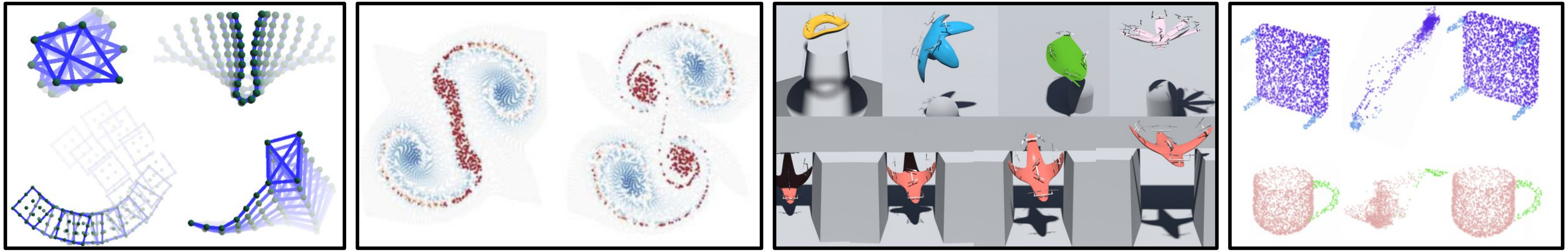


* Shiying Xiong, Yunjin Tong, Xingzhe He, Cheng Yang, Shuqi Yang, Bo Zhu. Nonseparable Symplectic Neural Networks. International Conference on Learning Representations (ICLR 2021)
https://shiyingxiong.github.io/proj/NSSNN/NSSNN

More work in our lab:
bridging physics simulation and machine learning



•Simulation: turbulent flows, vortex dynamics, bubbles, surface-tension-dominant contact



•Learning: solid systems, fluid systems, soft-bodied multicopter control, point cloud processing

Paper references: https://www.cs.dartmouth.edu/~bozhu/

Thank you!

For more information:

- **https://www.cs.dartmouth.edu/~bozhu/**
- **https://y-sq.github.io/**
- **https://www.youtube.com/playlist?list=PLPkEv32KJxkrZZDviP3XG9mJNBHBLBmlm**