Yale

Applications in Graphics and Scientific Simulations

CPSC483: Deep Learning on Graph-Structured Data

Rex Ying

Outline of Today's Lecture

- Physical Simulation in Science and Engineering
- Graph Neural Networks for Simulation
  - Graph Networks Simulator (GNS)
- Constrained-based Graph Networks Simulator (C-GNS)
- Application: Reservoir Simulation
  - Subsurface Graph Neural Network (SGNN)

Outline of Today's Lecture

- Physical Simulation in Science and Engineering

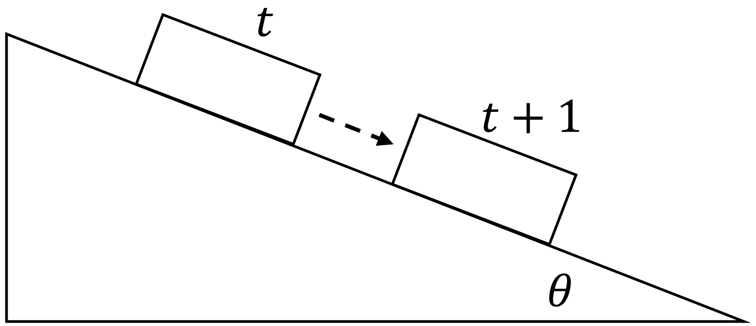
- Graph Neural Networks for Simulation
  - Graph Networks Simulator (GNS)

- Constrained-based Graph Networks Simulator (C-GNS)

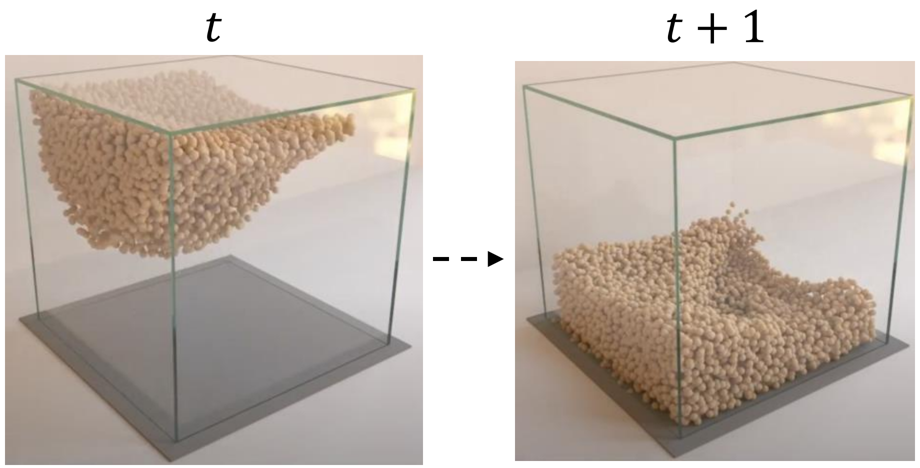
- Application: Reservoir Simulation
  - Subsurface Graph Neural Network (SGNN)

Physical Simulation in Science and Engineering (1)

- **What is simulation?**
  - Predicting the status of a(some) moving matter(s) at the given time



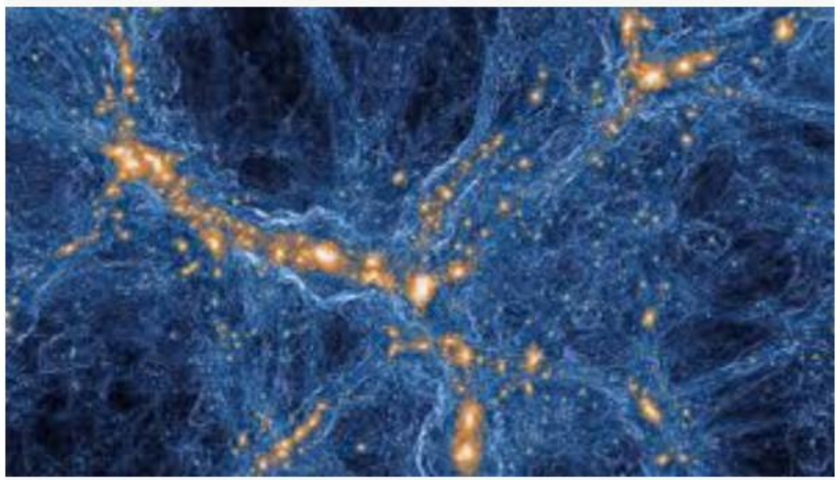
A moving block

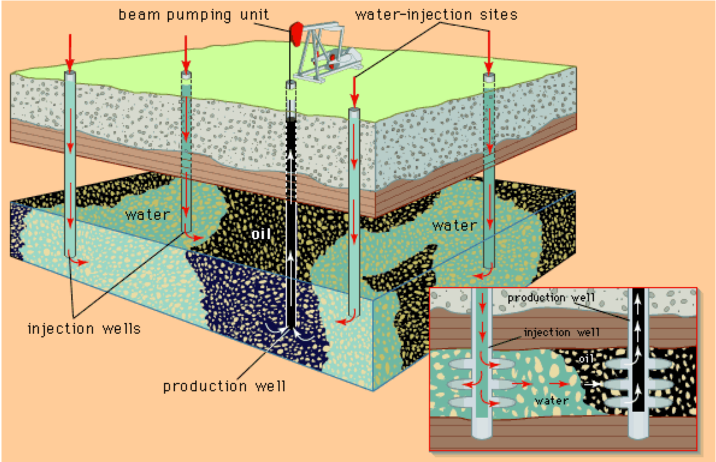


Some falling sands

Physical Simulation in Science and Engineering (2)

- Why do we need physical simulation?
  - For **science**, it can be used to test theories in real world
  - For **engineering**, it can be used to assess the performance of a planned system
  - Example:

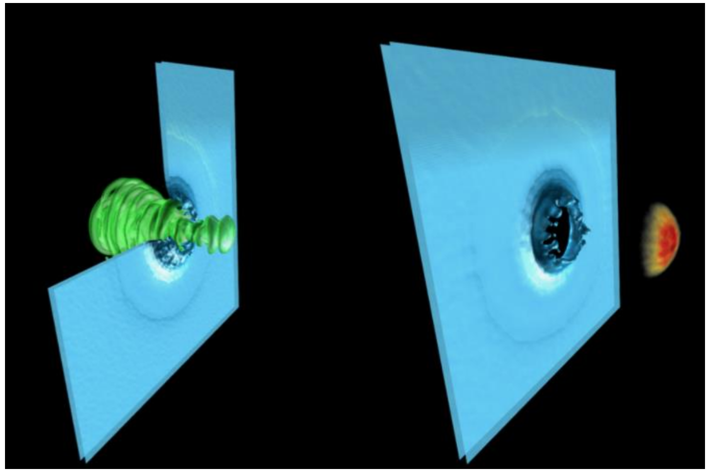




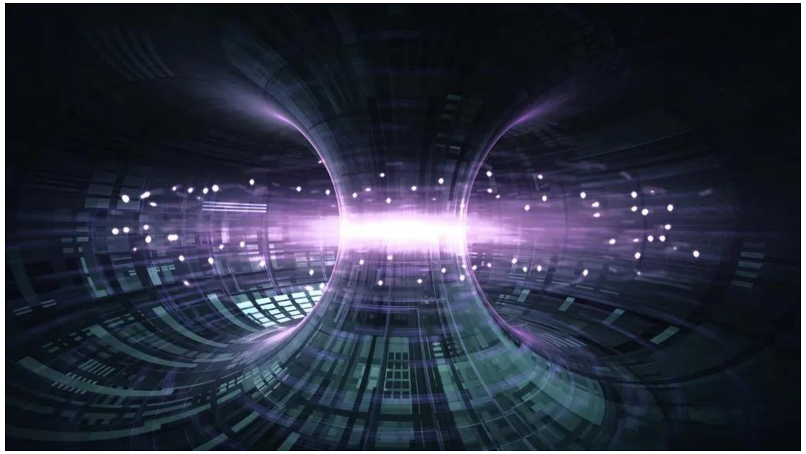
- **Particle-particle interactions**
  - Predicting galaxy formation with time
- **Reservoir simulation**
  - Predicting properties of fluids

Physical Simulation in Science and Engineering (3)

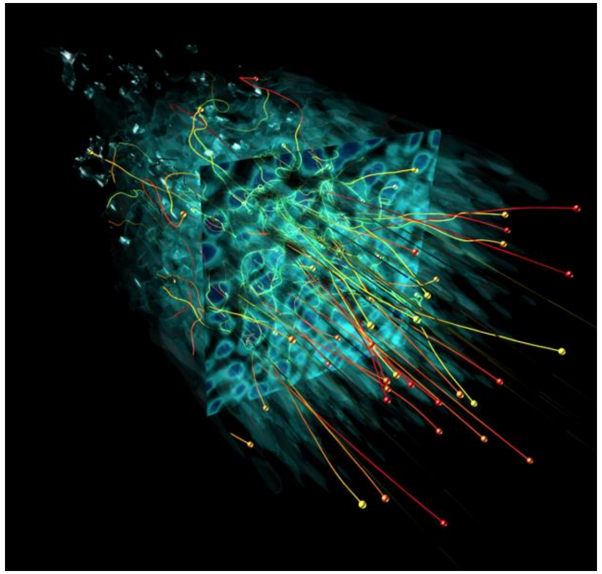
- High Performance Computing (HPC) for Simulation
  - Skyrocketing HPC can be applied to do some computationally intensive simulation



Laser-plasma particle acceleration



Fusion



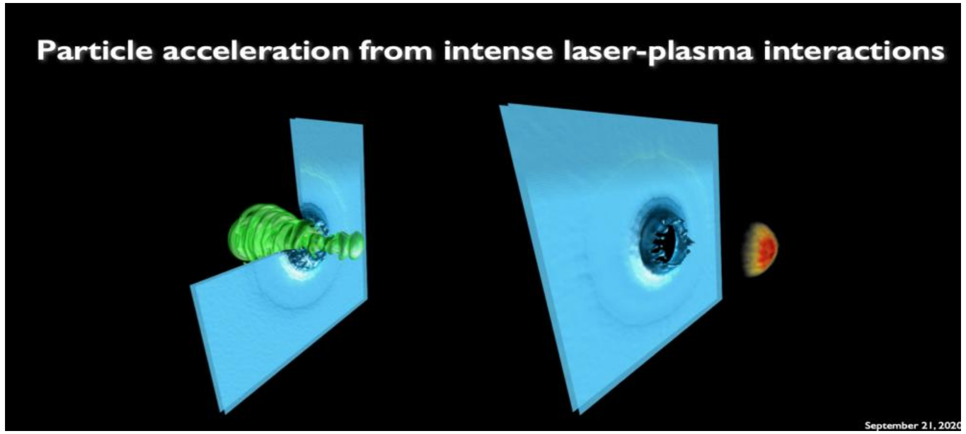
Cosmic-ray acceleration

Physical Simulation in Science and Engineering (4)

- **Characteristics**
  - **1) Large scale in size:** at the forefront of HPC
    - Nevertheless, even those large compute with long-time simulation may only do reasonably **small systems** in practice
    - E.g., for a reasonable 3D laser-plasma interaction system, it has **100B grid vertices, 1T particles, over 100k time steps**
    - Largest simulations (1/year): 10^{-1} of that scale, most studies: $< 10^{-2}$ of that scale

Physical Simulation in Science and Engineering (5)

- **Characteristics**
  - **2) Multi-scale and large dynamic range**
    - The dynamics involves multiple scales
    - Kinetic, many-body processes operating at **microscopic scales** significantly influence the fluid dynamics at **large scales** (and vice-versa)
    - E.g., Only **~0.01%** of the particles are accelerated but can carry **10-50%** of system energy



Machine Learning in Physical Simulation (1)

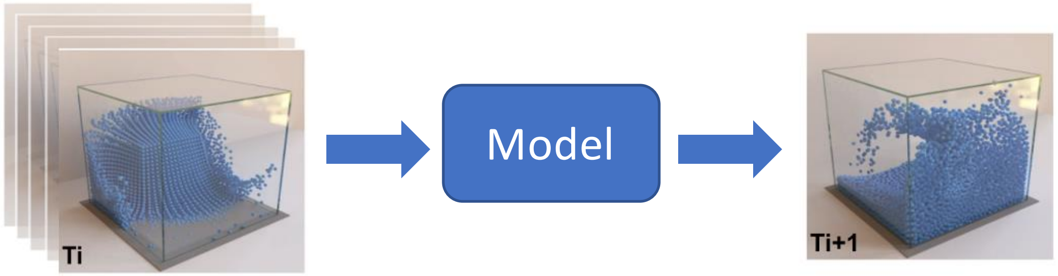
- An attractive alternative: **Machine learning**
  - Simulators trained from observed data can directly predict the next status

$$\text{Model}\left(\boxed{S_{T_0}, \dots, S_{T_i}}\right) = \boxed{S_{T_{i+1}}}$$

Previous status before step T_{i+1}

Predicted status at step T_{i+1}

- Prediction



Input

Target

Machine Learning in Physical Simulation (2)

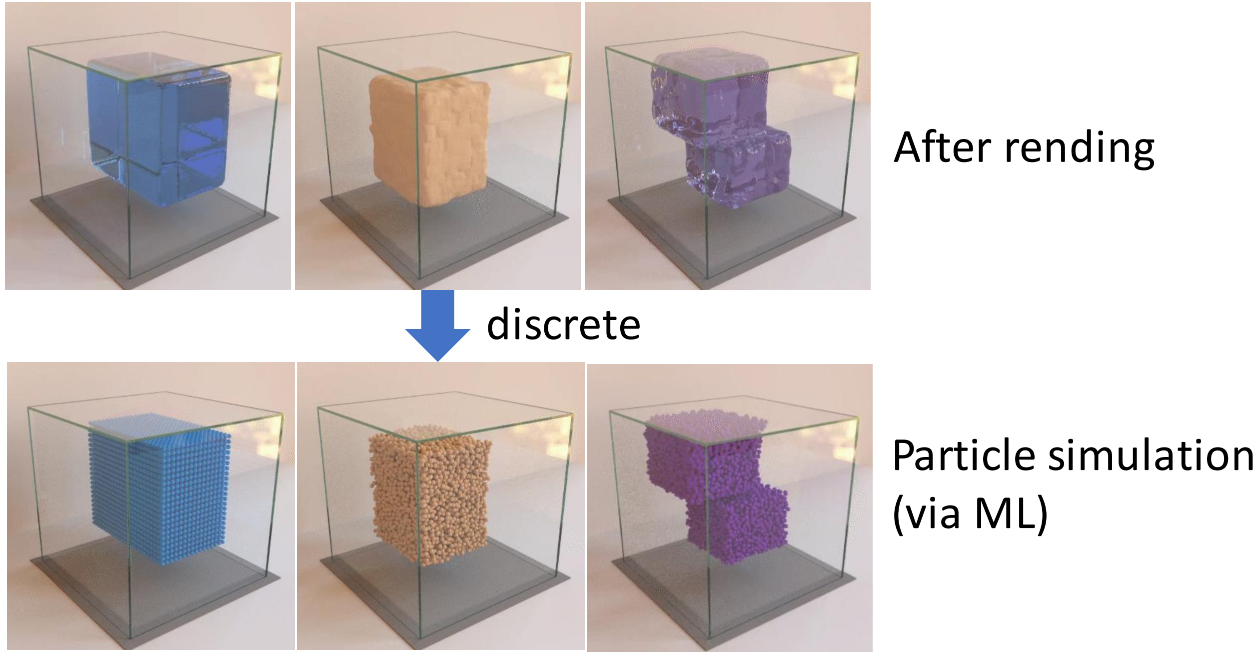
- An attractive alternative: Machine learning
  - Why do we use ML?
    - Multiscale dynamics raise an opportunity for optimization
    - Model can only focus on local information of each unit
    - Some powerful tools can be used to model the interactions between local neighbors, e.g., **Graphs**
  - Goal
    - For large-scale simulations, can we design **accurate** and **generalizable** ML models that capture the **essential dynamics** of the system with significant speedups?

Outline of Today's Lecture

- Physical Simulation in Science and Engineering
- Graph Neural Networks for Simulation
  - Graph Networks Simulator (GNS)
- Constrained-based Graph Networks Simulator (C-GNS)
- Application: Reservoir Simulation
  - Subsurface Graph Neural Network (SGNN)

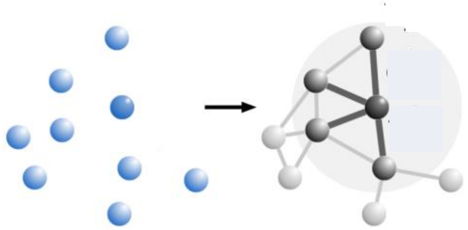
Graph Neural Networks for Simulation (1)

- Problem setting: Particle-based simulation
  - Given the initial conditions of particles (position, velocity)
  - Simulate the evolution of material over long time range



Graph Neural Networks for Simulation (2)

- A powerful tool for learning to simulate: **Graph**
  - Rich physical states are represented by graphs of interacting particles
  - Node: Particle, Edge: Interaction



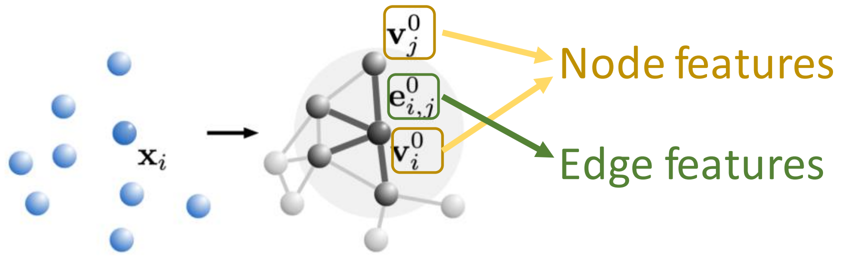
- Why use graph
  - Graph structure naturally captures the **local interaction** (e.g., collision) between a particle and its adjacent particles
  - Complex dynamics can be approximated by modeling such interaction

Graph Neural Networks for Simulation: GNS (1)

- Graph Network-based Simulator (GNS)
  - m: message passing step
  - v_i^m: feature of node i at m step
  - $\mathrm{e}_{i,j}^m$: interaction between node i and node j at m step
  - For every prediction, GNS can be carried out in three steps
    - **Construct graph**: Connect the node to its neighbors
    - **Pass messages**: Update the status of each node
    - **Extract dynamics info**: Predict the next status

GNS: Graph Construction

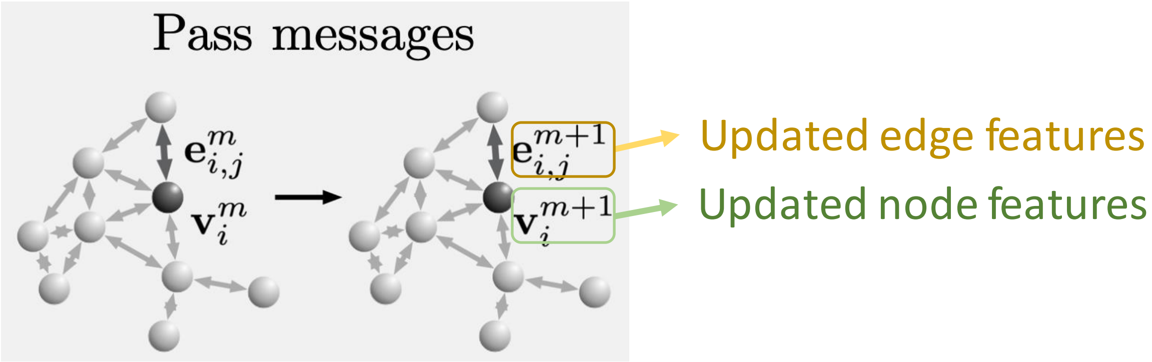
- Construct graph
  - The other nodes within the radius are regarded as neighbors



  - Node features
    - Node Position
    - Previous velocities
    - Particle type
  - Edge features
    - Relative positional displacement
    - Magnitude

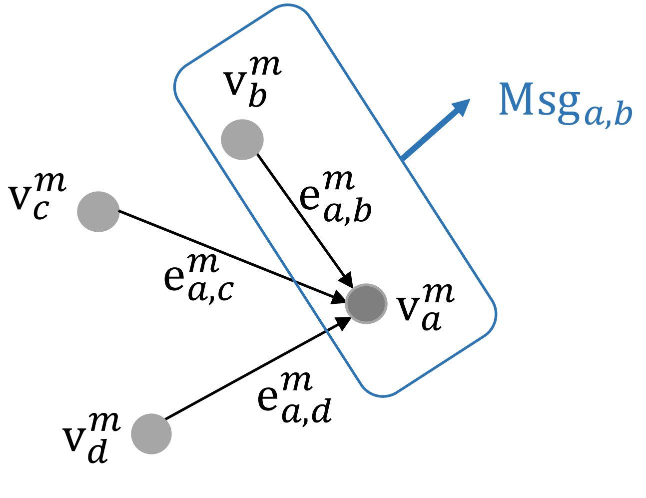
GNS: Message Passing (1)

- Pass messages
  - Aggregate the information from multi-hop neighbors
  - A stack of M graph convolution layers
    - Layer-0 inputs are input features $V^0 = \{v_i^0\}, E^0 = \{e_{i,j}^0\}$
    - Layer-m inputs are the updated embeddings V^m, E^m from previous $m - 1$ layers



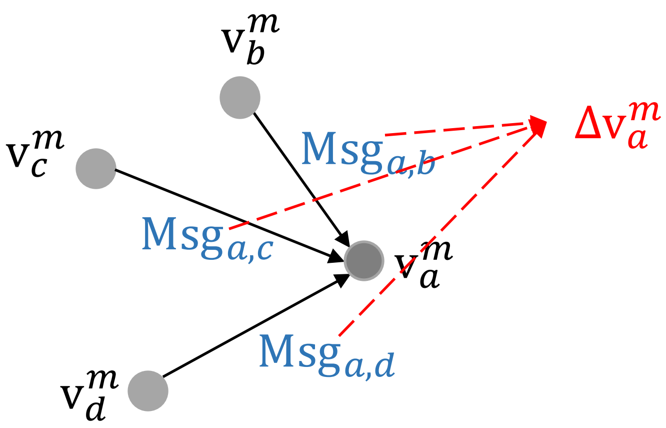
GNS: Message Passing (2)

- Example of passing messages



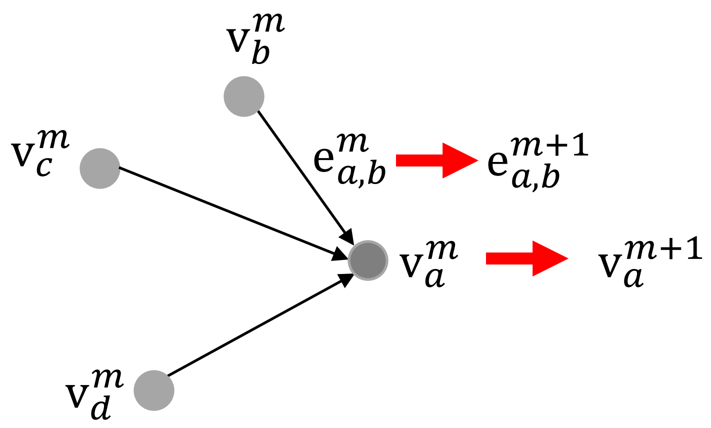
$\mathrm{Msg}_{a,b} = \mathrm{MsgPass}(v_b^m, e_{a,b}^m, v_a^m)$

Message passing



$\Delta v_a^m = \mathrm{Agg}(\mathrm{Msg}_{a,*})$

Aggregate message



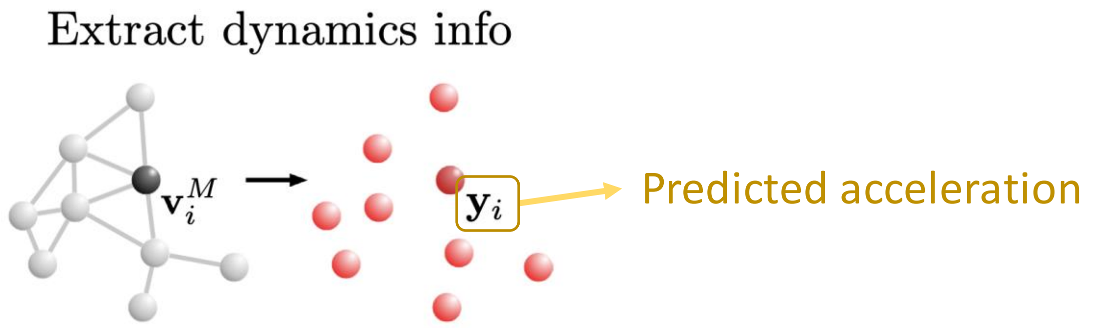
$v_a^{m+1} = v_a^m + \mathrm{Update}(v_a^m, \Delta v_a^m)$

$e_{a,b}^{m+1} = e_{a,b}^m + \mathrm{Msg}_{a,b}$

Update node and edge representation

GNS: Extract Dynamics Info

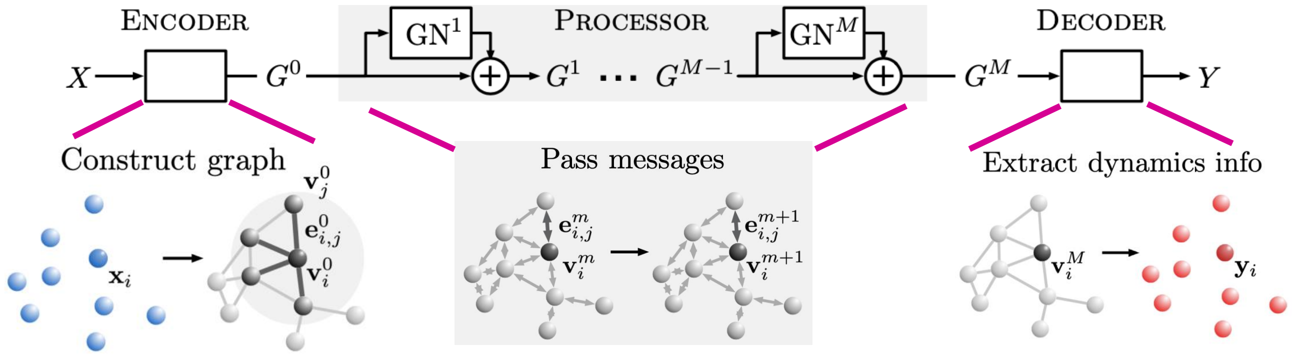
- Extract dynamics info
  - Use updated node representation to predict next status
    - Feed the representation into a learnable **MLP** to predict current acceleration
    - Apply **Euler integrator**[1] to update position and velocity
  - Optimizing for acceleration is equivalent to optimizing for position
    - Acceleration is computed as first order finite difference from the position



[1] https://en.wikipedia.org/wiki/Euler_method

GNS: Pipeline (1)

- Pipeline of GNS



Encoder

- Node input features:
  - Position
  - Previous velocities
  - Particle type
- Edge input features: displacements
- Embed features with MLP
- Construct neighborhood graph

Processor

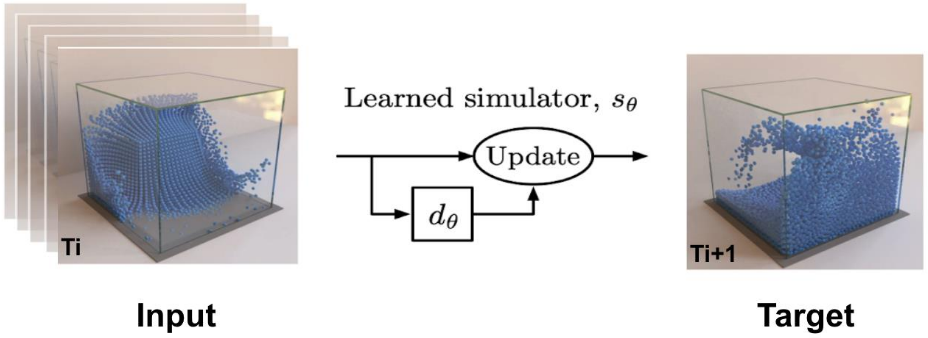
- Message-passing layers (x10)
  - Use neighborhood graph
  - Edge function: MLP
  - Node function: MLP
- Outputs embeddings (next step)

Decoder

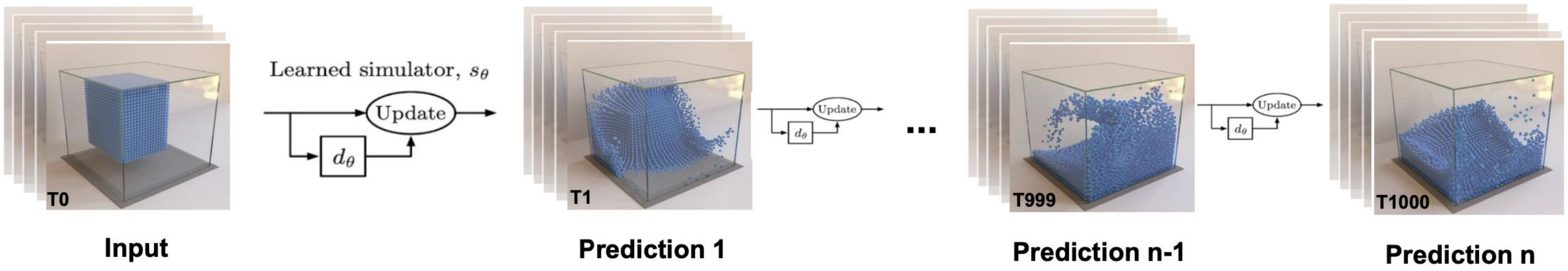
- Decode acceleration
- Feed into Euler integrator to obtain position and velocity

GNS: Pipeline (2)

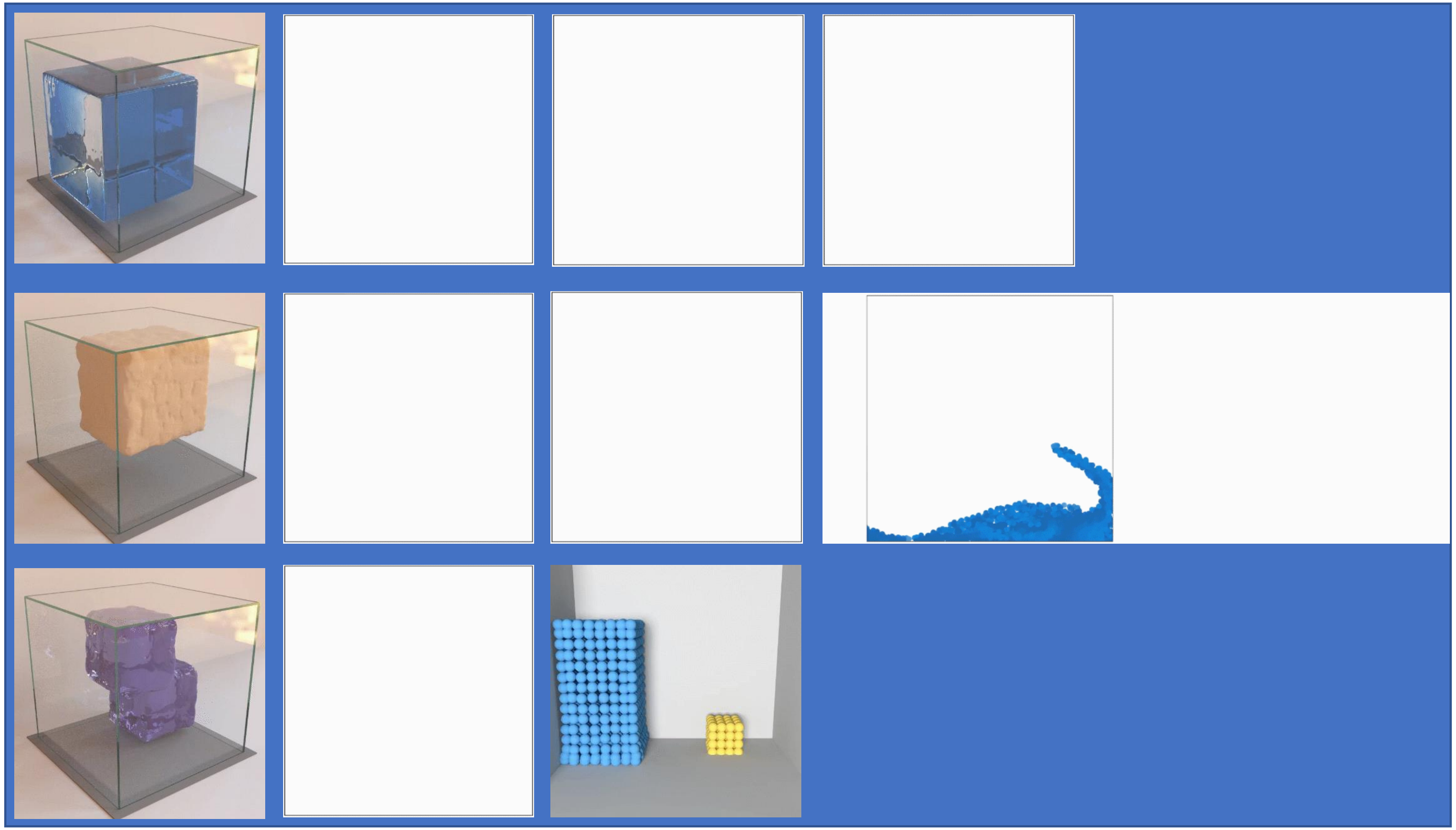
- Training time: one-step minibatch training



- Inference time: 1000s of steps

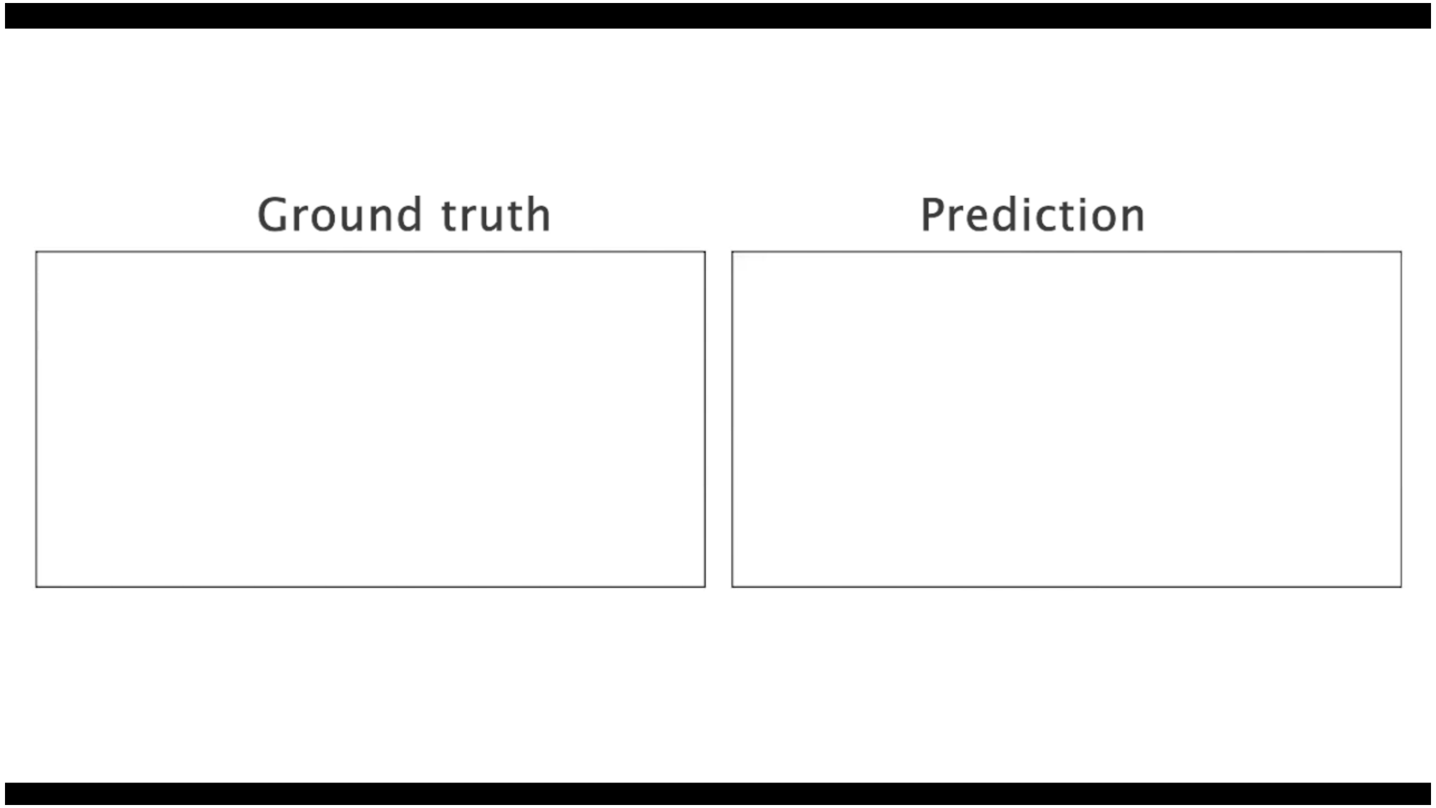


GNS: Demo (1)



GNS: Demo (2)

- More complex scenarios

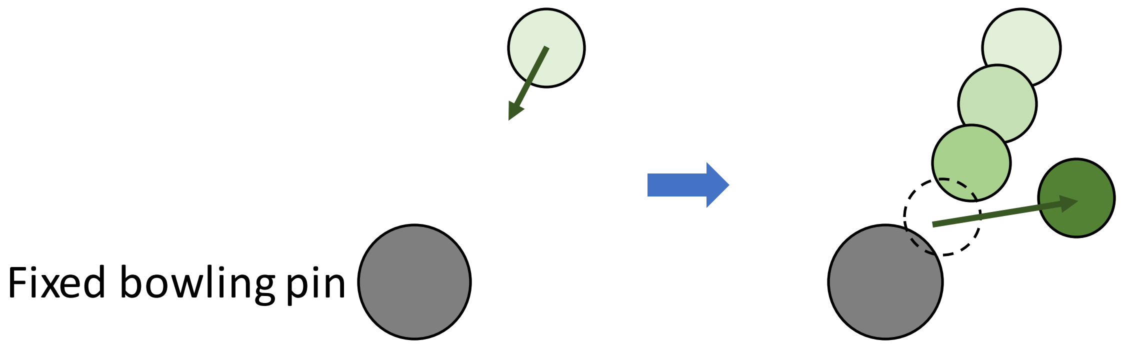


Outline of Today's Lecture

- Physical Simulation in Science and Engineering
- Graph Neural Networks for Simulation
  - Graph Networks Simulator (GNS)
- Constrained-based Graph Networks Simulator (C-GNS)
- Application: Reservoir Simulation
  - Subsurface Graph Neural Network (SGNN)

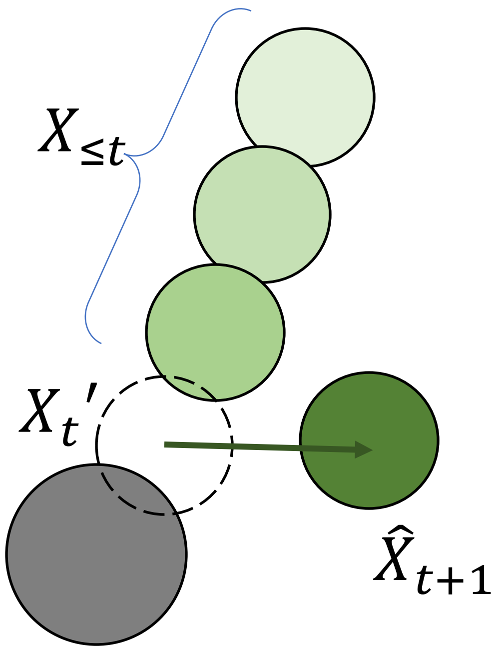
Graph Neural Networks for Simulation: C-GNS

- Motivation of C-GNS
  - GNS applies an **explicit** forward model to calculate next state directly from the current one
  - However, an equally valid way is to explain the motion/interaction in terms of **constraint** satisfaction
  - Let's consider an example: a bowling ball colliding with a fixed bowling pin



Constraint-based Simulator (1)

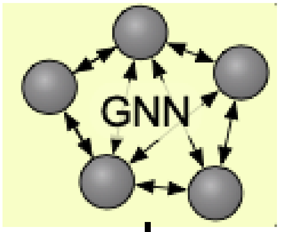
- Collision between two balls
  - Fixed ball causes the other one to move
  - **Physical constraint**: Objects do not overlap
  - Notation
    - $X_{\leq t}$: position of object before time step t



Constraint-based Simulator (2)

- The way GNS models
  - Predicts the next position **directly**
    - Directly predict the position of ball after collision
  - Requires to implicitly resolve physical constraints
    - Objects do not overlap
- Many traditional physical simulators don't work like that!

$X_{\le t}$

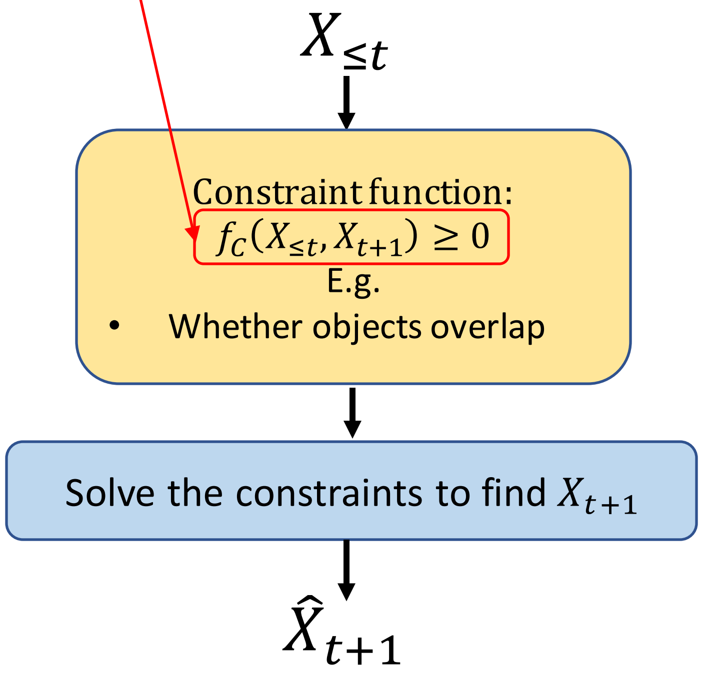


$\hat{X}_{t+1}$

Constraint-based Simulator (3)

- The way constraint-based simulator models
  - Define a **constraint function** to quantify how well X_{t+1} agrees with $X_{\le t}$
  - Find X_{t+1} by solving the constraints

Quantify how well X_{t+1} agrees with $X_{\le t}$



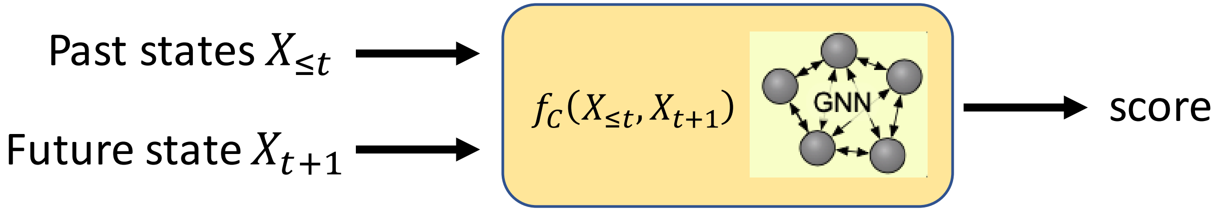
Rubanova, Yulia, et al. "Constraint-based graph network simulator." ICML, 2022.

Comparison between GNS and C-GNS

- Graph Network-based Simulator (GNS)
  - Given previous states, GNS **directly** predicts the state update at the next step
- Constraint–based Graph Network Simulator (C-GNS)
  - Constraint (f_C): a **learnable** function which indicates the next step state is **consistent with** the current and previous states
  - Different from GNS, C-GNS begins with an initial state update and **iteratively refines** until it satisfies the constraint

Constrained-based Graph Networks Simulator (1)

- The learnable constraint function f_C:
  - **GNN-based** constraint function
  - f_C determines whether X_{t+1} is consistent with $X_{\le t}$ (the smaller the better)

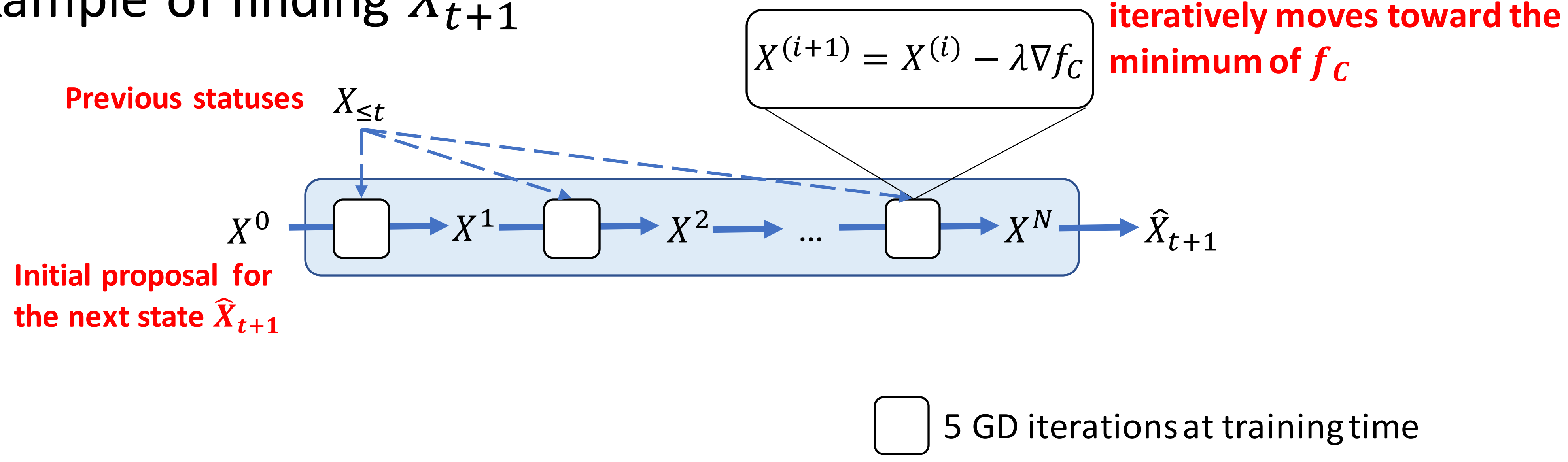


- Predict the next status by **minimize the constraint function**

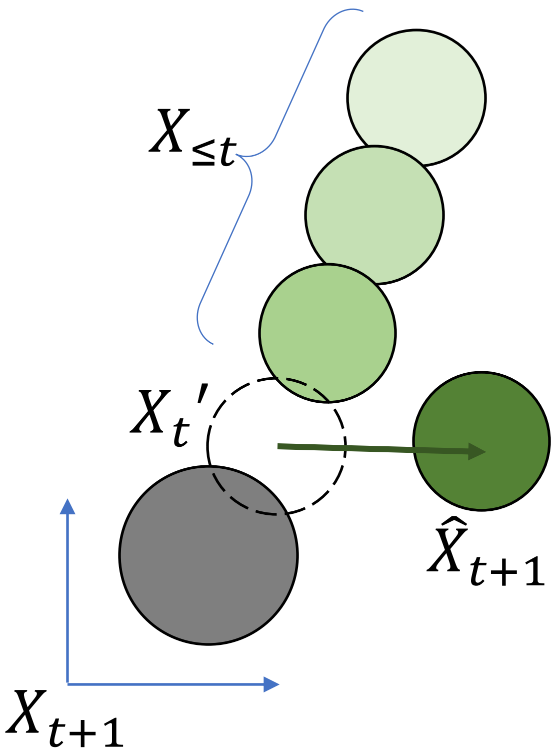
$$\hat{X}_{t+1} = \arg\min_{X_{t+1}} f_C(X_{\le t}, X_{t+1})$$

Constrained-based Graph Networks Simulator (2)

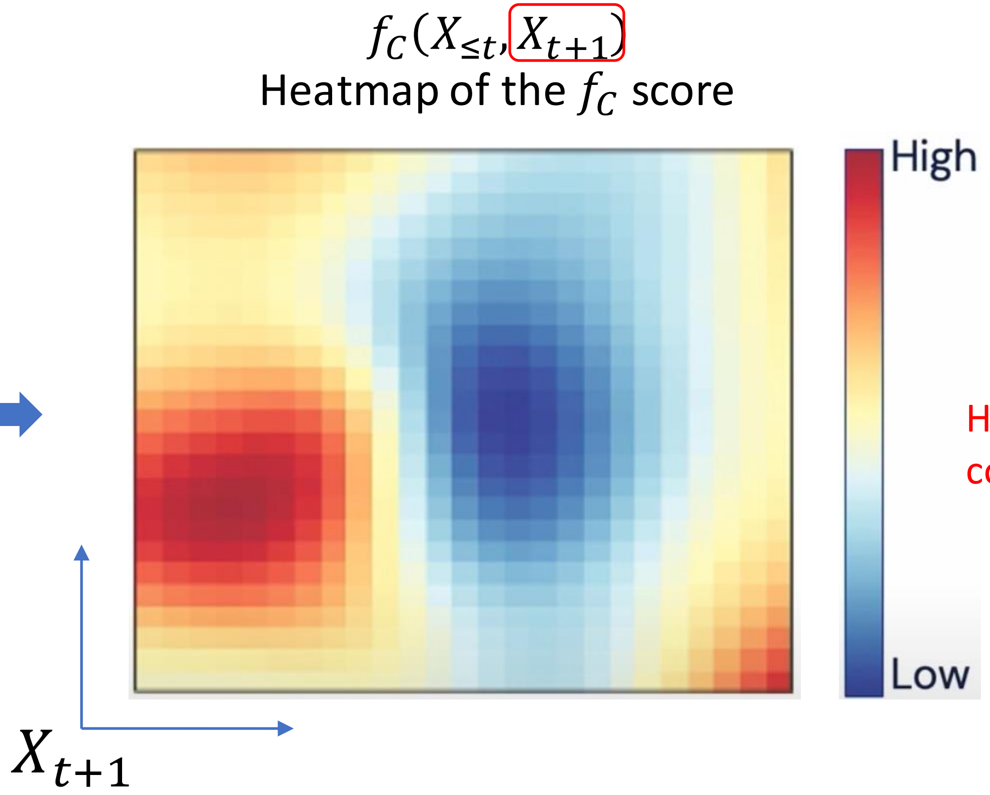
- How can we find a minimum of a function?
  - Gradient descent (GD) !
- An example of finding $\hat{X}_{t+1}$



Constrained-based Graph Networks Simulator (3)



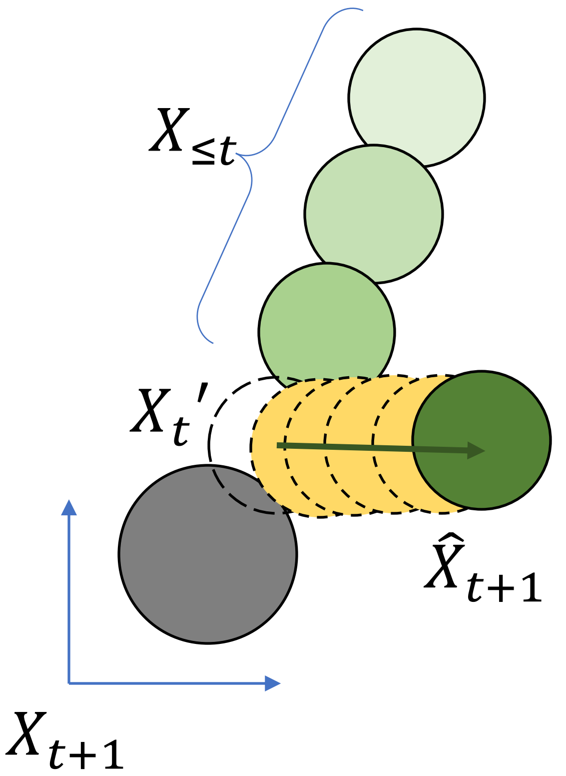
Position of next status

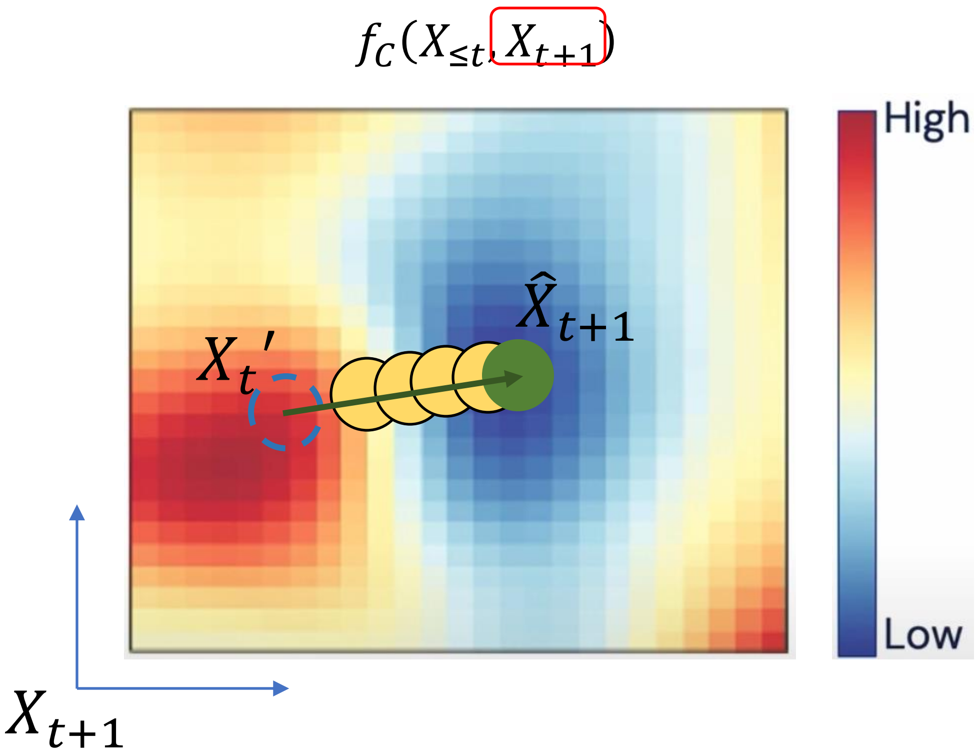


Position of next status

Higher score means less consistency

Constrained-based Graph Networks Simulator (4)

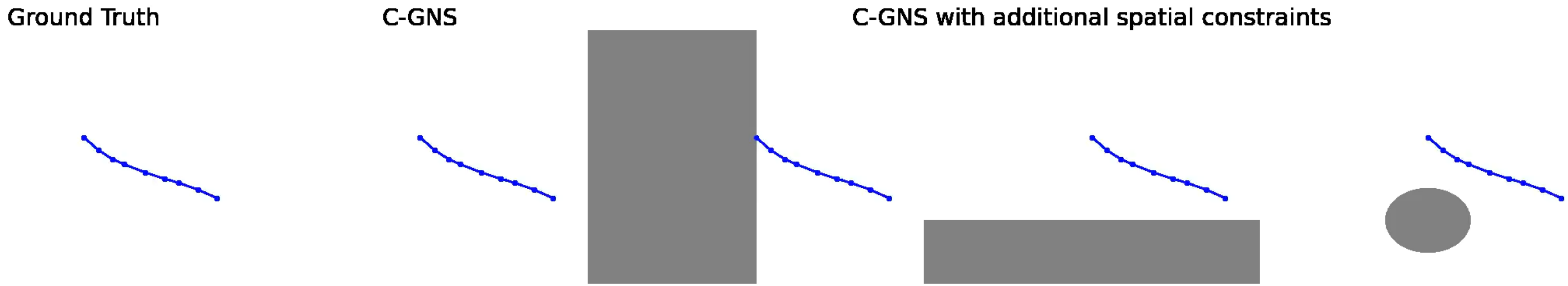




Refine the position to find the minimum of $f_C(X_{\leq t}, X_{t+1})$

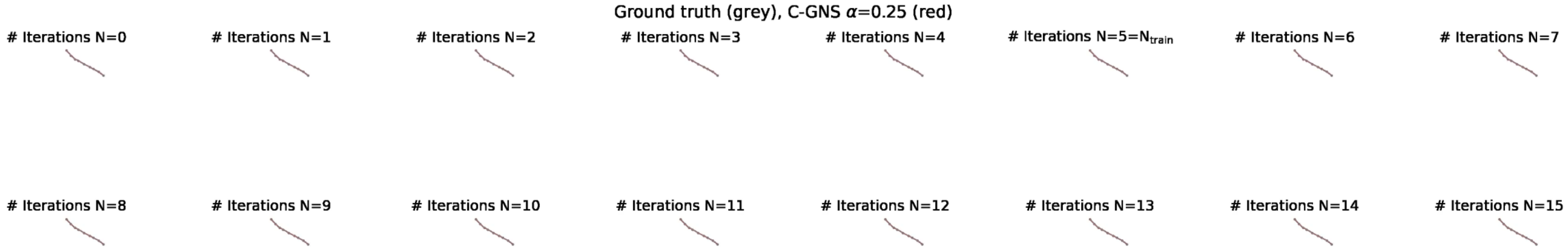
Demo (1)

- More demos can be found in https://sites.google.com/view/constraint-based-simulator



Demo (2)

- Performance with different number of iterations



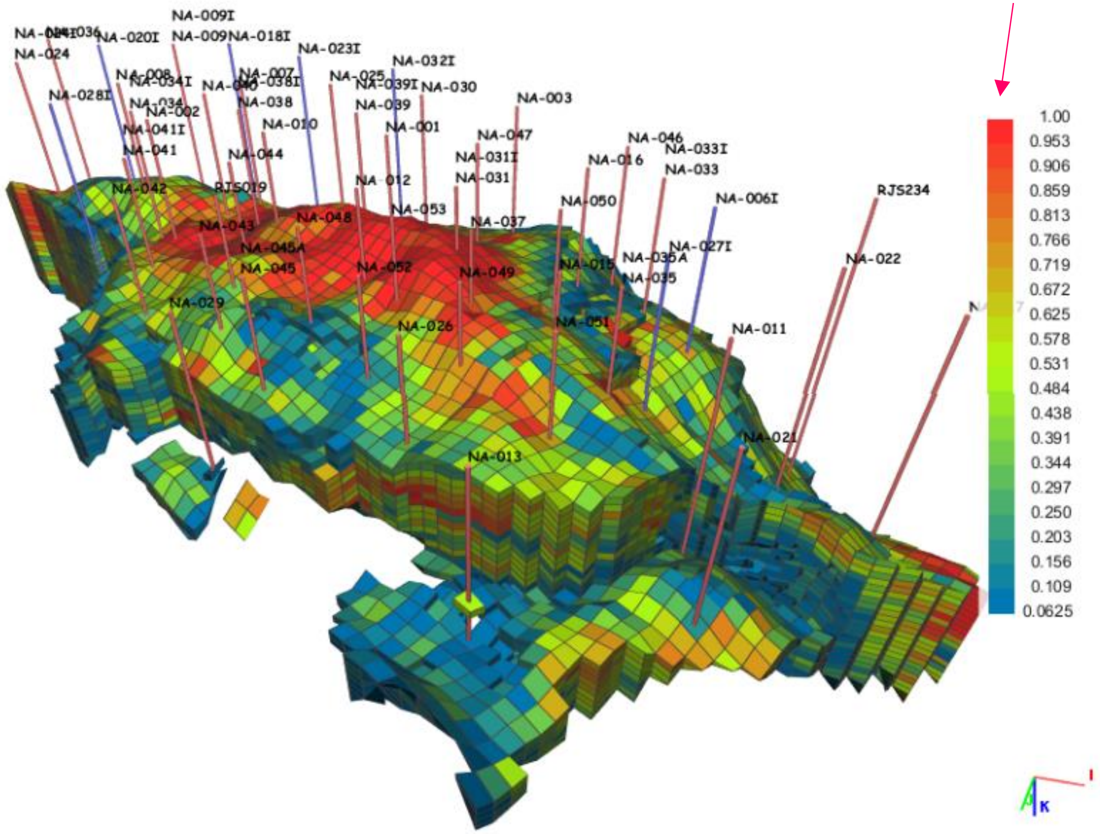
Outline of Today's Lecture

- Physical Simulation in Science and Engineering
- Graph Neural Networks for Simulation
  - Graph Networks Simulator (GNS)
- Constrained-based Graph Networks Simulator (C-GNS)
- Application: Reservoir Simulation
  - Subsurface Graph Neural Network (SGNN)

Reservoir Simulation

- Water and oil exist in the porous rock which is discretized into cells
- Water is pumped into the reservoir using injectors (blue pins)
- Preferential fluid flow path from the injectors to the oil producers (red pins)

Oil saturation level

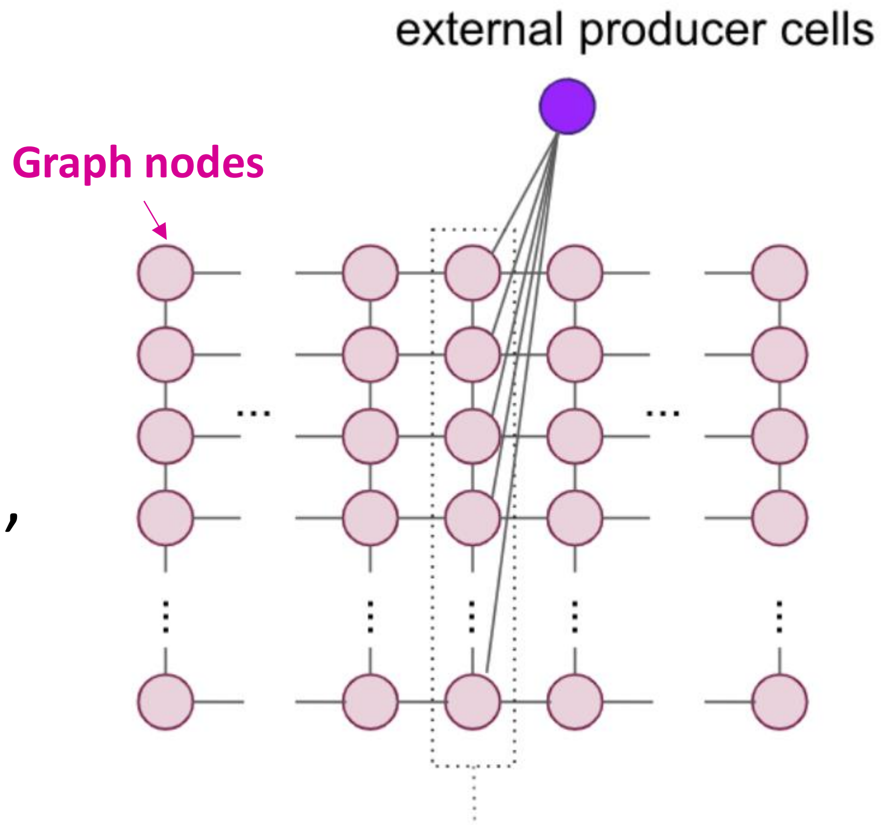


Problem formulation

- **Input**: A large grid with million of cells. Each cell in grid has static and dynamic features
  - **Static features**: transmissibility, por-vol, porosity etc.
  - **Dynamic features**: water/oil/gas (in barrels) and pressure (in PSI)
    - Given the dynamic features for the initial 3 steps (time step 0, 1, 2)
    - Size**:** 1M to 15M cells
  - **Well metadata and well rates**
- **Output**
  - **Water/oil/gas (in barrels) and pressure (in PSI) predictions** for all cells (including injectors/producers) at subsequent time steps: 3, 4, ..., 22 (20 months)
  - **Production prediction**: 3 values WSWP, WWPR, WOPR for each producer cell at each time step - oil production prediction

Reservoir Simulation

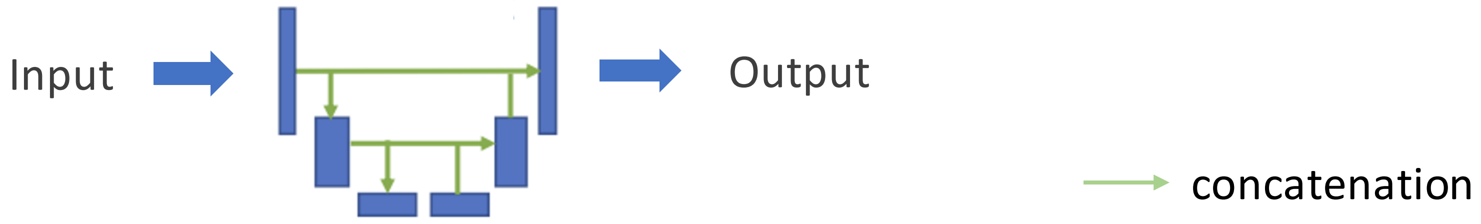
- **Graph nodes** consist of grid cells and special cell types (injector, producer, boundary cells)
- **Graph edges** consist of connectivity between adjacent grid cells (allows transmission of water / oil)
- Apply **GNS framework** to simulate the oil, water saturation at each cell, and pressure at each cell
  - Given the previous statuses of graph nodes, predict the next status



[1] Long, Jonathan, et al. "Fully convolutional networks for semantic segmentation." *CVPR*. 2015.

Improvement: Hybrid Model (1)

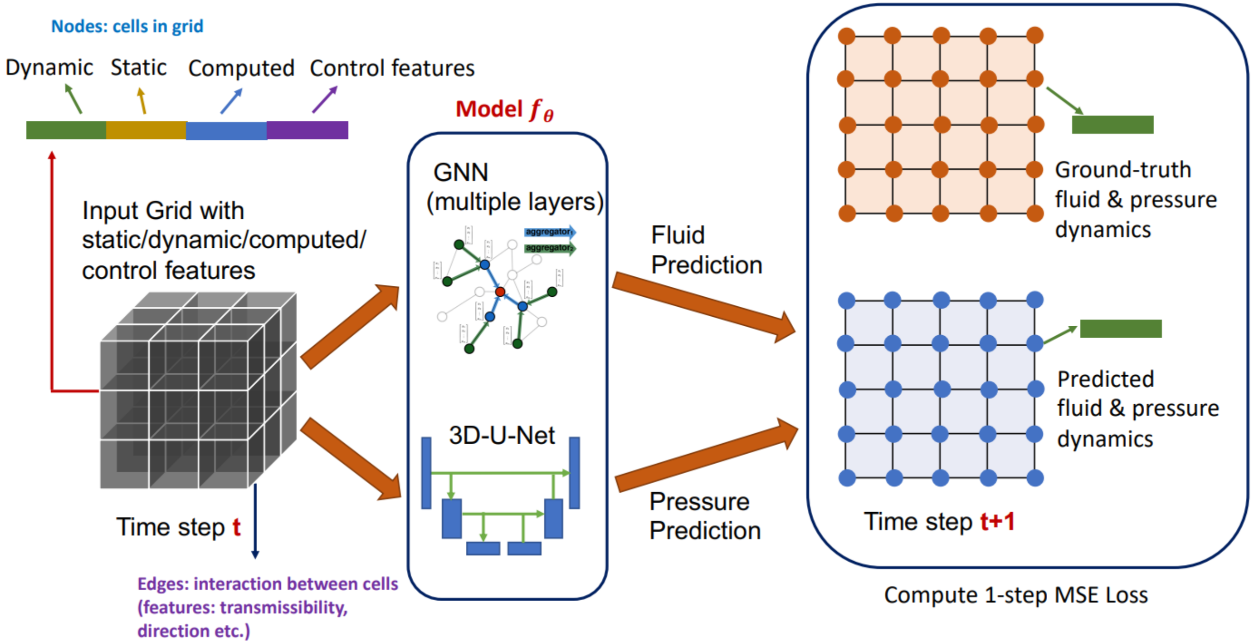
- Model
  - Use 2 models to make separate **water** and **pressure** prediction
  - Benefit: combine the advantages of both models
- As for pressure prediction
  - Apply **UNet**[1] to model interactions at global scale
  - Unet is suitable for **pressure modeling**
  - Next step pressure could depend on the entire grid information



U-shaped encoder-decoder network architecture

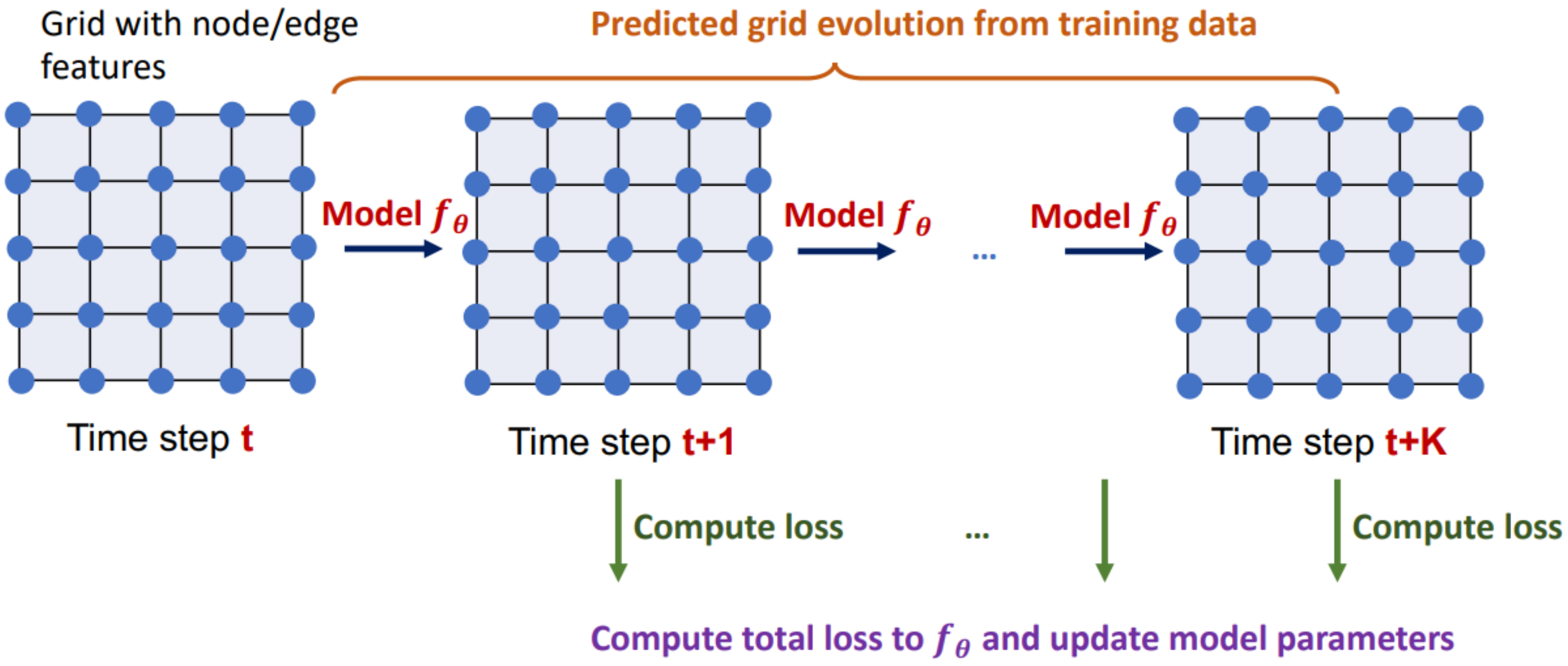
Improvement: Hybrid Model (2)

- As for **water** prediction
  - Apply GNN to model water volume
  - Utilize edge features and local interaction
- Combine these two models



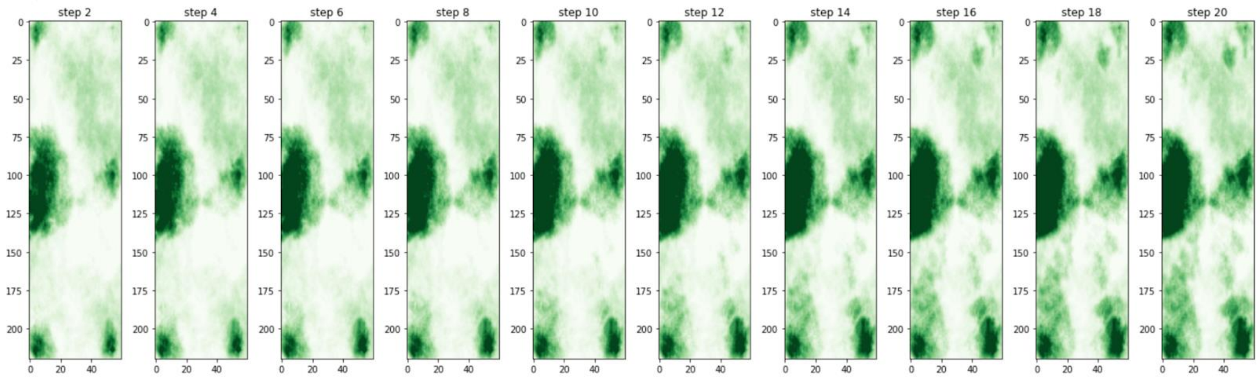
Multi-step Rollout During Training

- The loss consists of error on each time step
- During training, the gradient can pass through the full rollout

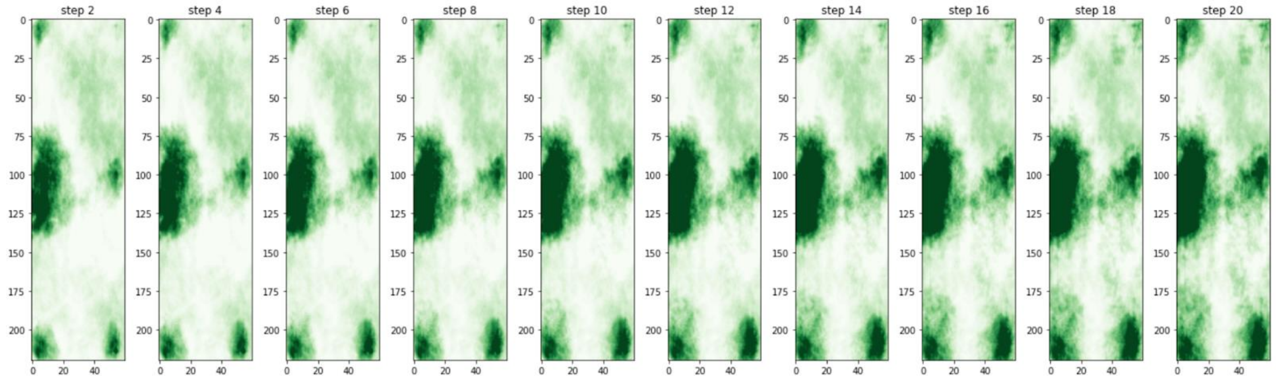


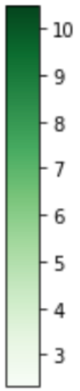
Reservoir Simulation over 20 steps (20 months)

- Groundtruth rollout (20 steps)



- Simulation results (20 steps)





barrel

Summary

- What is physical simulation?
- GNN for simulation (GNS)
  - Given previous statuses of nodes, predict the next status
  - Design message passing method to aggregate the information of surrounding neighbors
- Constraint-based Graph Neural Network Simulator
  - Explicitly model the physical constraint by learnable constraint function
- Reservoir Simulation
  - Consider a large grid as a graph
  - Apply GNN to simulate the oil, water saturation