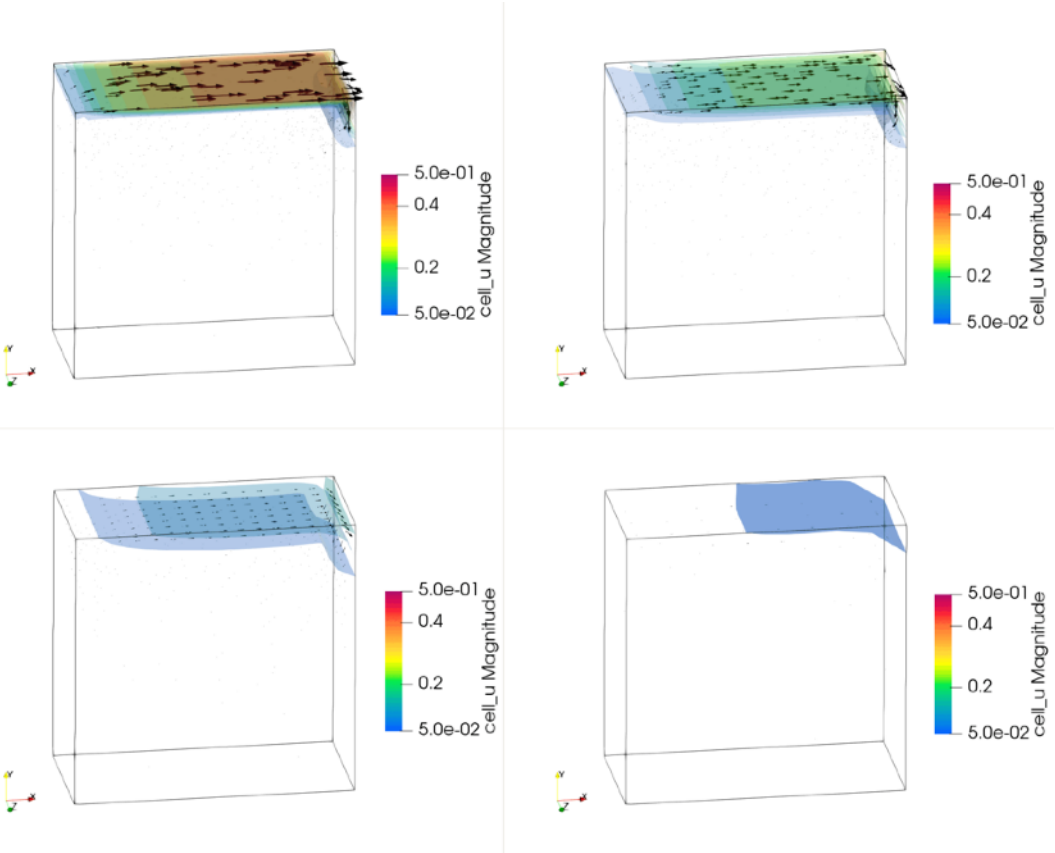
# 局所保存的グラフニューラルネットワークによるマルチスケール流体機械学習

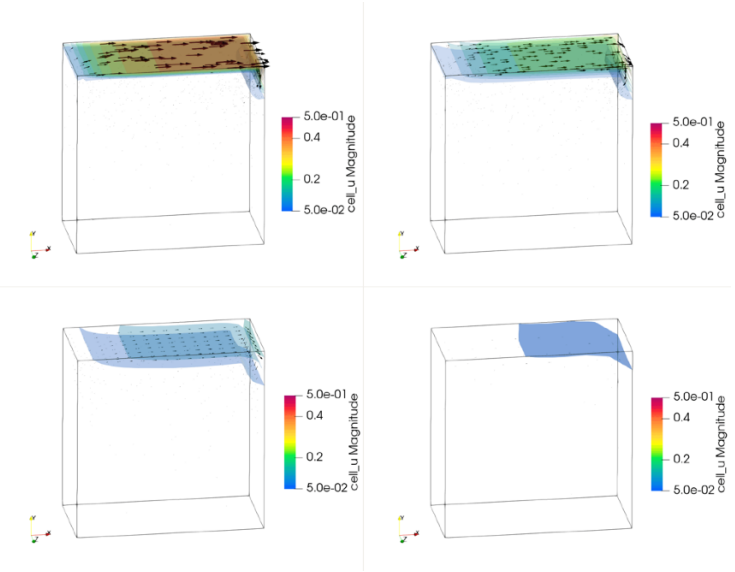



数理・計算・データに基づく流体解析の最前線
2025 年 6 月 12 日

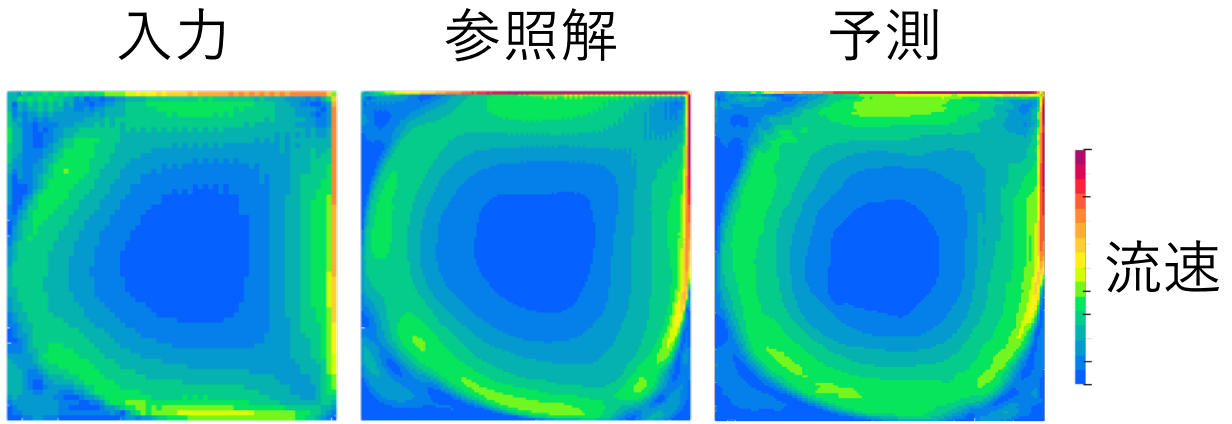
堀江 正信[12]
[1]株式会社RICOS　[2]理研AIP

# 研究の概要とまとめ

- 高 Reynolds 数 (～ $10^4$) 流れの予測に適した機械学習手法を構築した
  - 複数の解像度を協働させる非線形マルチグリッド法を
    導入することによりさまざまなスケールの相互作用を考慮
  - 任意の Reynolds 数に対して汎化性能を持つ機械学習モデルの手がかりを得た

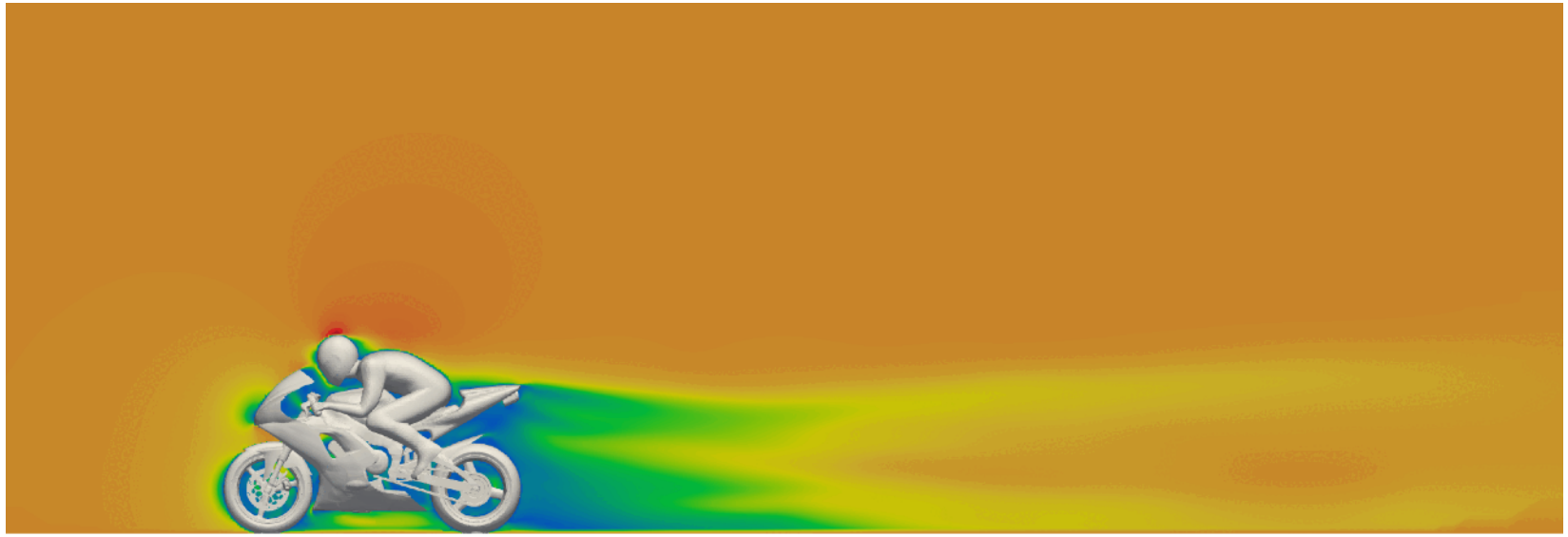



非線形
マルチグリッド法



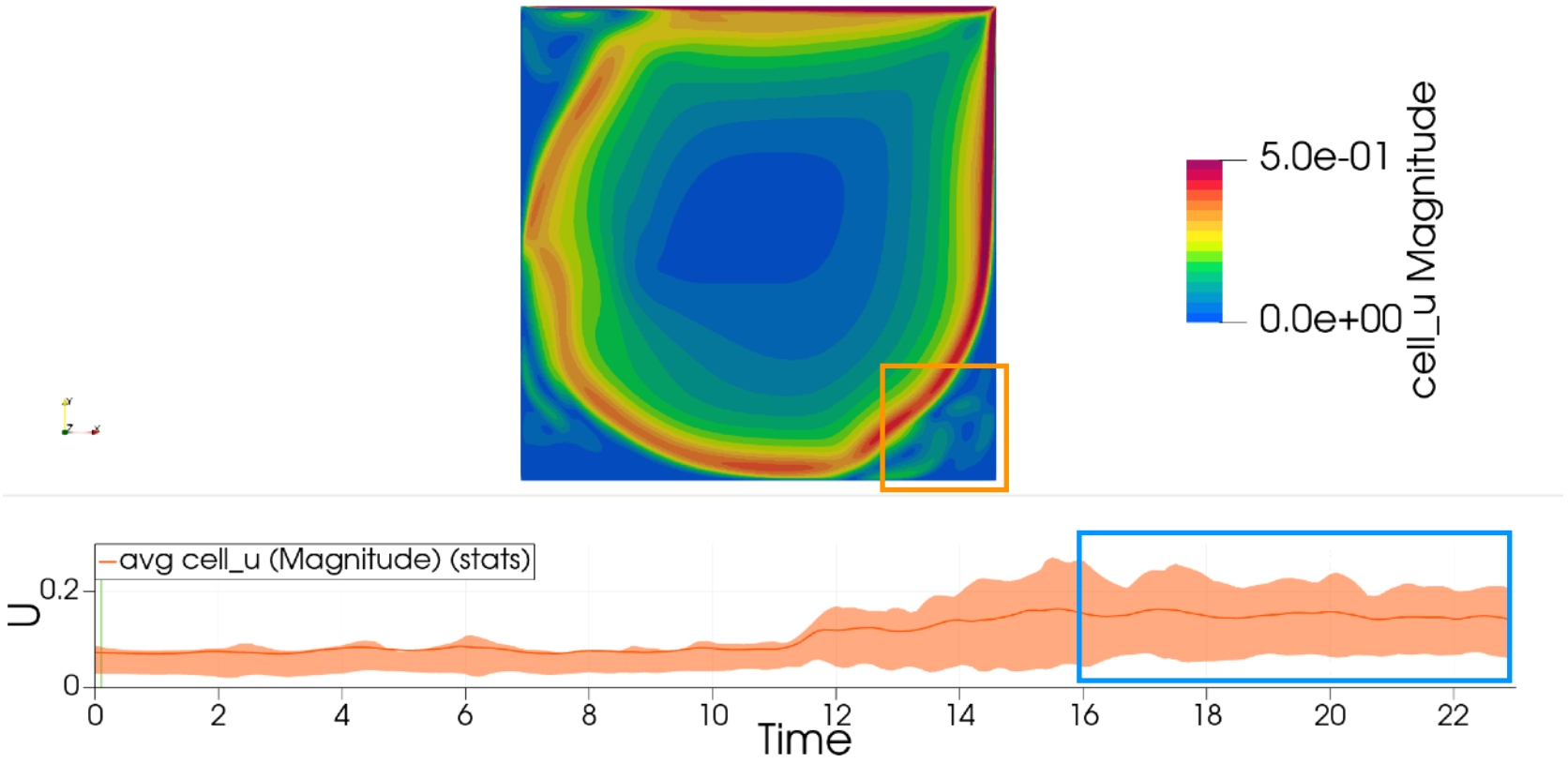
# 目的: 高 Reynolds 数流れの高速な予測

- 身の回りの流体現象の多くは乱流であり数値解析での予測は高コスト
- 機械学習による予測が期待されているが乱流は高次元・カオスであり
  学習データの作成・瞬時場の高精度な予測は原理的に困難
  - 微小の予測誤差が指数関数的に拡大していってしまう
- 統計的定常状態を予測する Reynolds 数の外挿が可能な機械学習モデルを構築したい



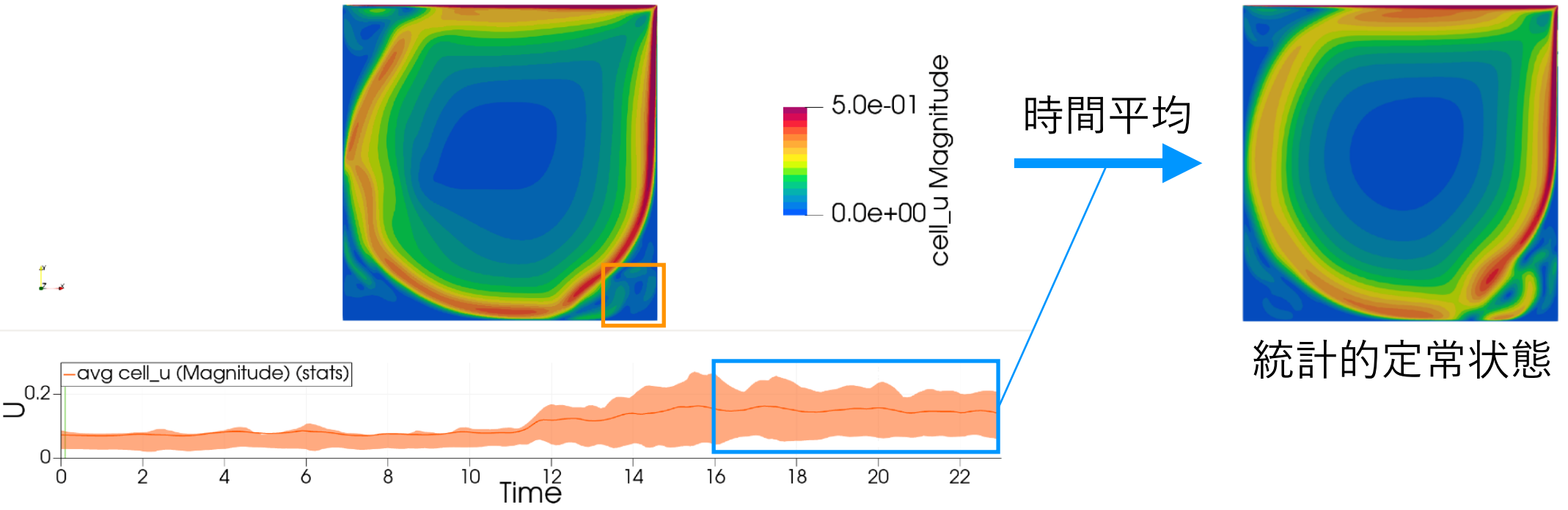
# 目的: 高 Reynolds 数流れの高速な予測

- 統計的定常状態を予測する Reynolds 数の外挿が可能な機械学習モデルを構築したい
- 統計的定常状態: 十分長い時間平均をとることによって定常とみなせる状態



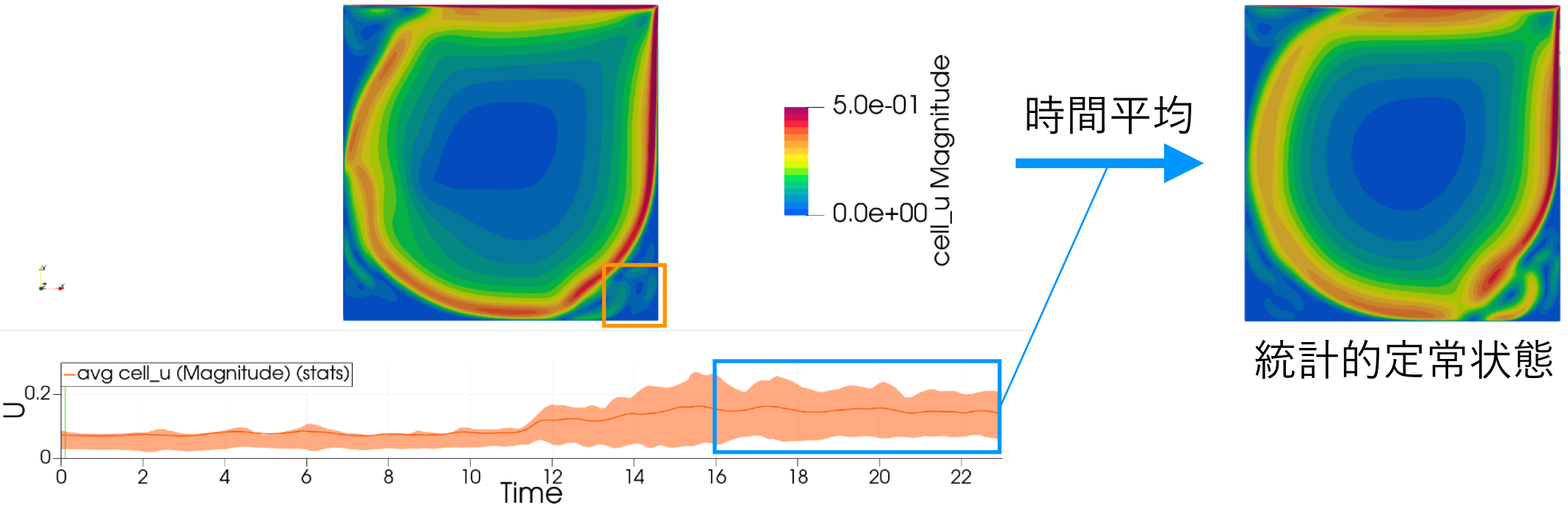
# 目的: 高 Reynolds 数流れの高速な予測

- 統計的定常状態を予測する Reynolds 数の外挿が可能な機械学習モデルを構築したい
- 統計的定常状態: 十分長い時間平均をとることによって定常とみなせる状態



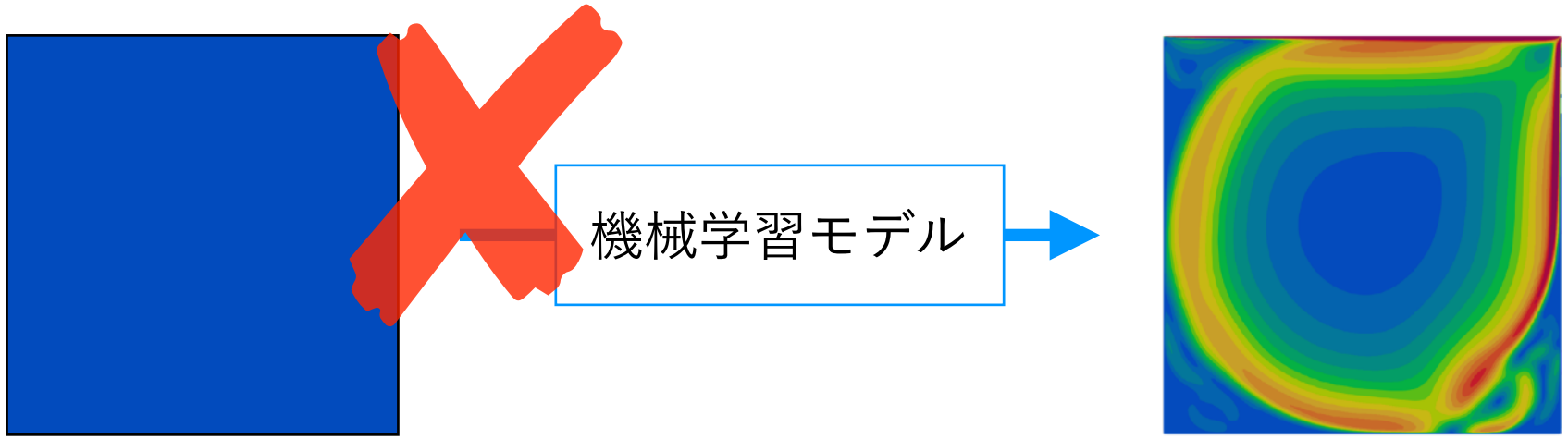
# 目的: 高 Reynolds 数流れの高速な予測

- 統計的定常状態を予測する Reynolds 数の外挿が可能な機械学習モデルを構築したい
- 統計的定常状態: 十分長い時間平均をとることによって定常とみなせる状態
- 空力性能など工学的に重要な指標は統計的定常状態において評価されるため
  統計的定常状態の高速・高精度・汎用的な予測は応用上重要



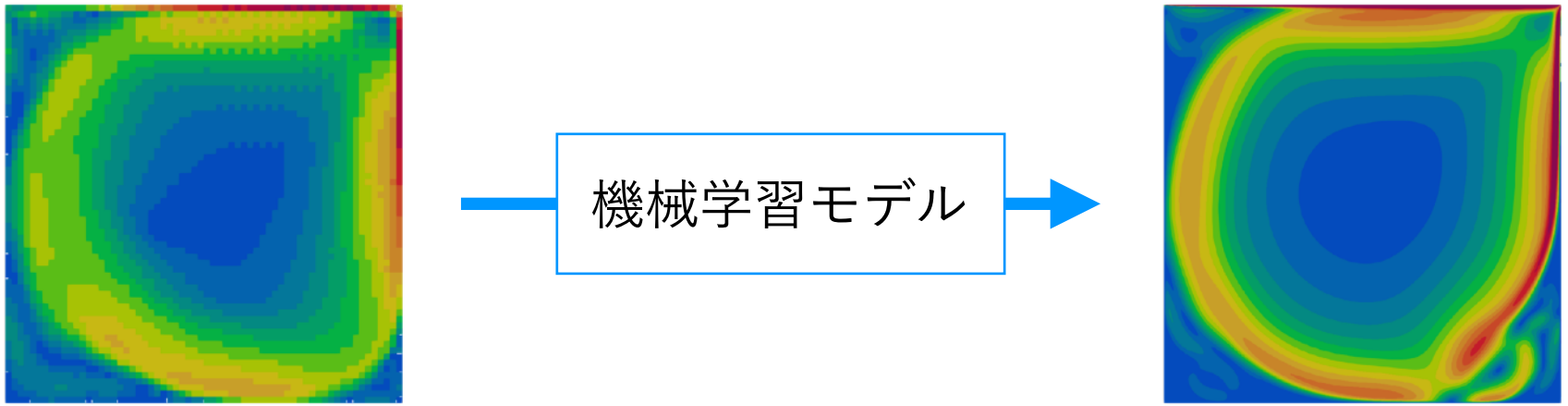
# 目的: 高 Reynolds 数流れの高速な予測

- 統計的定常状態を予測する Reynolds 数の外挿が可能な機械学習モデルを構築したい
- 通常の数値解析で用いられる一様な初期条件では汎用的な機械学習が困難
  - 形状の特徴量 (座標・壁からの距離など) を導入すると汎用性が失われやすい



# 目的: 高 Reynolds 数流れの高速な予測

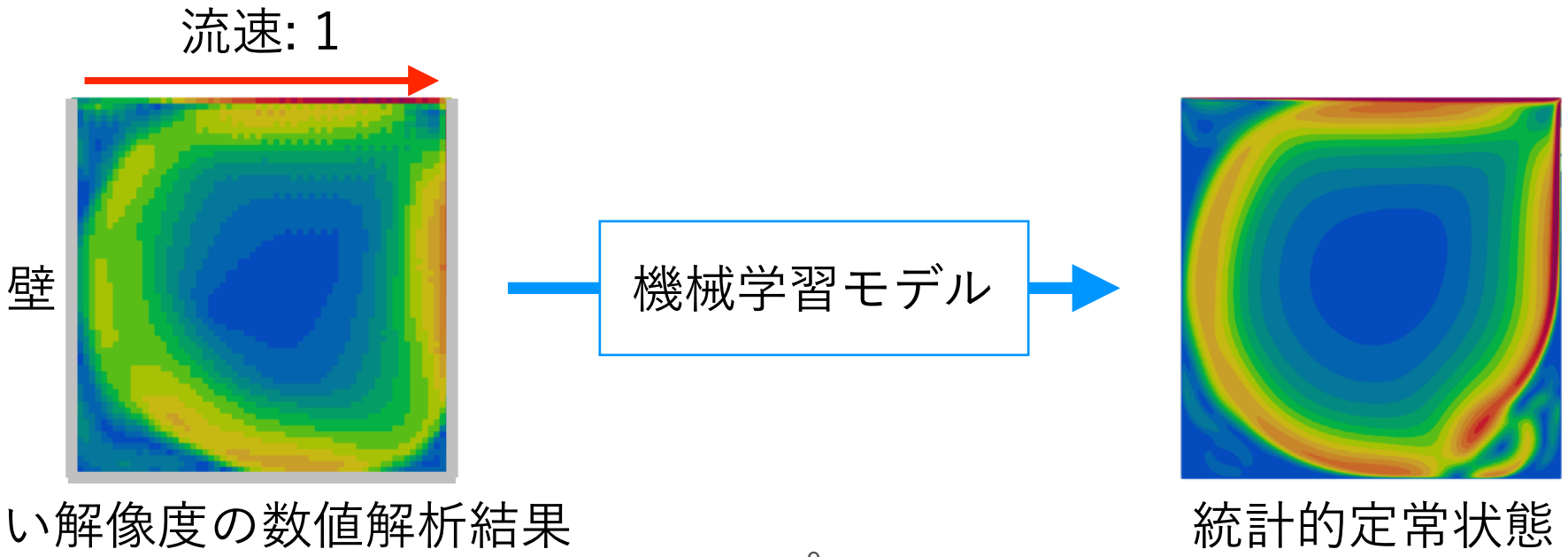
- 統計的定常状態を予測する Reynolds 数の外挿が可能な機械学習モデルを構築したい
- 通常の数値解析で用いられる一様な初期条件では汎用的な機械学習が困難
  - 形状の特徴量 (座標・壁からの距離など) を導入すると汎用性が失われやすい
- 粗い解像度の数値解析結果を入力として細かい解像度の統計的定常状態を予測する
  → 超解像シミュレーション[Yasuda & Onishi, 2025]
  - 単なるぼかし除去ではなく渦の位置などが変わるため流体現象の高度な理解が必要




粗い解像度の数値解析結果　　統計的定常状態

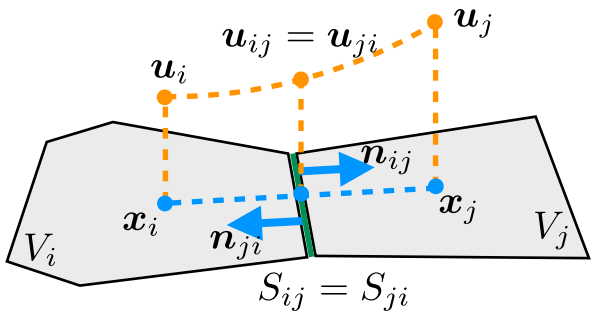
# 問題設定: 超解像シミュレーション

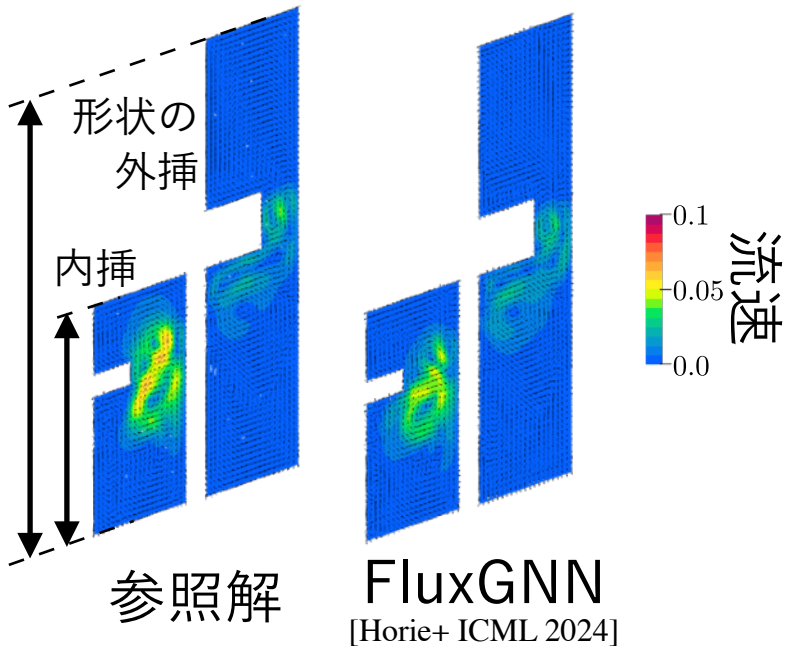
- 粗い解像度の数値解析結果を入力として細かい解像度の統計的定常状態を予測する
  → 超解像シミュレーション[Yasuda & Onishi, 2025]
  - 単なるぼかし除去ではなく渦の位置などが変わるため流体現象の高度な理解が必要
- 標準的な数値流体力学の問題設定である 2 次元キャビティ流れにおいて Reynolds 数に対する外挿性能を評価する

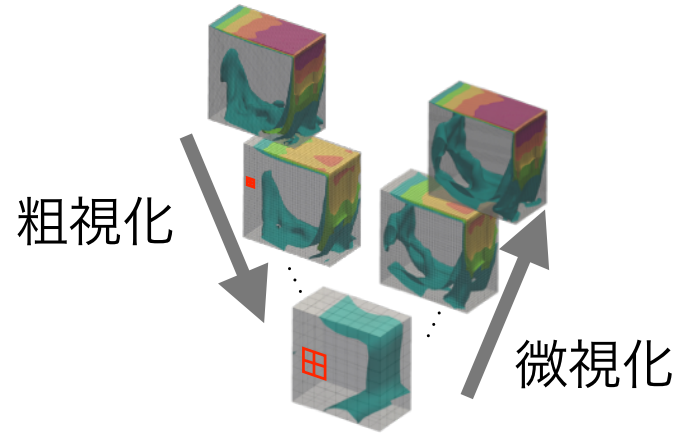


# 方針: 高 Reynolds 数流れに適した機械学習モデル

- 高 Reynolds 数流れの学習に適したモデルを適用・新たに構成する
  - FluxGNN[Horie+ ICML 2024]: 局所保存性を厳密に満たし、形状の外挿が可能な機械学習モデル
  - 非線形マルチグリッド法: さまざまなスケールの現象を考慮できる数値解析手法

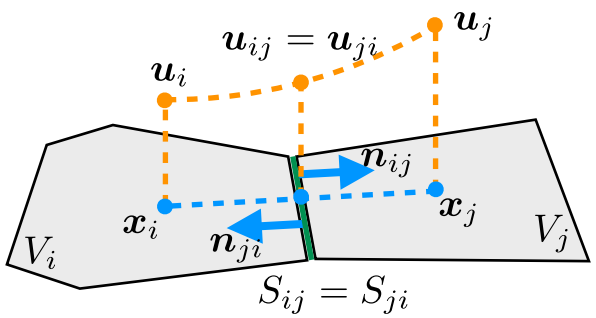


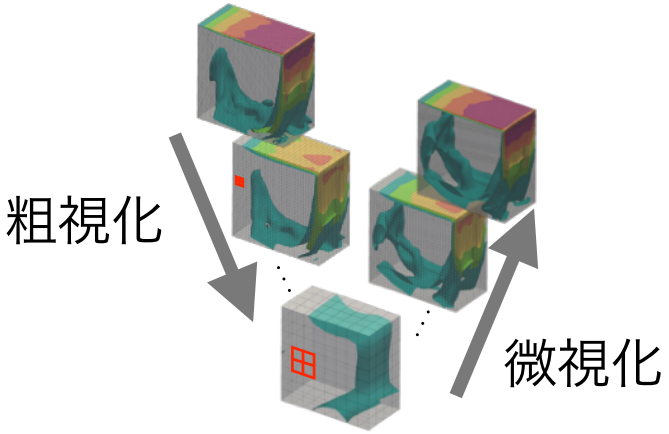



# 目次

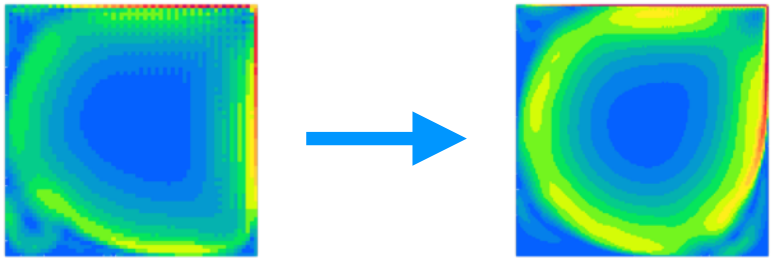
FluxGNN:
有限体積法ベースの
汎用的機械学習モデル




非線形マルチグリッド:
複数の解像度を協働させた
シミュレーション手法

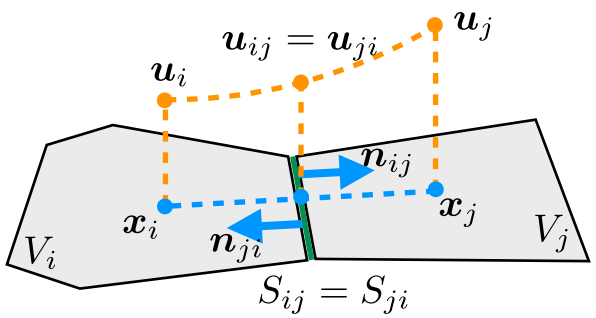



数値実験:
超解像シミュレーションと
Reynolds 数への汎化性能

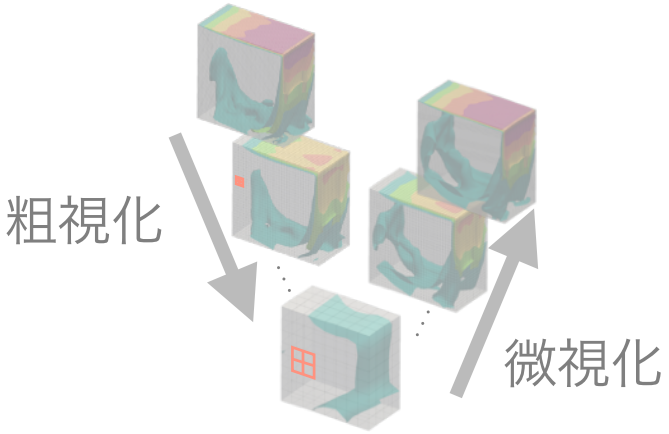


# 目次

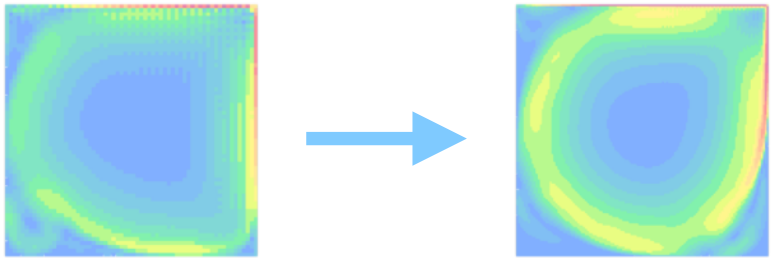
FluxGNN:
有限体積法ベースの
汎用的機械学習モデル




非線形マルチグリッド:
複数の解像度を協働させた
シミュレーション手法




数値実験:
超解像シミュレーションと
Reynolds 数への汎化性能



# FluxGNN: 有限体積法ベースの汎用的機械学習モデル

- FVM は保存形式 (移流拡散や NS 方程式の一般化) を離散化することにより得られる

$$\frac{\partial}{\partial t}\int_{V_P} dV\boldsymbol{u} = -\oint_{\partial V_P} dS\boldsymbol{n}\cdot\boldsymbol{F}(\boldsymbol{u}) \quad \text{保存形式}$$

# FluxGNN: 有限体積法ベースの汎用的機械学習モデル

- FVM は保存形式 (移流拡散や NS 方程式の一般化) を離散化することにより得られる

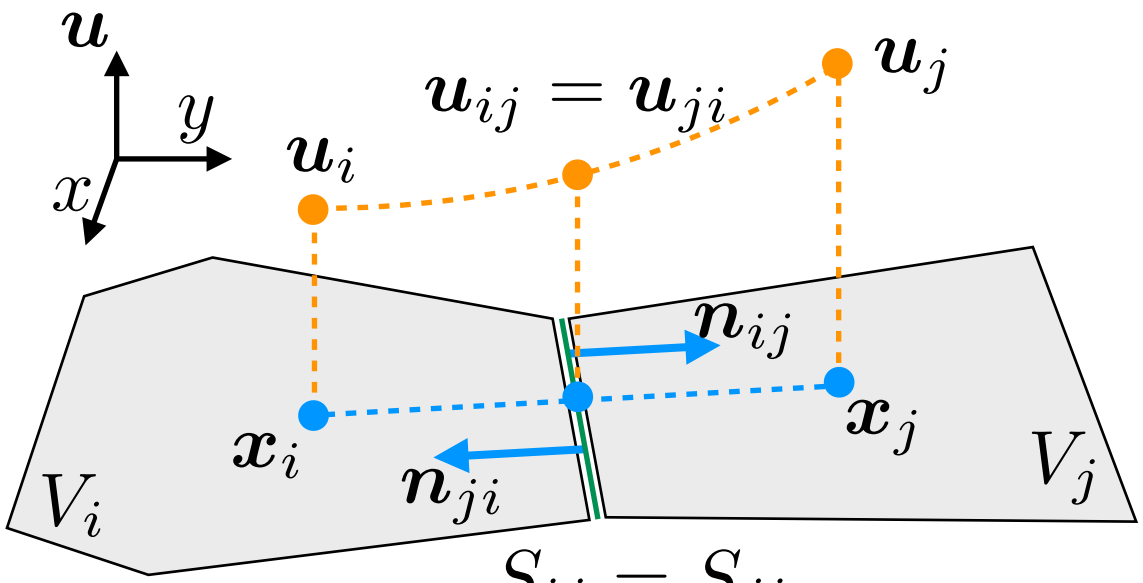
$$\frac{\partial}{\partial t}\int_{V_P} dV\boldsymbol{u} = -\oint_{\partial V_P} dS\boldsymbol{n}\cdot\boldsymbol{F}(\boldsymbol{u})$$ 保存形式

Navier–Stokes 方程式の場合:

$$\boldsymbol{F}_{\mathrm{NS}}(\boldsymbol{u}) = \boldsymbol{u}\otimes\boldsymbol{u} - \nu\nabla\otimes\boldsymbol{u} + p\boldsymbol{I}$$

# FluxGNN: 有限体積法ベースの汎用的機械学習モデル

- FVM は保存形式 (移流拡散や NS 方程式の一般化) を離散化することにより得られる




$$S_{ij} = S_{ji}$$

セル $i$ とセル $j$ を
隔てる面 $(i,j)$ の面積

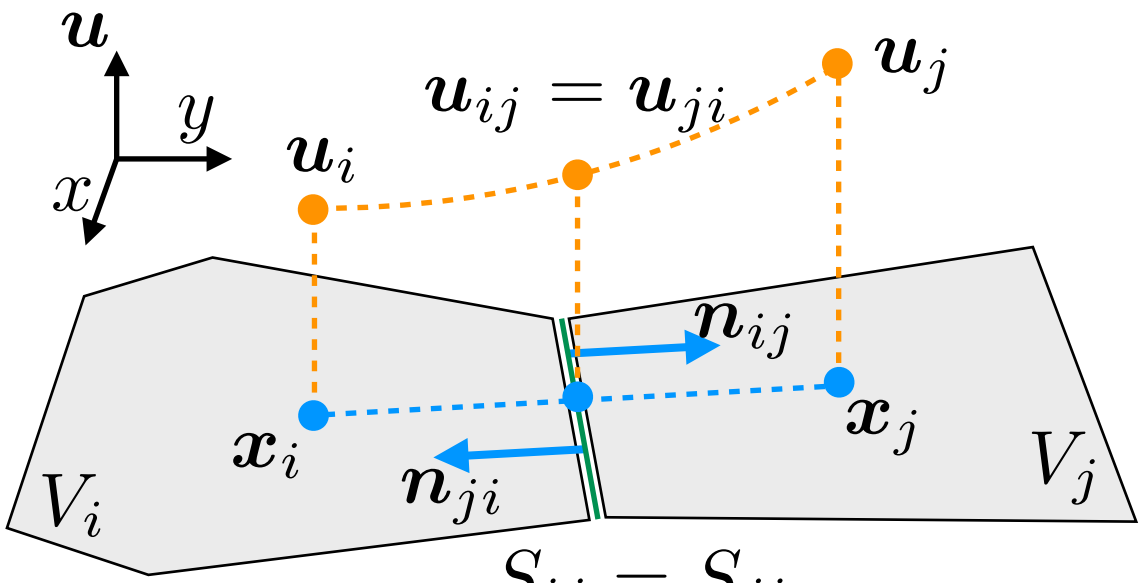
$$\frac{\partial}{\partial t}\int_{V_P} dV\boldsymbol{u} = -\oint_{\partial V_P} dS\boldsymbol{n}\cdot\boldsymbol{F}(\boldsymbol{u})$$ 保存形式

↓ セル・面中心の値で代表させて離散化

$$\frac{\partial}{\partial t}V_i\boldsymbol{u}_i = -\sum_{j\in\mathcal{N}_i} S_{ij}\boldsymbol{n}_{ij}\cdot\boldsymbol{F}(\boldsymbol{u}_{ij})$$

# FluxGNN: 有限体積法ベースの汎用的機械学習モデル

- FVM は保存形式 (移流拡散や NS 方程式の一般化) を離散化することにより得られる




$S_{ij} = S_{ji}$

セル $i$ とセル $j$ を
隔てる面 $(i,j)$ の面積

$$\frac{\partial}{\partial t}\int_{V_P} dV\boldsymbol{u} = -\oint_{\partial V_P} dS\boldsymbol{n}\cdot\boldsymbol{F}(\boldsymbol{u})$$ 保存形式

↓ セル・面中心の値で代表させて離散化

$$\frac{\partial}{\partial t}V_i\boldsymbol{u}_i = -\sum_{j\in\mathcal{N}_i} S_{ij}\boldsymbol{n}_{ij}\cdot\boldsymbol{F}(\boldsymbol{u}_{ij})$$

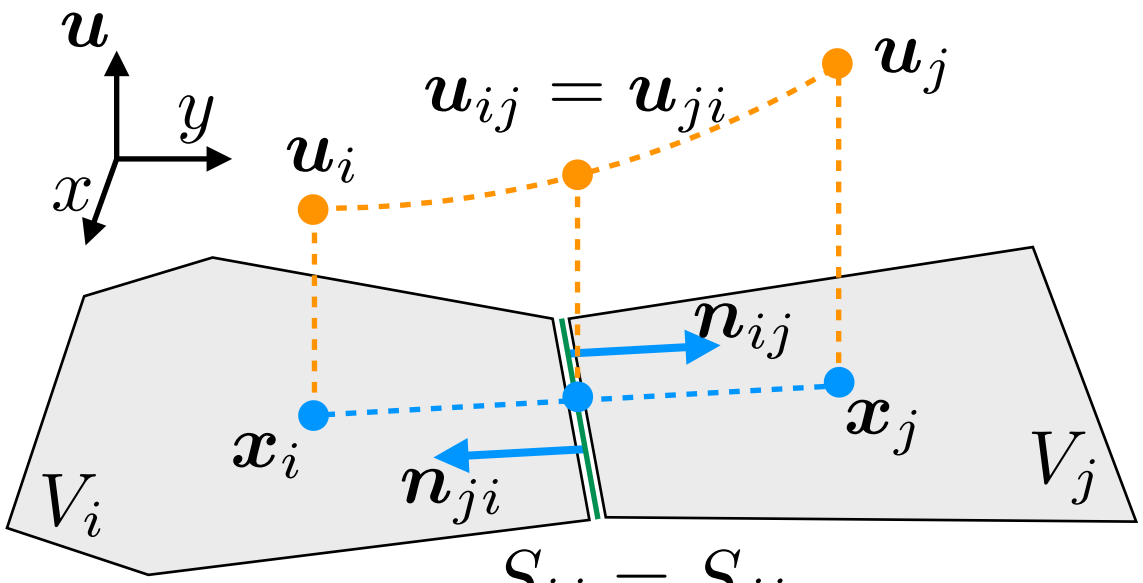
↓ 面 $(i,j)$ $(=(j,i))$ のみに着目

$$\left.\frac{\partial}{\partial t}V_i\boldsymbol{u}_i\right|_{ij} = -S_{ij}\boldsymbol{n}_{ij}\cdot\boldsymbol{F}(\boldsymbol{u}_{ij})$$

$$\left.\frac{\partial}{\partial t}V_j\boldsymbol{u}_j\right|_{ji} = -S_{ji}\boldsymbol{n}_{ji}\cdot\boldsymbol{F}(\boldsymbol{u}_{ji})$$

# FluxGNN: 有限体積法ベースの汎用的機械学習モデル

- FVM は保存形式 (移流拡散や NS 方程式の一般化) を離散化することにより得られる




$S_{ij} = S_{ji}$

セル $i$ とセル $j$ を

隔てる面 $(i,j)$ の面積

$$\frac{\partial}{\partial t} V_i \boldsymbol{u}_i = -\sum_{j \in \mathcal{N}_i} S_{ij} \boldsymbol{n}_{ij} \cdot \boldsymbol{F}(\boldsymbol{u}_{ij})$$

↓ 面 $(i,j)$ $(= (j,i))$ のみに着目

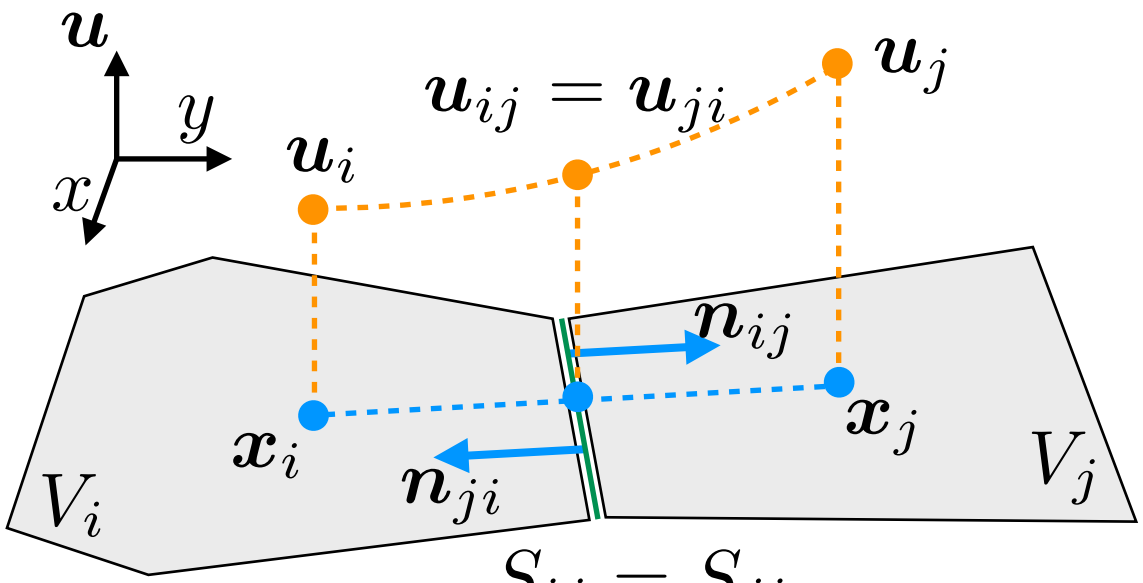
$$\left.\frac{\partial}{\partial t} V_i \boldsymbol{u}_i\right|_{ij} = -S_{ij} \boldsymbol{n}_{ij} \cdot \boldsymbol{F}(\boldsymbol{u}_{ij})$$

$$\left.\frac{\partial}{\partial t} V_j \boldsymbol{u}_j\right|_{ji} = -S_{ji} \boldsymbol{n}_{ji} \cdot \boldsymbol{F}(\boldsymbol{u}_{ji})$$

$$= S_{ij} \boldsymbol{n}_{ij} \cdot \boldsymbol{F}(\boldsymbol{u}_{ij}) = -\left.\frac{\partial}{\partial t} V_i \boldsymbol{u}_i\right|_{ij}$$

# FluxGNN: 有限体積法ベースの汎用的機械学習モデル

- FVM は保存形式 (移流拡散や NS 方程式の一般化) を離散化することにより得られる




$$S_{ij} = S_{ji}$$

セル $i$ とセル $j$ を
隔てる面 $(i,j)$ の面積

$$\frac{\partial}{\partial t} V_i \boldsymbol{u}_i = -\sum_{j \in \mathcal{N}_i} S_{ij} \boldsymbol{n}_{ij} \cdot \boldsymbol{F}(\boldsymbol{u}_{ij})$$

↓ 面 $(i,j)$ $(= (j,i))$ のみに着目

$$\left.\frac{\partial}{\partial t} V_i \boldsymbol{u}_i\right|_{ij} = -S_{ij} \boldsymbol{n}_{ij} \cdot \boldsymbol{F}(\boldsymbol{u}_{ij})$$

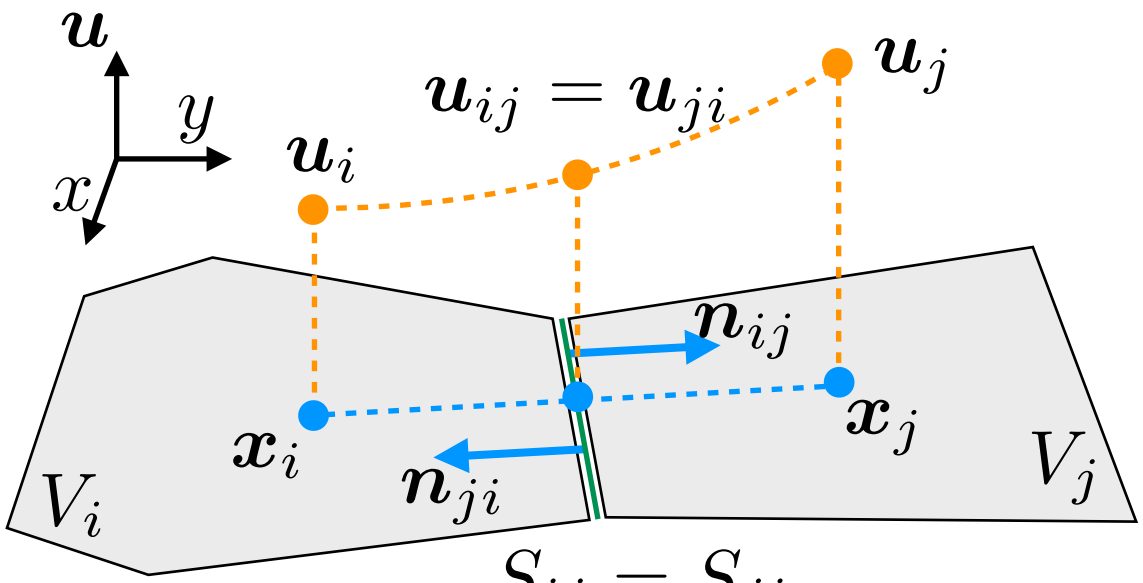
$$\left.\frac{\partial}{\partial t} V_j \boldsymbol{u}_j\right|_{ji} = -S_{ji} \boldsymbol{n}_{ji} \cdot \boldsymbol{F}(\boldsymbol{u}_{ji})$$

$$= S_{ij} \boldsymbol{n}_{ij} \cdot \boldsymbol{F}(\boldsymbol{u}_{ij}) = -\left.\frac{\partial}{\partial t} V_i \boldsymbol{u}_i\right|_{ij}$$

$$\left.\frac{\partial}{\partial t} V_i \boldsymbol{u}_i\right|_{ij} + \left.\frac{\partial}{\partial t} V_j \boldsymbol{u}_j\right|_{ji} = \boldsymbol{0}$$ → 面 $(i,j)$ を通した物理量のやり取りは保存される(= 局所保存性)

# FluxGNN: 有限体積法ベースの汎用的機械学習モデル

- FVM は保存形式 (移流拡散や NS 方程式の一般化) を離散化することにより得られる




$$S_{ij} = S_{ji}$$

セル $i$ とセル $j$ を
隔てる面 $(i, j)$ の面積

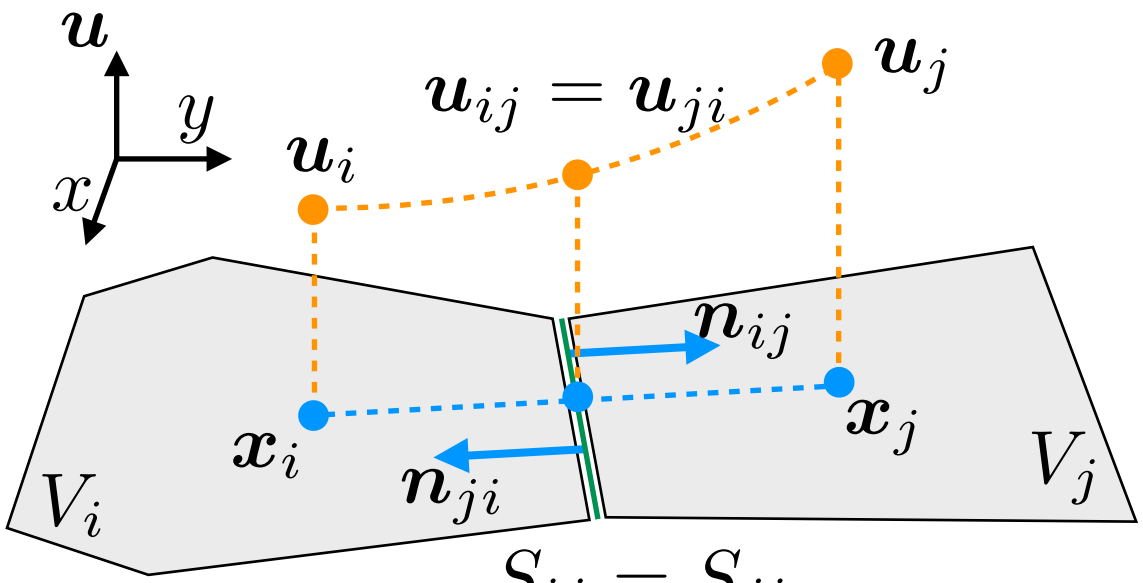
$$\frac{\partial}{\partial t} V_i \boldsymbol{u}_i = -\sum_{j \in \mathcal{N}_i} S_{ij} \boldsymbol{n}_{ij} \cdot \boldsymbol{F}(\boldsymbol{u}_{ij})$$

↓ Numerical flux $\boldsymbol{F}$ を機械学習モデル化

$$\frac{\partial}{\partial t} V_i \boldsymbol{u}_i = -\sum_{j \in \mathcal{N}_i} S_{ij} \boldsymbol{n}_{ij} \cdot \boldsymbol{F}_{\mathrm{ML}}(\boldsymbol{u}_i, \boldsymbol{u}_j)$$

# FluxGNN: 有限体積法ベースの汎用的機械学習モデル

- FVM は保存形式 (移流拡散や NS 方程式の一般化) を離散化することにより得られる




$$S_{ij} = S_{ji}$$

セル $i$ とセル $j$ を
隔てる面 $(i, j)$ の面積

$$\frac{\partial}{\partial t} V_i \boldsymbol{u}_i = -\sum_{j \in \mathcal{N}_i} S_{ij} \boldsymbol{n}_{ij} \cdot \boldsymbol{F}(\boldsymbol{u}_{ij})$$

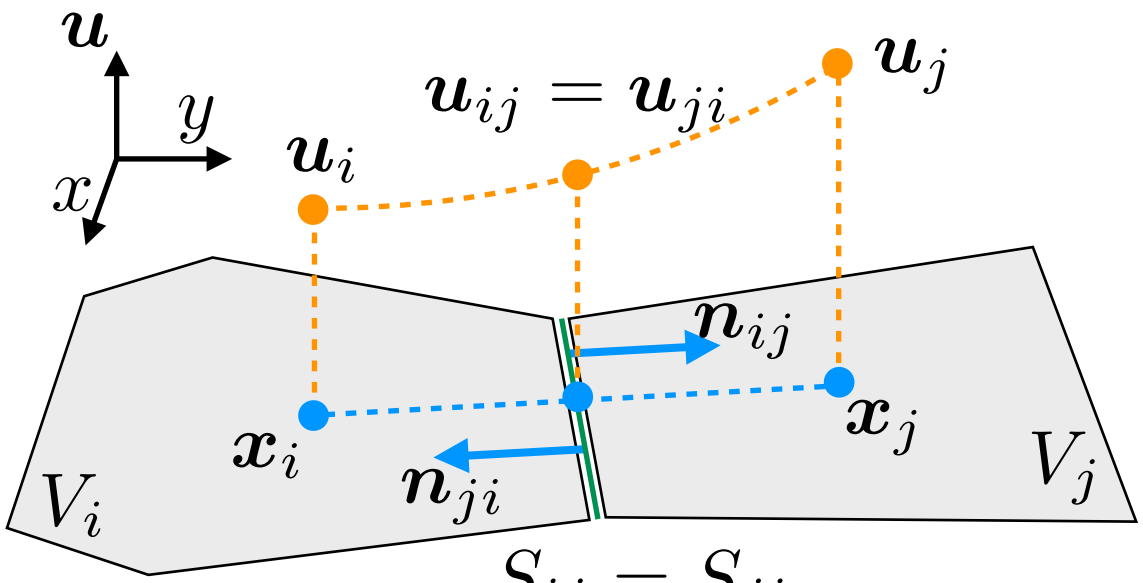
↓ Numerical flux $\boldsymbol{F}$ を機械学習モデル化

$$\frac{\partial}{\partial t} V_i \boldsymbol{u}_i = -\sum_{j \in \mathcal{N}_i} S_{ij} \boldsymbol{n}_{ij} \cdot \boldsymbol{F}_{\mathrm{ML}}(\boldsymbol{u}_i, \boldsymbol{u}_j)$$

学習可能パラメータを持つ関数

# FluxGNN: 有限体積法ベースの汎用的機械学習モデル

- FVM は保存形式 (移流拡散や NS 方程式の一般化) を離散化することにより得られる




$S_{ij} = S_{ji}$

セル $i$ とセル $j$ を
隔てる面 $(i, j)$ の面積

$$\frac{\partial}{\partial t} V_i \boldsymbol{u}_i = -\sum_{j \in \mathcal{N}_i} S_{ij} \boldsymbol{n}_{ij} \cdot \boldsymbol{F}(\boldsymbol{u}_{ij})$$

↓ Numerical flux $\boldsymbol{F}$ を機械学習モデル化

$$\frac{\partial}{\partial t} V_i \boldsymbol{u}_i = -\sum_{j \in \mathcal{N}_i} S_{ij} \boldsymbol{n}_{ij} \cdot \boldsymbol{F}_{\mathrm{ML}}(\boldsymbol{u}_i, \boldsymbol{u}_j)$$

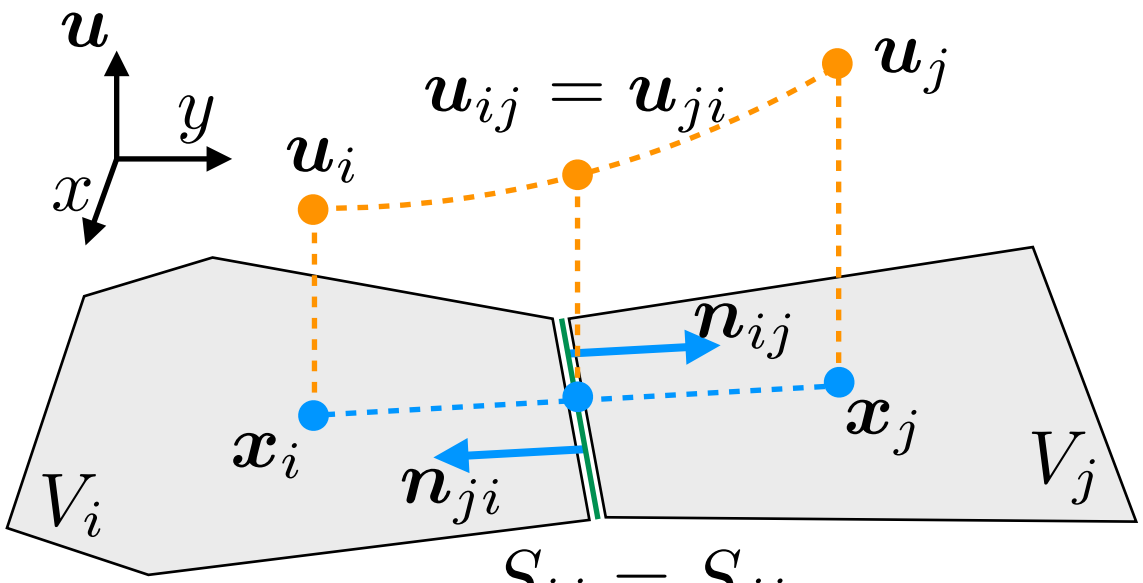
学習可能パラメータを持つ関数

- 局所保存性 $\left.\frac{\partial}{\partial t} V_i \boldsymbol{u}_i\right|_{ij} + \left.\frac{\partial}{\partial t} V_j \boldsymbol{u}_j\right|_{ji} = \boldsymbol{0}$ を満たすには

  $\forall \boldsymbol{u}, \boldsymbol{v}, \quad \boldsymbol{F}_{\mathrm{ML}}(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{F}_{\mathrm{ML}}(\boldsymbol{v}, \boldsymbol{u})$ が必要

# FluxGNN: 有限体積法ベースの汎用的機械学習モデル

- FVM は保存形式 (移流拡散や NS 方程式の一般化) を離散化することにより得られる




$$S_{ij} = S_{ji}$$

セル $i$ とセル $j$ を
隔てる面 $(i,j)$ の面積

$$\frac{\partial}{\partial t} V_i \boldsymbol{u}_i = -\sum_{j \in \mathcal{N}_i} S_{ij} \boldsymbol{n}_{ij} \cdot \boldsymbol{F}(\boldsymbol{u}_{ij})$$

↓ Numerical flux $\boldsymbol{F}$ を機械学習モデル化

$$\frac{\partial}{\partial t} V_i \boldsymbol{u}_i = -\sum_{j \in \mathcal{N}_i} S_{ij} \boldsymbol{n}_{ij} \cdot \boldsymbol{F}_{\mathrm{ML}}(\boldsymbol{u}_i, \boldsymbol{u}_j)$$

学習可能パラメータを持つ関数

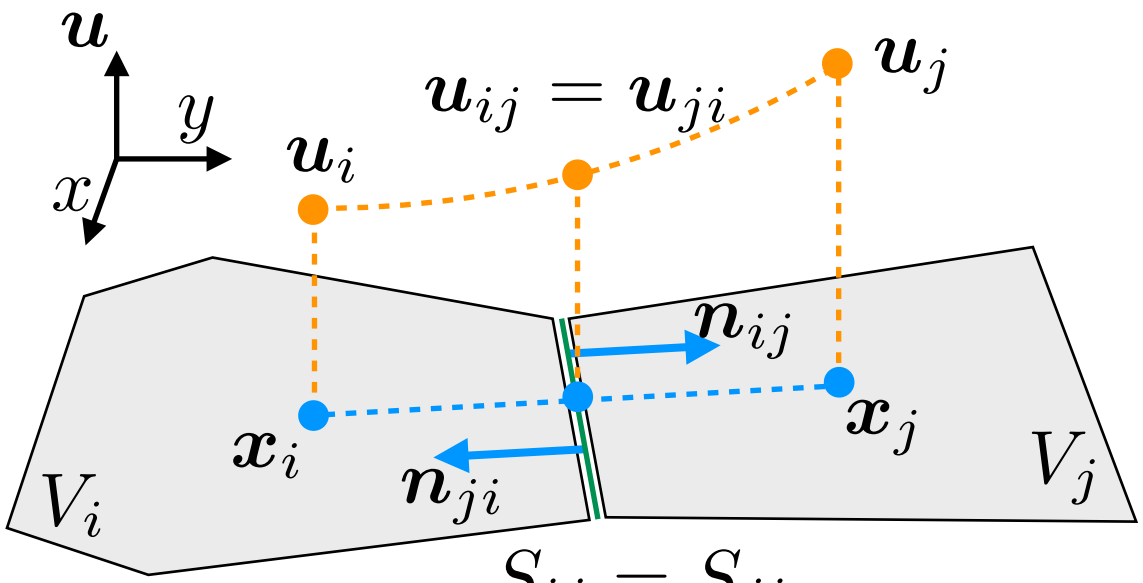
- 局所保存性 $\left.\frac{\partial}{\partial t} V_i \boldsymbol{u}_i\right|_{ij} + \left.\frac{\partial}{\partial t} V_j \boldsymbol{u}_j\right|_{ji} = \boldsymbol{0}$ を満たすには

  $\forall \boldsymbol{u}, \boldsymbol{v}, \quad \boldsymbol{F}_{\mathrm{ML}}(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{F}_{\mathrm{ML}}(\boldsymbol{v}, \boldsymbol{u})$ が必要

- 置換不変な機械学習モデルの必要十分条件である Deep Set[Zaheer+ 2017] を使用
- さらに $\boldsymbol{F}_{\mathrm{ML}}$ をテンソルの座標変換則に従うよう構成 (→ 群同変 NN)

# FluxGNN: 有限体積法ベースの汎用的機械学習モデル

- FVM は保存形式 (移流拡散や NS 方程式の一般化) を離散化することにより得られる




$S_{ij} = S_{ji}$

セル $i$ とセル $j$ を
隔てる面 $(i, j)$ の面積

$$\frac{\partial}{\partial t} V_i \boldsymbol{u}_i = -\sum_{j \in \mathcal{N}_i} S_{ij} \boldsymbol{n}_{ij} \cdot \boldsymbol{F}(\boldsymbol{u}_{ij})$$

↓ Numerical flux $\boldsymbol{F}$ を機械学習モデル化

$$\frac{\partial}{\partial t} V_i \boldsymbol{u}_i = -\sum_{j \in \mathcal{N}_i} S_{ij} \boldsymbol{n}_{ij} \cdot \boldsymbol{F}_{\mathrm{ML}}(\boldsymbol{u}_i, \boldsymbol{u}_j)$$

学習可能パラメータを持つ関数

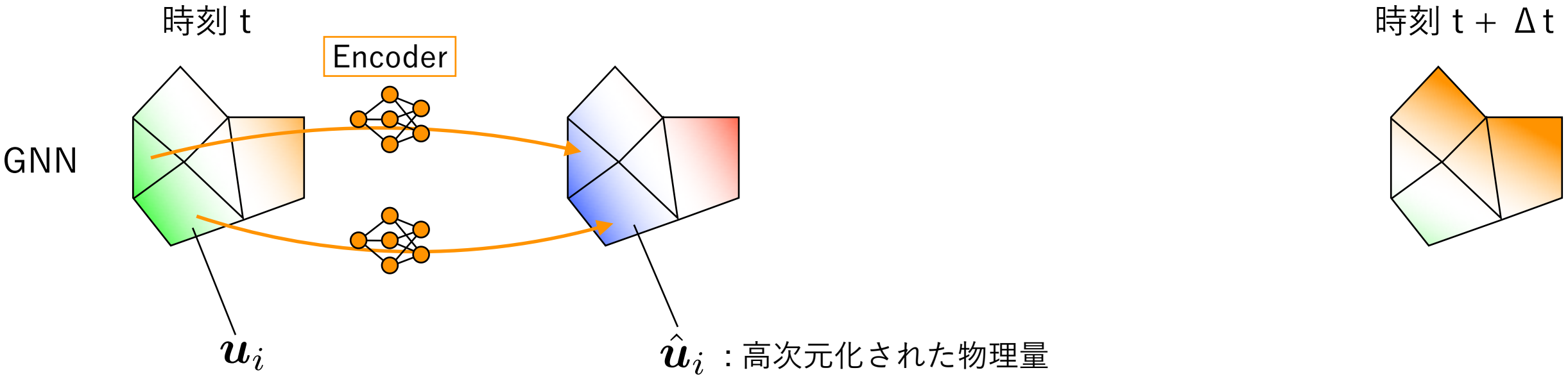
- 提案手法は微分・補間スキームの
  非線形一般化とみなせる
  - Cf. Godunov の定理
    「**線形な**単調スキームは高々 1 次精度」
  - TODO: 収束次数の解析

# FluxGNN: グラフニューラルネットワークとの類似性

GNN

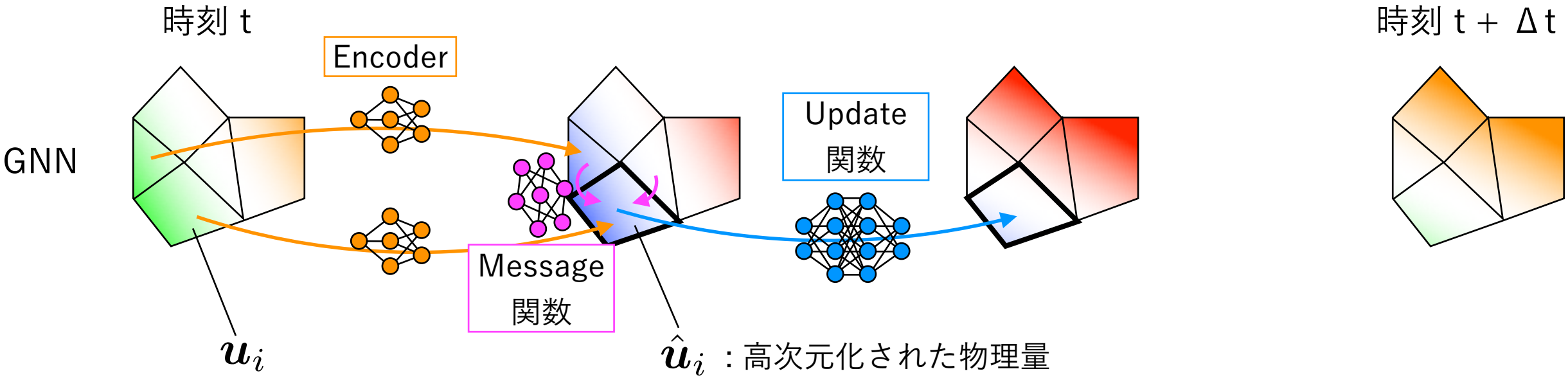

- グラフニューラルネットワーク (GNN) は任意のグラフ (メッシュ) に適用可能

# FluxGNN: グラフニューラルネットワークとの類似性



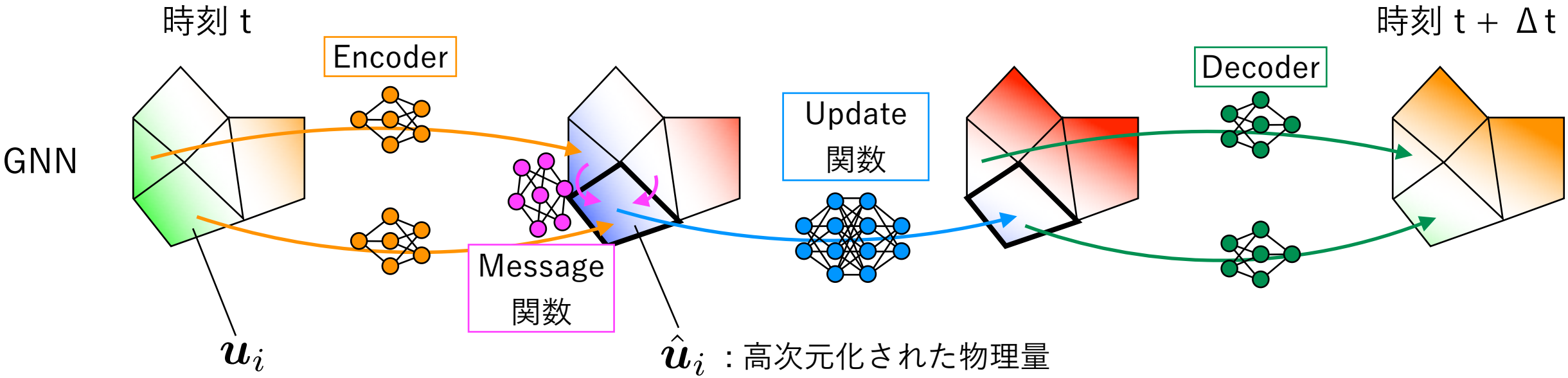

- グラフニューラルネットワーク（GNN）は任意のグラフ（メッシュ）に適用可能

# FluxGNN: グラフニューラルネットワークとの類似性



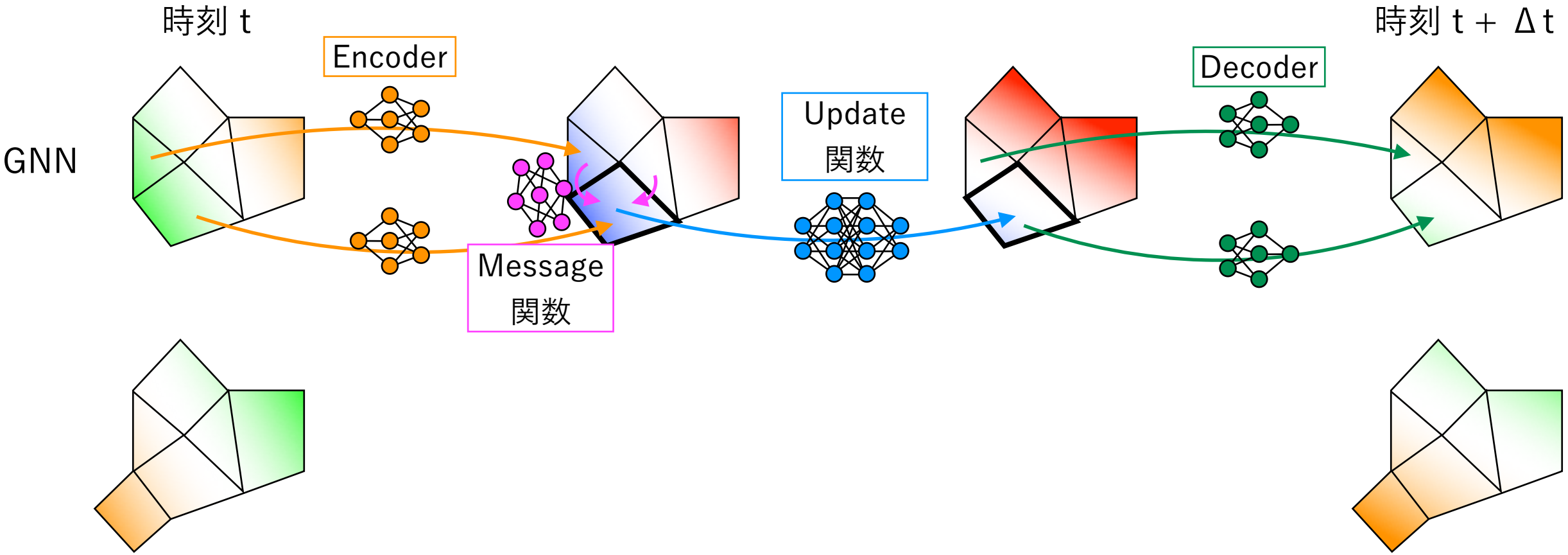

- グラフニューラルネットワーク（GNN）は任意のグラフ（メッシュ）に適用可能

# FluxGNN: グラフニューラルネットワークとの類似性



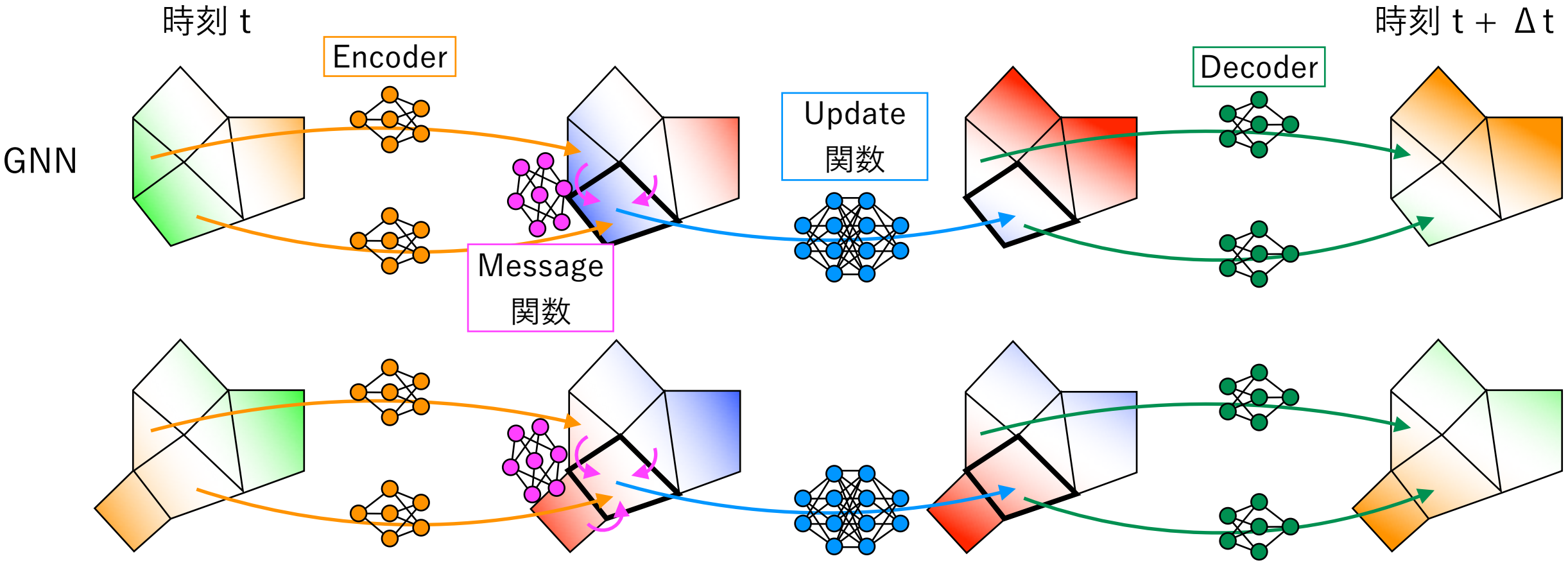

- グラフニューラルネットワーク（GNN）は任意のグラフ（メッシュ）に適用可能

# FluxGNN: グラフニューラルネットワークとの類似性



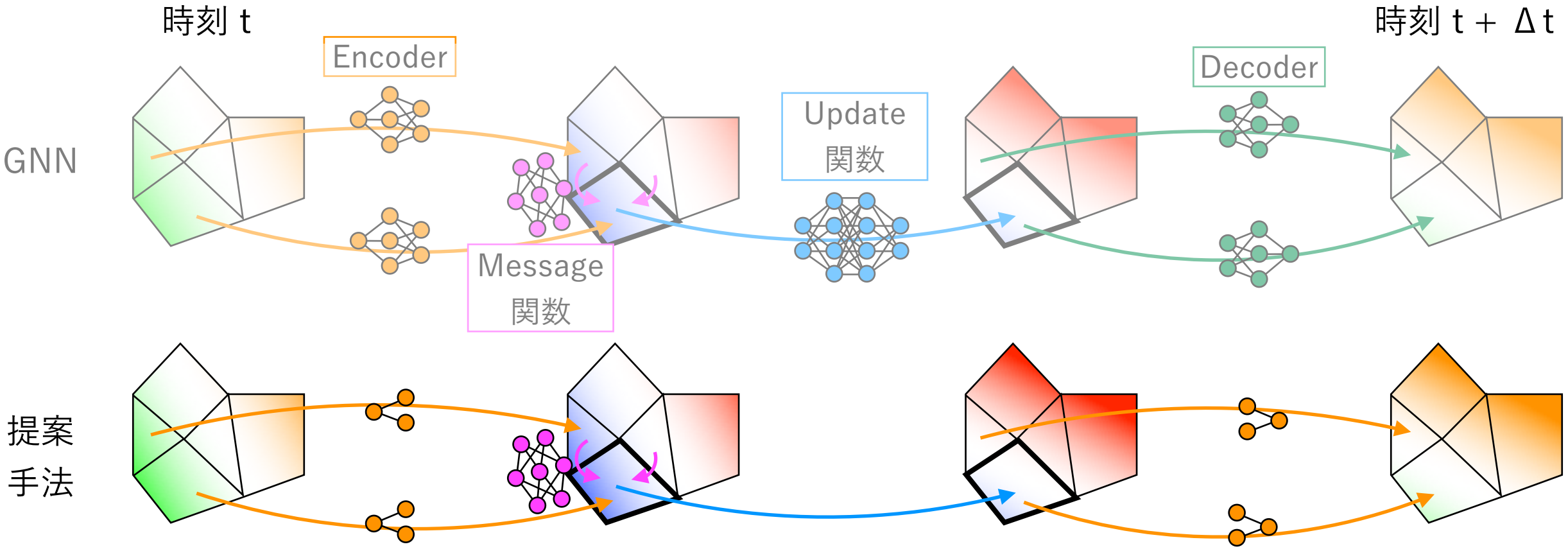

- グラフニューラルネットワーク（GNN）は任意のグラフ（メッシュ）に適用可能

# FluxGNN: グラフニューラルネットワークとの類似性



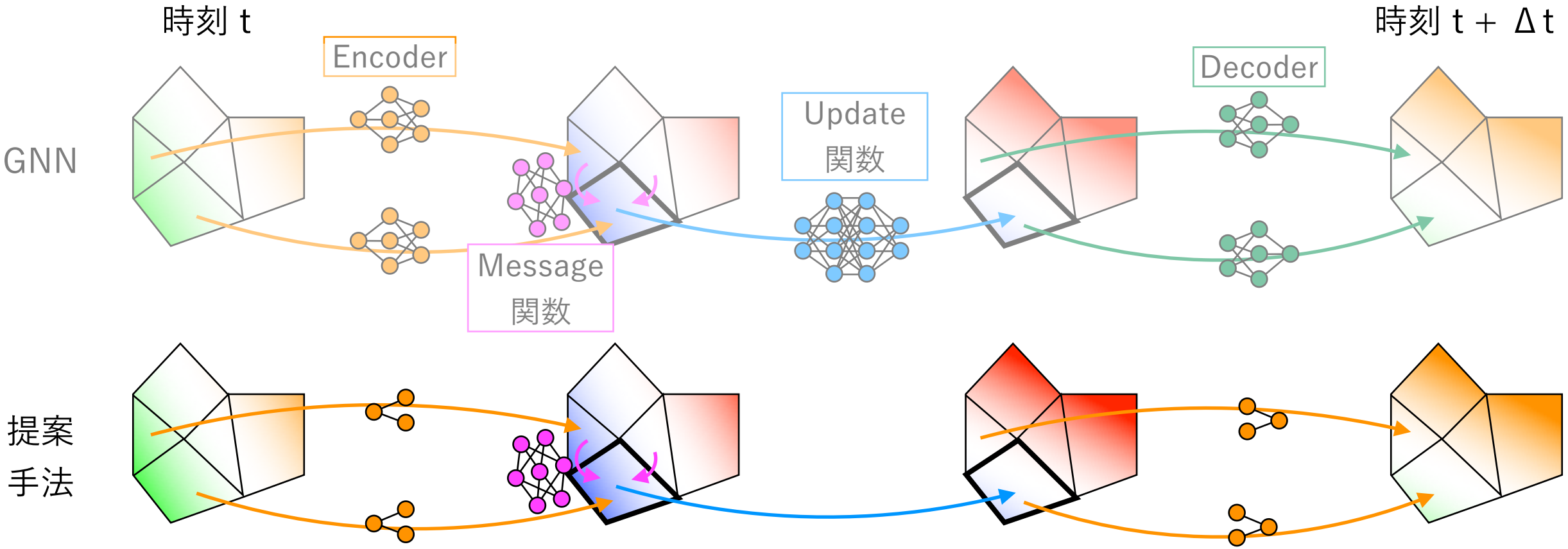

- グラフニューラルネットワーク（GNN）は任意のグラフ（メッシュ）に適用可能

# FluxGNN: グラフニューラルネットワークとの類似性



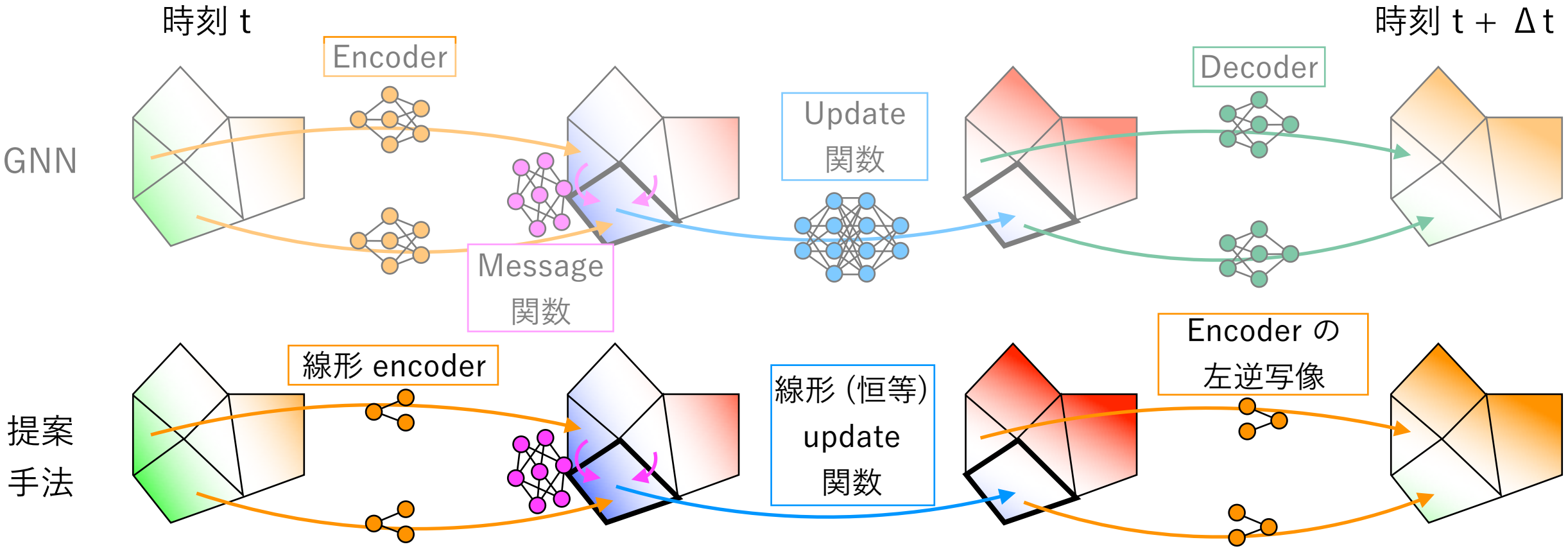

- グラフニューラルネットワーク (GNN) は任意のグラフ (メッシュ) に適用可能
- 提案手法は GNN の特殊な例であるため、任意のメッシュに適用可能

# FluxGNN: グラフニューラルネットワークとの類似性




- グラフニューラルネットワーク (GNN) は任意のグラフ (メッシュ) に適用可能
- 提案手法は GNN の特殊な例であるため、任意のメッシュに適用可能

# FluxGNN: グラフニューラルネットワークとの類似性




- グラフニューラルネットワーク (GNN) は任意のグラフ (メッシュ) に適用可能
- 提案手法は GNN の特殊な例であるため、任意のメッシュに適用可能
- 局所保存性のためのエンコーダ・デコーダ・update 関数についての制約を数学的に導出

# FluxGNN: グラフニューラルネットワークとの類似性

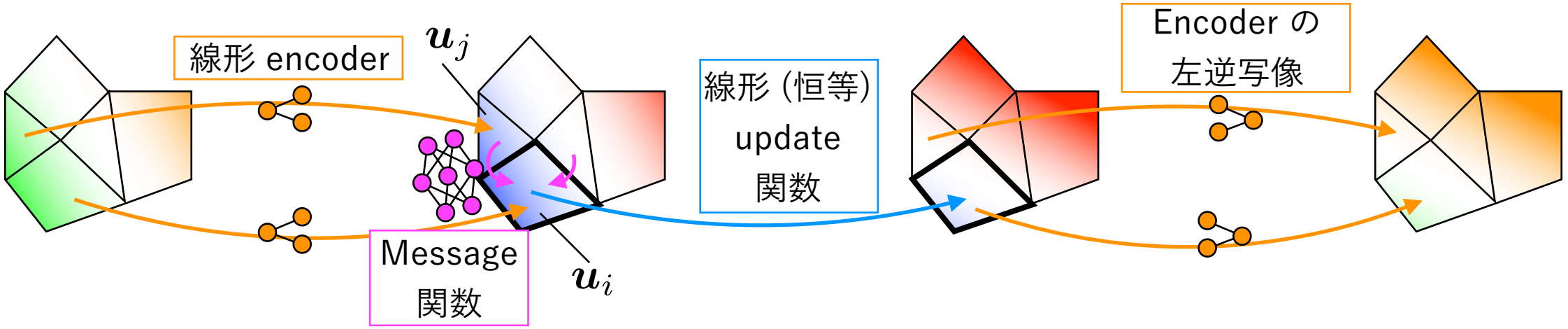
定義: GNN (message-passing neural network)

[Gilmer et al. *ICML*. 2017]

$$\boldsymbol{m}_{ij} = \boldsymbol{f}_{\text{message}}(\boldsymbol{u}_i, \boldsymbol{u}_j, \boldsymbol{u}_{ij})$$

$$\boldsymbol{u}_i^{\text{out}} = \boldsymbol{f}_{\text{update}}\left(\boldsymbol{h}_i, \sum_{j \in \text{Neighbors}(i)} \boldsymbol{m}_{ij}\right)$$

提案手法



# FluxGNN: グラフニューラルネットワークとの類似性

定義: GNN (message-passing neural network)

[Gilmer et al. *ICML*. 2017]

$$\boldsymbol{m}_{ij} = \boldsymbol{f}_{\text{message}}(\boldsymbol{u}_i, \boldsymbol{u}_j, \boldsymbol{u}_{ij})$$

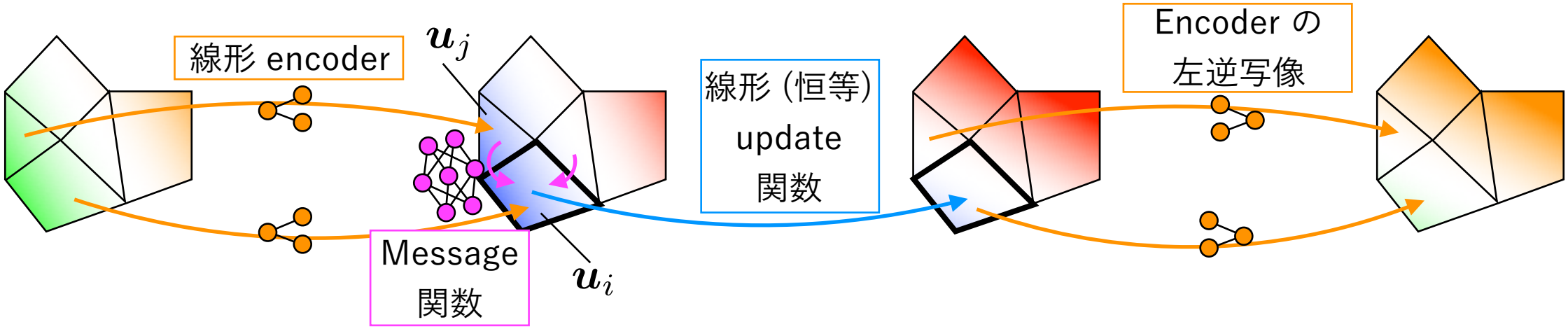
$$\boldsymbol{u}_i^{\text{out}} = \boldsymbol{f}_{\text{update}}\left(\boldsymbol{h}_i, \sum_{j \in \text{Neighbors}(i)} \boldsymbol{m}_{ij}\right)$$

定理: GNN の保存性[Horie & Mitsume. *ICML*. 2024]

GNN が保存的となる必要十分条件は以下:

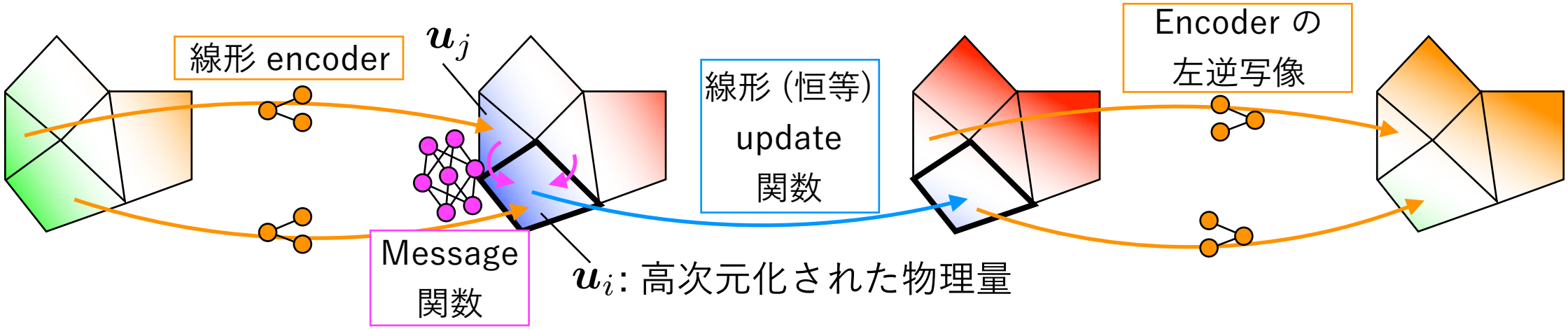
- $\forall (i,j) \in \{\text{Edge}\}, \boldsymbol{m}_{ij} = -\boldsymbol{m}_{ji}$ かつ
- Update 関数が線形 (恒等) 写像かつ
- Encoder が線形かつ
- Decoder が encoder の左逆写像

提案手法

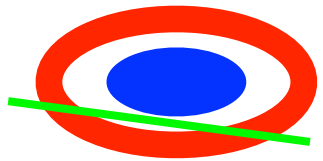


# FluxGNN: グラフニューラルネットワークとの類似性

提案手法

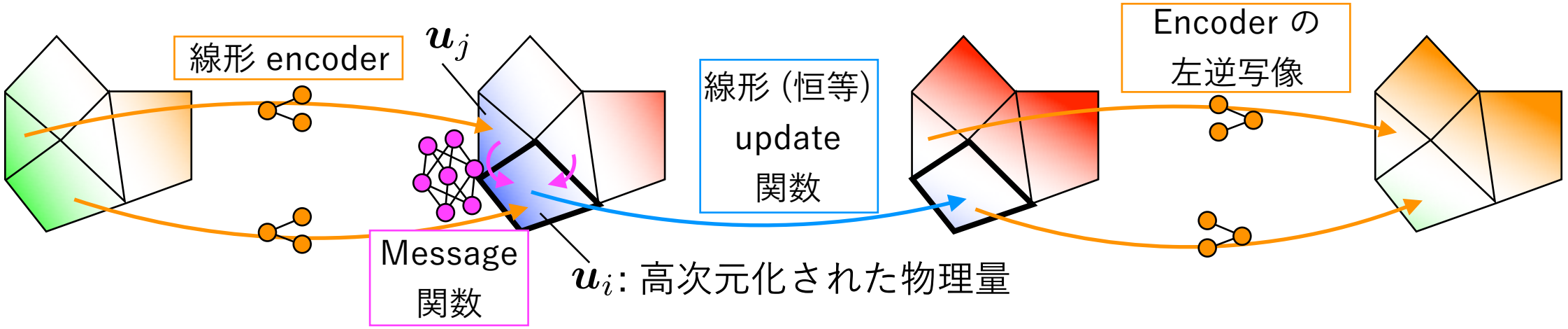



- 提案手法では他の GNN ベースの手法と同様
  入力物理量を高次元化し処理する
  - 高次元化することにより機械学習モデルの表現力が高まる (cf. universal approximation)
  - 高次の基底で物理量を展開していることに対応

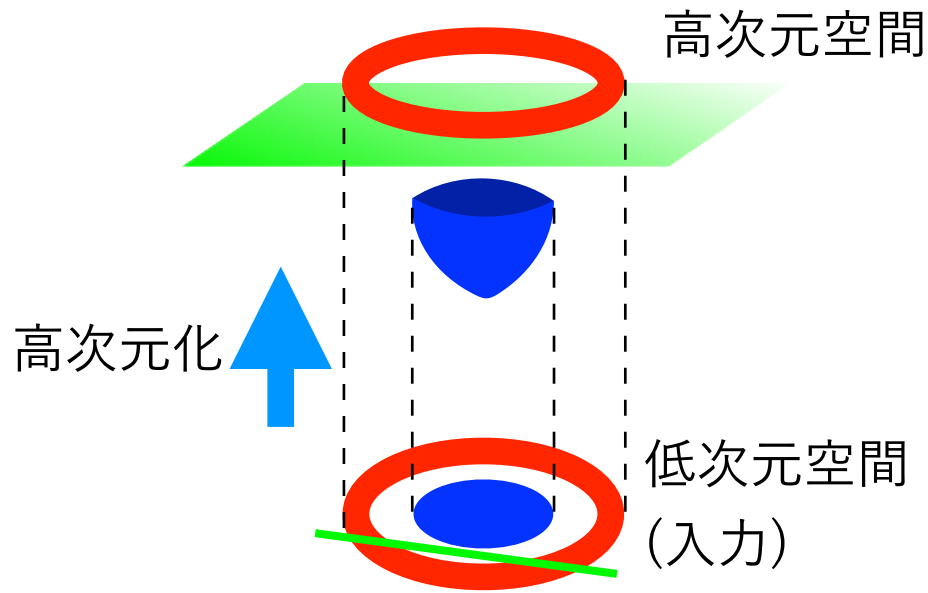


低次元空間
(入力)

# FluxGNN: グラフニューラルネットワークとの類似性

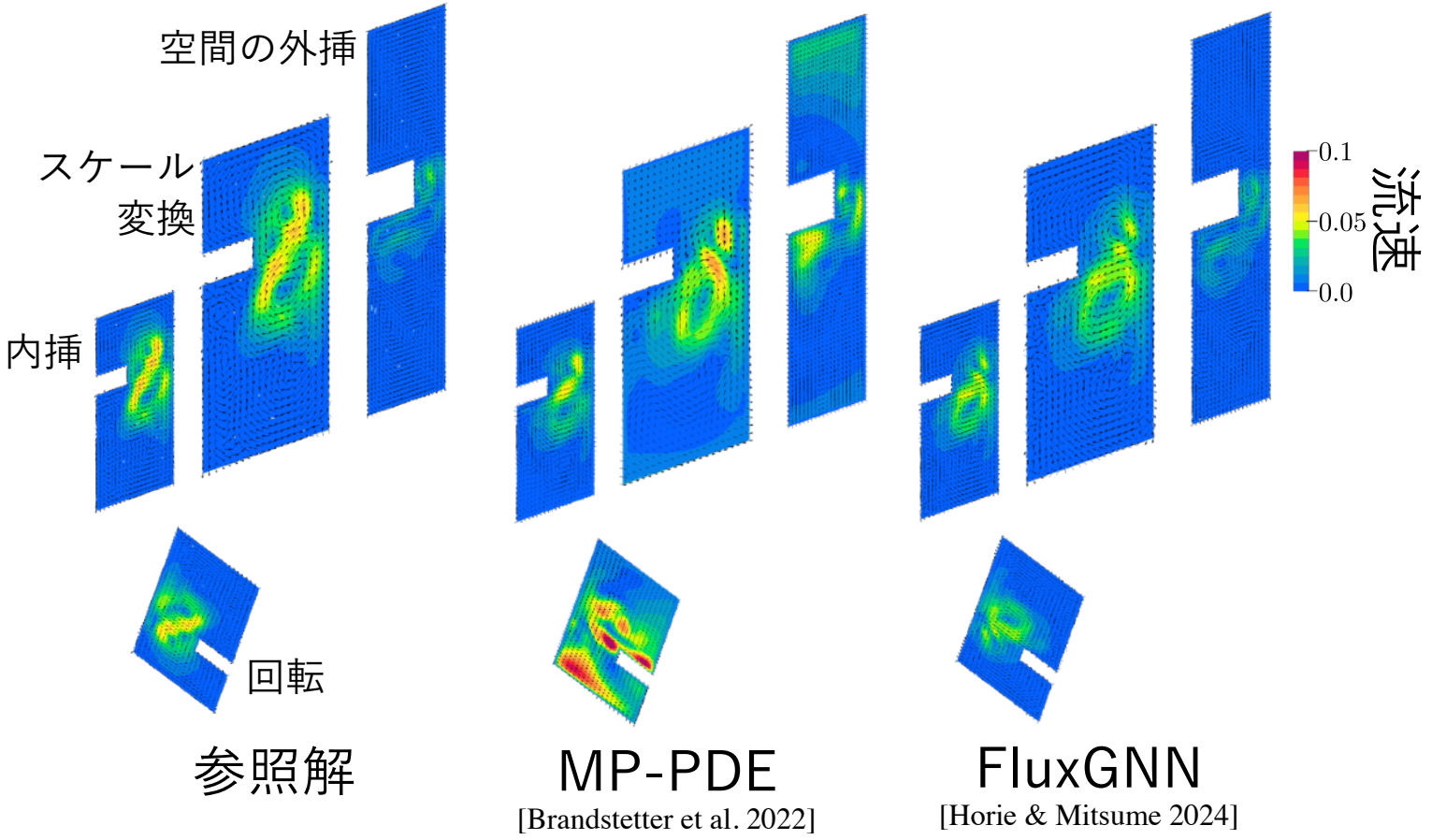



- 提案手法では他の GNN ベースの手法と同様
  入力物理量を高次元化し処理する
  - 高次元化することにより機械学習モデルの
    表現力が高まる (cf. universal approximation)
  - 高次の基底で物理量を展開していることに対応



# FluxGNN: 数値実験結果

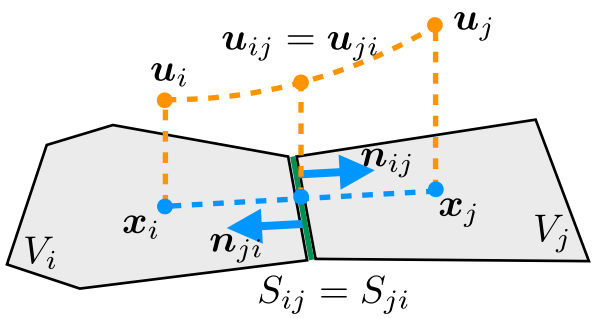
- FluxGNN[Horie & Mitsume. *ICML* 2024] は有限体積法の numerical flux を機械学習モデル化し流体現象の機械学習に特化したGNN (Graph Neural Network)



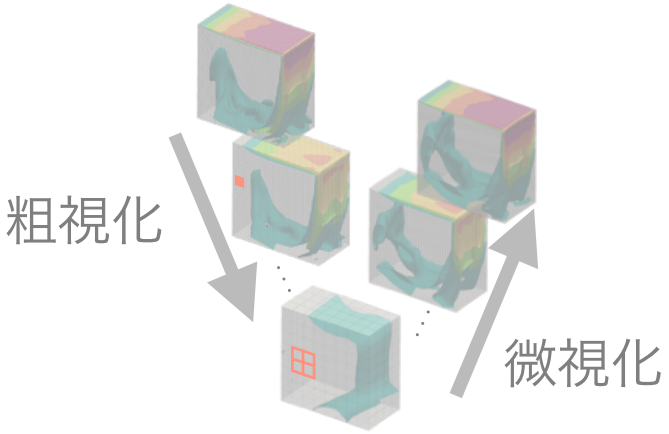

- FluxGNN は物理現象の対称性・保存則を厳密に満たす
  - 特に拡大縮小の対称性を満たす
- 形状に対する外挿性能を有する

# 目次

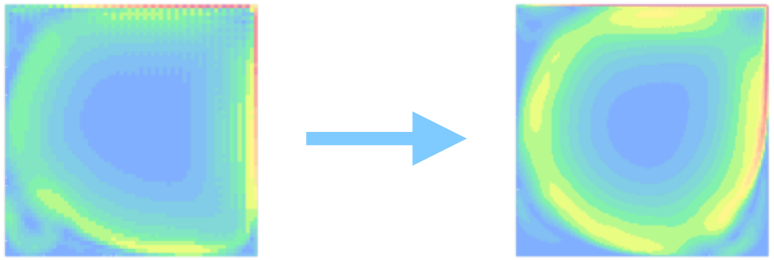
FluxGNN:
有限体積法ベースの
汎用的機械学習モデル




非線形マルチグリッド:
複数の解像度を協働させた
シミュレーション手法

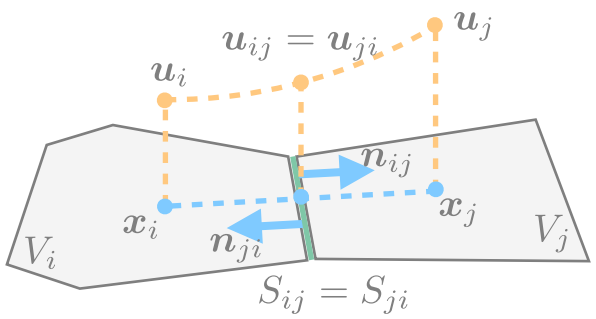



数値実験:
超解像シミュレーションと
Reynolds 数への汎化性能

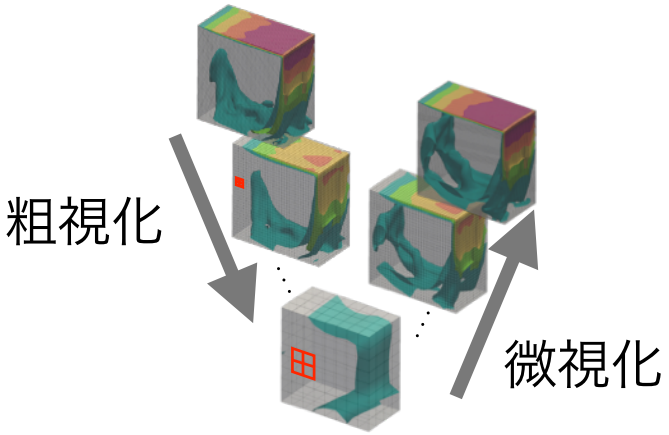


# 目次

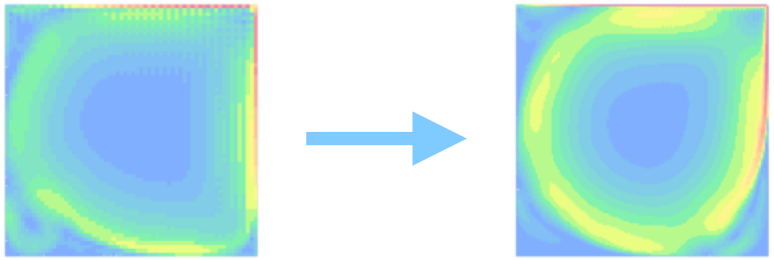
FluxGNN:
有限体積法ベースの
汎用的機械学習モデル




非線形マルチグリッド:
複数の解像度を協働させた
シミュレーション手法

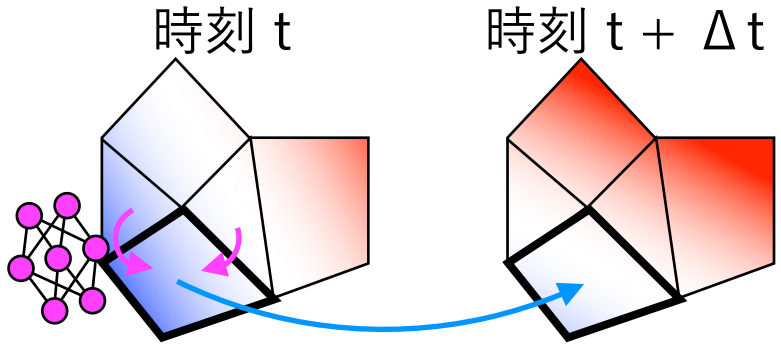


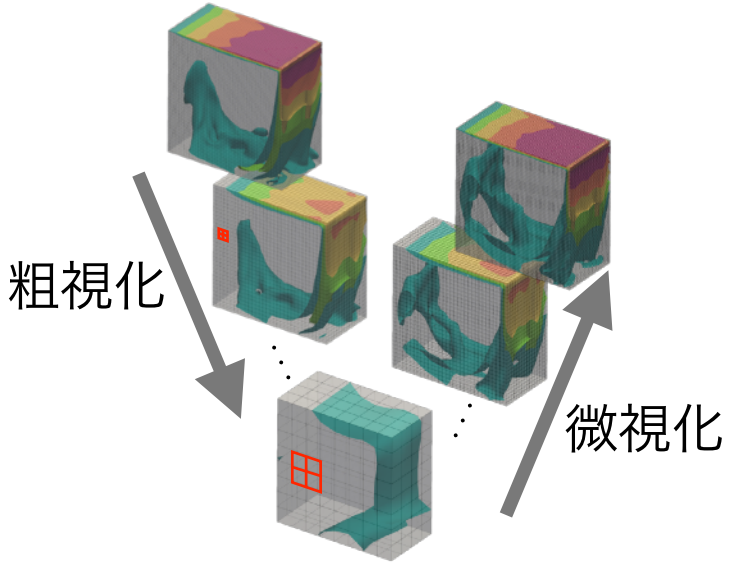

数値実験:
超解像シミュレーションと
Reynolds 数への汎化性能



# マルチグリッド: モチベーション

- FluxGNN は近傍のセルのみから影響を受ける
  モデルであるため多様なスケールを持った現象を考慮しにくい

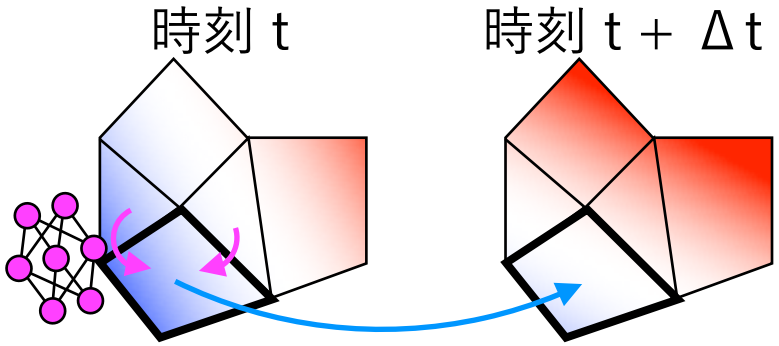


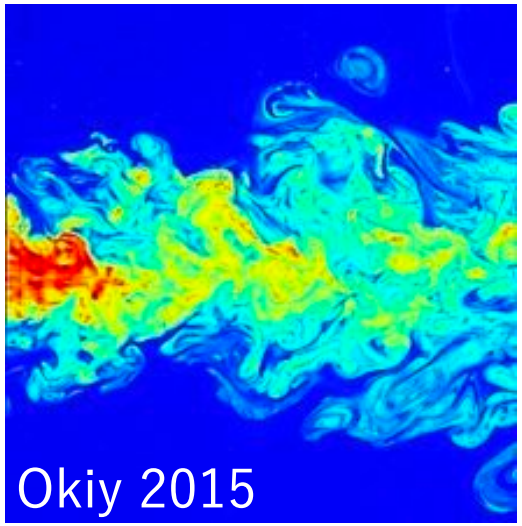



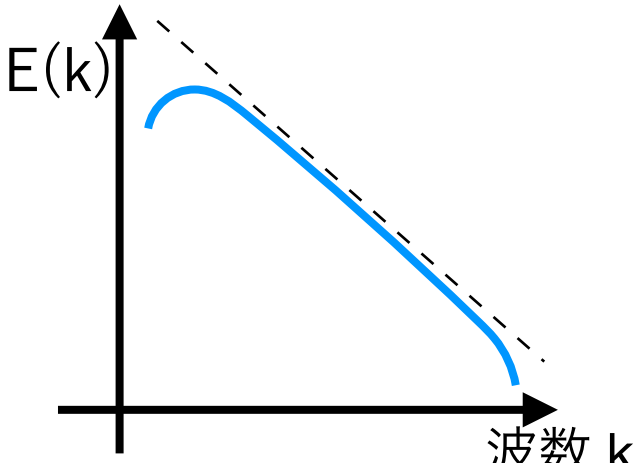
# マルチグリッド: モチベーション

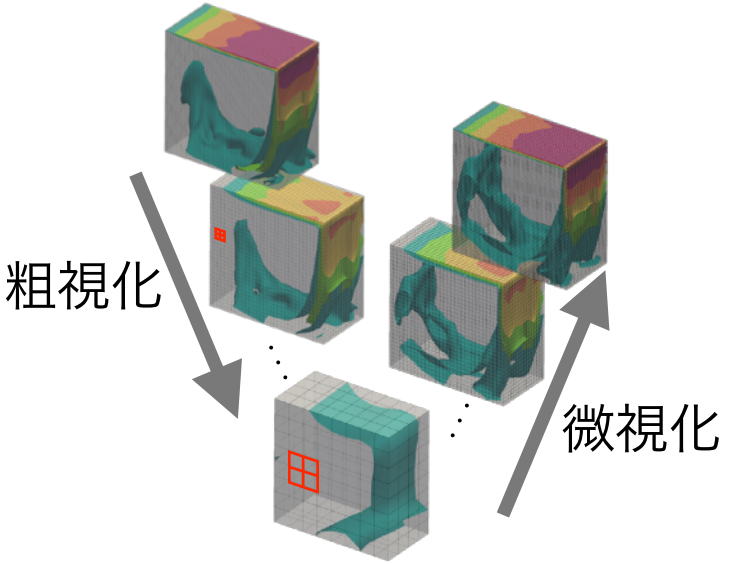
- FluxGNN は近傍のセルのみから影響を受ける
  モデルであるため多様なスケールを持った現象を考慮しにくい

- さまざまな解像度のメッシュに同一の機械学習モデル適用し
  マルチグリッド (MG) モデルを援用することで
  乱流の自己相似性・階層性に適したモデルを構築できないか？



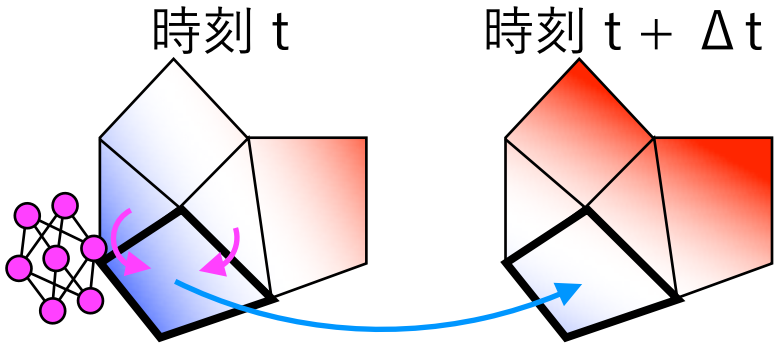



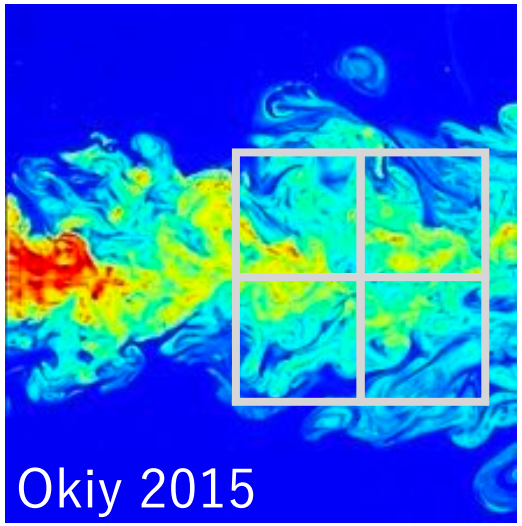
# マルチグリッド: モチベーション

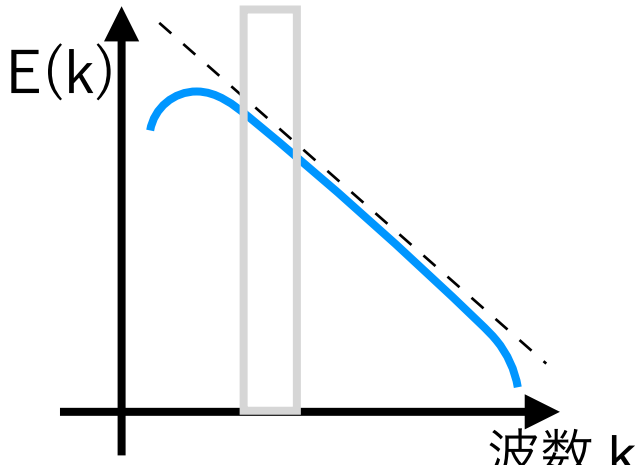
- FluxGNN は近傍のセルのみから影響を受ける
  モデルであるため多様なスケールを持った現象を考慮しにくい

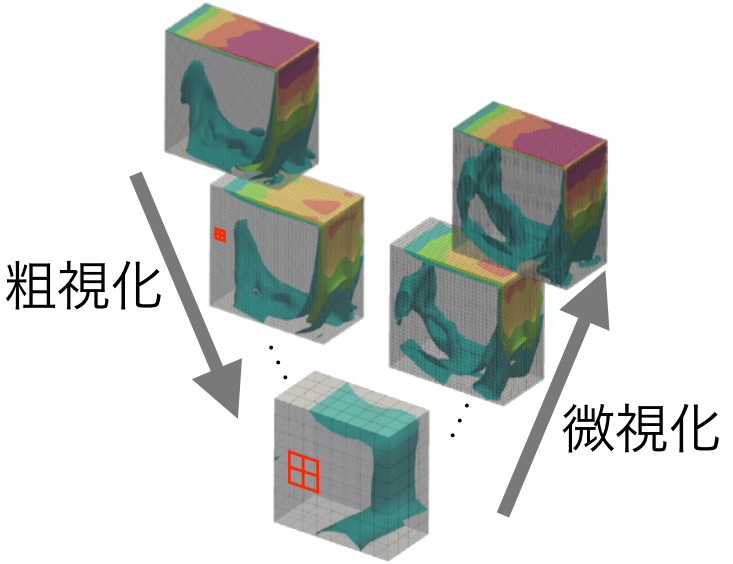
- さまざまな解像度のメッシュに同一の機械学習モデル適用し
  マルチグリッド (MG) モデルを援用することで
  乱流の自己相似性・階層性に適したモデルを構築できないか？




マルチグリッド + 機械学習モデル



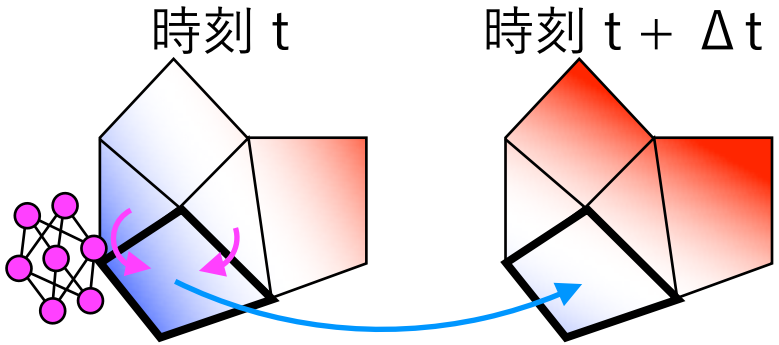



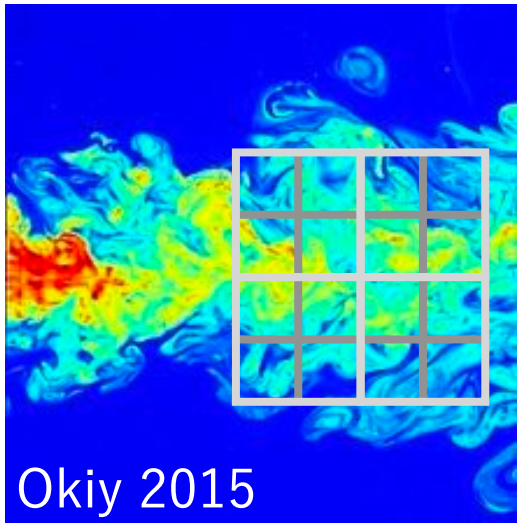



# マルチグリッド: モチベーション

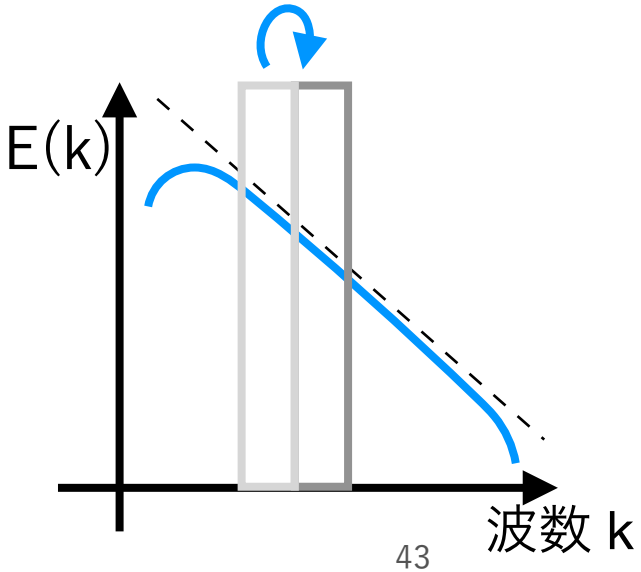
- FluxGNN は近傍のセルのみから影響を受ける
  モデルであるため多様なスケールを持った現象を考慮しにくい

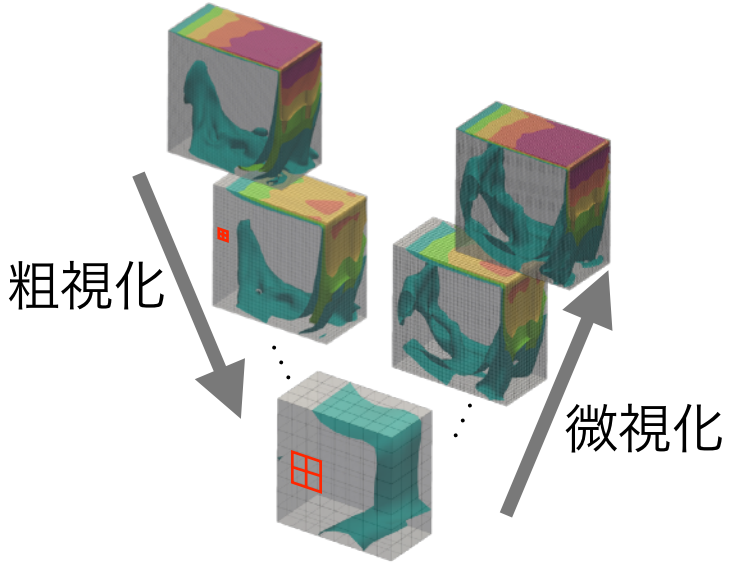
- さまざまな解像度のメッシュに同一の機械学習モデル適用し
  マルチグリッド (MG) モデルを援用することで
  乱流の自己相似性・階層性に適したモデルを構築できないか？




マルチグリッド + 機械学習モデル

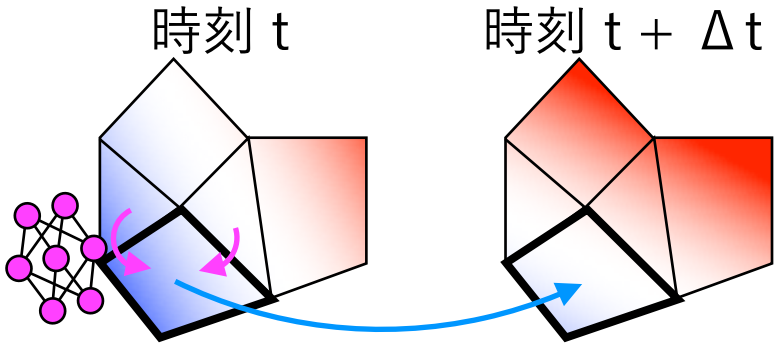


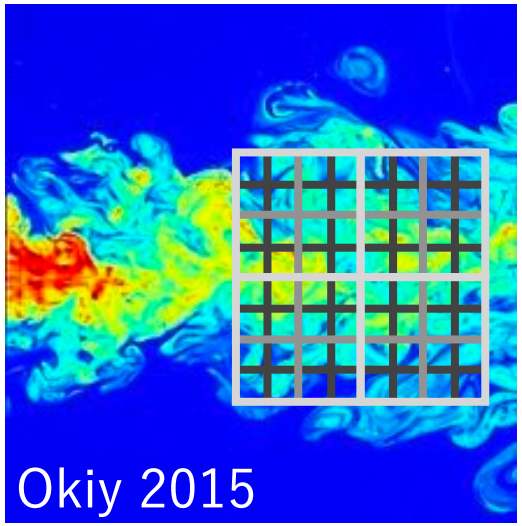



# マルチグリッド: モチベーション

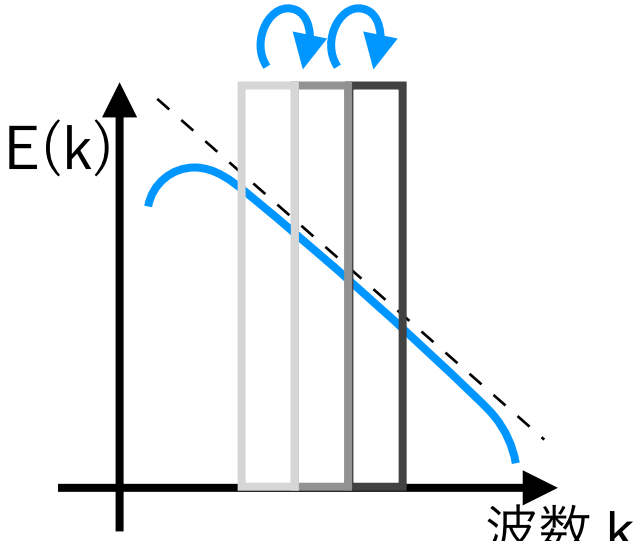
- FluxGNN は近傍のセルのみから影響を受ける
  モデルであるため多様なスケールを持った現象を考慮しにくい

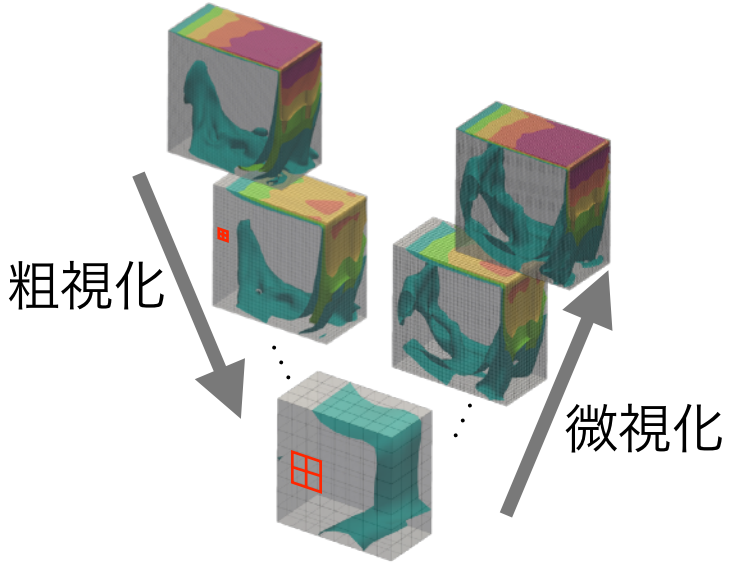
- さまざまな解像度のメッシュに同一の機械学習モデル適用し
  マルチグリッド (MG) モデルを援用することで
  乱流の自己相似性・階層性に適したモデルを構築できないか？




マルチグリッド + 機械学習モデル



# マルチグリッド: 定式化 (Full Approximation Scheme) [Briggs et al. 2022]

解像度 $h$ における解きたい方程式:

$$\boldsymbol{A}^h(\boldsymbol{u}^h) = \boldsymbol{f}^h$$

ただし $\boldsymbol{A}^h : \mathbb{R}^{n_h} \to \mathbb{R}^{n_h}$ は非線形作用素、 $\boldsymbol{f}^h \in \mathbb{R}^{n_h}$ は既知、 $\boldsymbol{u}^h \in \mathbb{R}^{n_h}$ は求めたい解

# マルチグリッド: 定式化 (Full Approximation Scheme) [Briggs et al. 2022]

解像度 $h$ における解きたい方程式: $\boldsymbol{A}^h(\boldsymbol{u}^h) = \boldsymbol{f}^h$

ただし $\boldsymbol{A}^h : \mathbb{R}^{n_h} \to \mathbb{R}^{n_h}$ は非線形作用素、 $\boldsymbol{f}^h \in \mathbb{R}^{n_h}$ は既知、 $\boldsymbol{u}^h \in \mathbb{R}^{n_h}$ は求めたい解

現在の予測 $\boldsymbol{v}^h$ に対する残差: $\boldsymbol{r}^h := \boldsymbol{f}^h - \boldsymbol{A}^h(\boldsymbol{v}^h)$

現在の予測 $\boldsymbol{v}^h$ に対する誤差: $\boldsymbol{e}^h := \boldsymbol{u}^h - \boldsymbol{v}^h$

# マルチグリッド: 定式化 (Full Approximation Scheme) [Briggs et al. 2022]

解像度 $h$ における解きたい方程式: $\boldsymbol{A}^h(\boldsymbol{u}^h) = \boldsymbol{f}^h$

ただし $\boldsymbol{A}^h : \mathbb{R}^{n_h} \to \mathbb{R}^{n_h}$ は非線形作用素、 $\boldsymbol{f}^h \in \mathbb{R}^{n_h}$ は既知、 $\boldsymbol{u}^h \in \mathbb{R}^{n_h}$ は求めたい解

現在の予測 $\boldsymbol{v}^h$ に対する残差: $\boldsymbol{r}^h := \boldsymbol{f}^h - \boldsymbol{A}^h(\boldsymbol{v}^h)$

現在の予測 $\boldsymbol{v}^h$ に対する誤差: $\boldsymbol{e}^h := \boldsymbol{u}^h - \boldsymbol{v}^h$

これまでの関係式を使用: $\boldsymbol{A}^h(\boldsymbol{u}^h) - \boldsymbol{A}^h(\boldsymbol{v}^h) = \boldsymbol{r}^h$

$h$ を粗い解像度 $2h$ に置き換え: $\boldsymbol{A}^{2h}(\boldsymbol{u}^{2h}) - \boldsymbol{A}^{2h}(\boldsymbol{v}^{2h}) = \boldsymbol{r}^{2h}$

# マルチグリッド: 定式化 (Full Approximation Scheme) [Briggs et al. 2022]

解像度 $h$ における解きたい方程式: $\boldsymbol{A}^h(\boldsymbol{u}^h) = \boldsymbol{f}^h$

ただし $\boldsymbol{A}^h : \mathbb{R}^{n_h} \to \mathbb{R}^{n_h}$ は非線形作用素、 $\boldsymbol{f}^h \in \mathbb{R}^{n_h}$ は既知、 $\boldsymbol{u}^h \in \mathbb{R}^{n_h}$ は求めたい解

現在の予測 $\boldsymbol{v}^h$ に対する残差: $\boldsymbol{r}^h := \boldsymbol{f}^h - \boldsymbol{A}^h(\boldsymbol{v}^h)$

現在の予測 $\boldsymbol{v}^h$ に対する誤差: $\boldsymbol{e}^h := \boldsymbol{u}^h - \boldsymbol{v}^h$

これまでの関係式を使用: $\boldsymbol{A}^h(\boldsymbol{u}^h) - \boldsymbol{A}^h(\boldsymbol{v}^h) = \boldsymbol{r}^h$

$h$ を粗い解像度 $2h$ に置き換え: $\boldsymbol{A}^{2h}(\boldsymbol{u}^{2h}) - \boldsymbol{A}^{2h}(\boldsymbol{v}^{2h}) = \boldsymbol{r}^{2h}$

粗い解像度への補間 $\boldsymbol{I}_h^{2h} : \mathbb{R}^{n_h} \to \mathbb{R}^{n_{2h}}$ を使用: $\boldsymbol{A}^{2h}(\boldsymbol{u}^{2h}) = \boldsymbol{I}_h^{2h}\boldsymbol{r}^h - \boldsymbol{A}^{2h}(\boldsymbol{I}_h^{2h}\boldsymbol{v}^h)$

# マルチグリッド: 定式化 (Full Approximation Scheme) [Briggs et al. 2022]

解像度 $h$ における解きたい方程式: $\boldsymbol{A}^h(\boldsymbol{u}^h) = \boldsymbol{f}^h$

ただし $\boldsymbol{A}^h : \mathbb{R}^{n_h} \to \mathbb{R}^{n_h}$ は非線形作用素、 $\boldsymbol{f}^h \in \mathbb{R}^{n_h}$ は既知、 $\boldsymbol{u}^h \in \mathbb{R}^{n_h}$ は求めたい解

現在の予測 $\boldsymbol{v}^h$ に対する残差: $\boldsymbol{r}^h := \boldsymbol{f}^h - \boldsymbol{A}^h(\boldsymbol{v}^h)$

現在の予測 $\boldsymbol{v}^h$ に対する誤差: $\boldsymbol{e}^h := \boldsymbol{u}^h - \boldsymbol{v}^h$

これまでの関係式を使用: $\boldsymbol{A}^h(\boldsymbol{u}^h) - \boldsymbol{A}^h(\boldsymbol{v}^h) = \boldsymbol{r}^h$

$h$ を粗い解像度 $2h$ に置き換え: $\boldsymbol{A}^{2h}(\boldsymbol{u}^{2h}) - \boldsymbol{A}^{2h}(\boldsymbol{v}^{2h}) = \boldsymbol{r}^{2h}$

粗い解像度への補間 $\boldsymbol{I}_h^{2h} : \mathbb{R}^{n_h} \to \mathbb{R}^{n_{2h}}$ を使用: $\boldsymbol{A}^{2h}(\boldsymbol{u}^{2h}) = \boldsymbol{I}_h^{2h}\boldsymbol{r}^h - \boldsymbol{A}^{2h}(\boldsymbol{I}_h^{2h}\boldsymbol{v}^h)$

もとの方程式を単純に粗くしたものではない！

# マルチグリッド: 定式化 (Full Approximation Scheme) [Briggs et al. 2022]

解像度 $h$ における解きたい方程式: $\boldsymbol{A}^h(\boldsymbol{u}^h) = \boldsymbol{f}^h$

ただし $\boldsymbol{A}^h : \mathbb{R}^{n_h} \to \mathbb{R}^{n_h}$ は非線形作用素、 $\boldsymbol{f}^h \in \mathbb{R}^{n_h}$ は既知、 $\boldsymbol{u}^h \in \mathbb{R}^{n_h}$ は求めたい解

現在の予測 $\boldsymbol{v}^h$ に対する残差: $\boldsymbol{r}^h := \boldsymbol{f}^h - \boldsymbol{A}^h(\boldsymbol{v}^h)$

現在の予測 $\boldsymbol{v}^h$ に対する誤差: $\boldsymbol{e}^h := \boldsymbol{u}^h - \boldsymbol{v}^h$

これまでの関係式を使用: $\boldsymbol{A}^h(\boldsymbol{u}^h) - \boldsymbol{A}^h(\boldsymbol{v}^h) = \boldsymbol{r}^h$

$h$ を粗い解像度 $2h$ に置き換え: $\boldsymbol{A}^{2h}(\boldsymbol{u}^{2h}) - \boldsymbol{A}^{2h}(\boldsymbol{v}^{2h}) = \boldsymbol{r}^{2h}$

粗い解像度への補間 $\boldsymbol{I}_h^{2h} : \mathbb{R}^{n_h} \to \mathbb{R}^{n_{2h}}$ を使用: $\boldsymbol{A}^{2h}(\boldsymbol{u}^{2h}) = \boldsymbol{I}_h^{2h}\boldsymbol{r}^h - \boldsymbol{A}^{2h}(\boldsymbol{I}_h^{2h}\boldsymbol{v}^h)$

もとの方程式を単純に粗くしたものではない！

$\boldsymbol{u}^{2h}$ を (近似的に) 求め得られた $\boldsymbol{e}^{2h}$ で予測を補正: $\boldsymbol{v}^h \leftarrow \boldsymbol{v}^h + \boldsymbol{I}_{2h}^h \boldsymbol{e}^{2h}$

# マルチグリッド: 定式化 (Full Approximation Scheme)

解きたい偏微分方程式:

$$\frac{\partial}{\partial t}\boldsymbol{u} = \mathcal{D}(\boldsymbol{u})$$

ただし、NS 方程式の場合は

$$\mathcal{D}(\boldsymbol{u}) := -\nabla \cdot [\boldsymbol{u} \otimes \boldsymbol{u} - \nu \nabla \otimes \boldsymbol{u} + p\boldsymbol{I}]$$

# マルチグリッド: 定式化 (Full Approximation Scheme)

解きたい偏微分方程式: $$\frac{\partial}{\partial t}\boldsymbol{u} = \mathcal{D}(\boldsymbol{u})$$

ただし、NS 方程式の場合は $$\mathcal{D}(\boldsymbol{u}) := -\nabla \cdot [\boldsymbol{u} \otimes \boldsymbol{u} - \nu \nabla \otimes \boldsymbol{u} + p\boldsymbol{I}]$$

既知の量 $\hat{\boldsymbol{u}} := \boldsymbol{u}(t)$ とし陰的 Euler 法で時間離散化: $$\boldsymbol{u} = \hat{\boldsymbol{u}} + \mathcal{D}(\boldsymbol{u})\Delta t$$

# マルチグリッド: 定式化 (Full Approximation Scheme)

解きたい偏微分方程式:

$$\frac{\partial}{\partial t}\boldsymbol{u} = \mathcal{D}(\boldsymbol{u})$$

ただし、NS 方程式の場合は

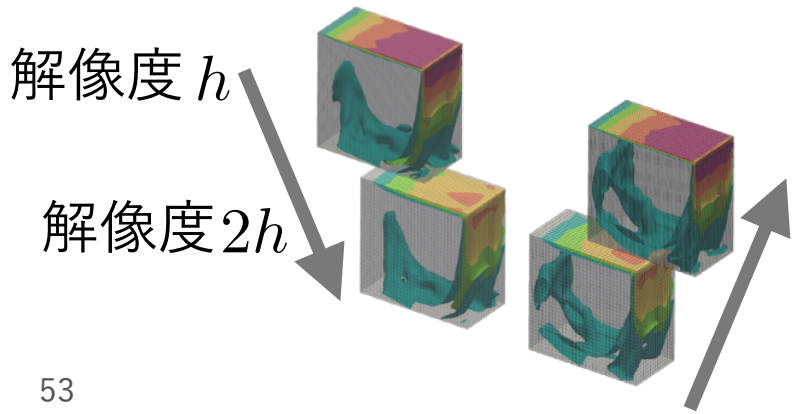
$$\mathcal{D}(\boldsymbol{u}) := -\nabla \cdot [\boldsymbol{u} \otimes \boldsymbol{u} - \nu \nabla \otimes \boldsymbol{u} + p\boldsymbol{I}]$$

既知の量 $\hat{\boldsymbol{u}} := \boldsymbol{u}(t)$ とし陰的 Euler 法で時間離散化:

$$\boldsymbol{u} = \hat{\boldsymbol{u}} + \mathcal{D}(\boldsymbol{u})\Delta t$$

したがって解くべき非線形代数方程式は:

$$\boldsymbol{A}^h(\boldsymbol{u}^h) = \boldsymbol{f}^h$$



# マルチグリッド: 定式化 (Full Approximation Scheme)

解きたい偏微分方程式:

$$\frac{\partial}{\partial t}\boldsymbol{u} = \mathcal{D}(\boldsymbol{u})$$

ただし、NS 方程式の場合は

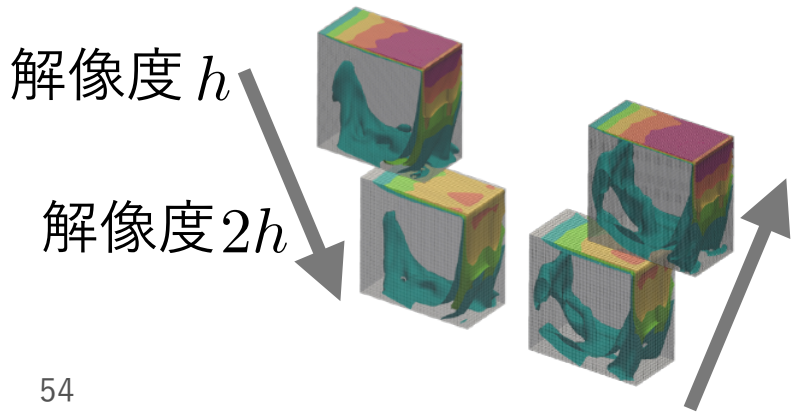
$$\mathcal{D}(\boldsymbol{u}) := -\nabla \cdot [\boldsymbol{u} \otimes \boldsymbol{u} - \nu \nabla \otimes \boldsymbol{u} + p\boldsymbol{I}]$$

既知の量 $\hat{\boldsymbol{u}} := \boldsymbol{u}(t)$ とし陰的 Euler 法で時間離散化:

$$\boldsymbol{u} = \hat{\boldsymbol{u}} + \mathcal{D}(\boldsymbol{u})\Delta t$$

したがって解くべき非線形代数方程式は:

$$\underbrace{\boldsymbol{A}^h(\boldsymbol{u}^h)}_{\boldsymbol{u}^h - \mathcal{D}^h(\boldsymbol{u}^h)\Delta t} = \underbrace{\boldsymbol{f}^h}_{\hat{\boldsymbol{u}}^h}$$



# マルチグリッド: 定式化 (Full Approximation Scheme)

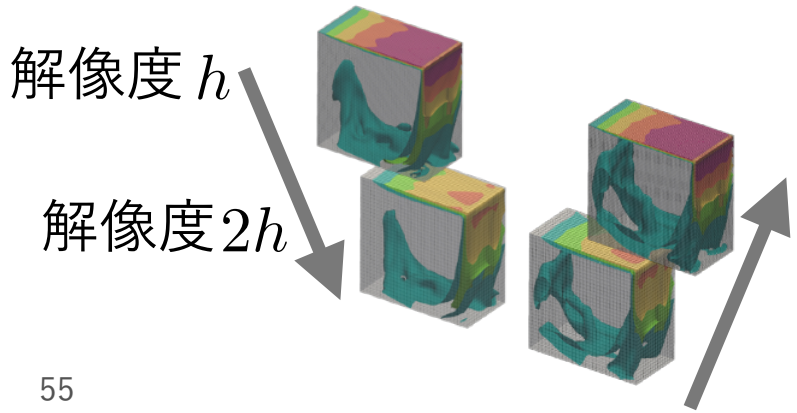
解きたい偏微分方程式: $$\frac{\partial}{\partial t}\boldsymbol{u} = \mathcal{D}(\boldsymbol{u})$$

ただし、NS 方程式の場合は $$\mathcal{D}(\boldsymbol{u}) := -\nabla \cdot [\boldsymbol{u} \otimes \boldsymbol{u} - \nu \nabla \otimes \boldsymbol{u} + p\boldsymbol{I}]$$

既知の量 $\hat{\boldsymbol{u}} := \boldsymbol{u}(t)$ とし陰的 Euler 法で時間離散化: $$\boldsymbol{u} = \hat{\boldsymbol{u}} + \mathcal{D}(\boldsymbol{u})\Delta t$$

したがって解くべき非線形代数方程式は: $$\underbrace{\boldsymbol{A}^h(\boldsymbol{u}^h)}_{\boldsymbol{u}^h - \mathcal{D}^h(\boldsymbol{u}^h)\Delta t} = \underbrace{\boldsymbol{f}^h}_{\hat{\boldsymbol{u}}^h}$$

粗いグリッドでの方程式は: $\boldsymbol{u}^{2h} - \mathcal{D}^{2h}(\boldsymbol{u}^{2h})\Delta t = \hat{\boldsymbol{u}}^{2h} + \boldsymbol{I}_h^{2h}\mathcal{D}^h(\boldsymbol{v}^h)\Delta t - \mathcal{D}^{2h}(\boldsymbol{v}^{2h})\Delta t$



# マルチグリッド: 定式化 (Full Approximation Scheme)

解きたい偏微分方程式: $$\frac{\partial}{\partial t}\boldsymbol{u} = \mathcal{D}(\boldsymbol{u})$$

ただし、NS 方程式の場合は $$\mathcal{D}(\boldsymbol{u}) := -\nabla \cdot [\boldsymbol{u} \otimes \boldsymbol{u} - \nu \nabla \otimes \boldsymbol{u} + p\boldsymbol{I}]$$

既知の量 $\hat{\boldsymbol{u}} := \boldsymbol{u}(t)$ とし陰的 Euler 法で時間離散化: $$\boldsymbol{u} = \hat{\boldsymbol{u}} + \mathcal{D}(\boldsymbol{u})\Delta t$$

したがって解くべき非線形代数方程式は:

$$\begin{aligned} \overbrace{\boldsymbol{A}^h(\boldsymbol{u}^h)}^{\boldsymbol{u}^h - \mathcal{D}^h(\boldsymbol{u}^h)\Delta t} &= \underbrace{\boldsymbol{f}^h}_{\hat{\boldsymbol{u}}^h} \end{aligned}$$

粗いグリッドでの方程式は:

$$\begin{aligned} \overbrace{\boldsymbol{A}^{2h}(\boldsymbol{u}^{2h})}^{\boldsymbol{u}^{2h} - \mathcal{D}^{2h}(\boldsymbol{u}^{2h})\Delta t} &= \overbrace{\boldsymbol{f}^{2h}}^{\hat{\boldsymbol{u}}^{2h} + \boldsymbol{I}_h^{2h}\mathcal{D}^h(\boldsymbol{v}^h)\Delta t - \mathcal{D}^{2h}(\boldsymbol{v}^{2h})\Delta t} \end{aligned}$$

# マルチグリッド: 定式化 (Full Approximation Scheme)

解きたい偏微分方程式: $\dfrac{\partial}{\partial t}\boldsymbol{u} = \mathcal{D}(\boldsymbol{u})$

ただし、NS 方程式の場合は $\mathcal{D}(\boldsymbol{u}) := -\nabla \cdot [\boldsymbol{u} \otimes \boldsymbol{u} - \nu \nabla \otimes \boldsymbol{u} + p\boldsymbol{I}]$

既知の量 $\hat{\boldsymbol{u}} := \boldsymbol{u}(t)$ とし陰的 Euler 法で時間離散化: $\boldsymbol{u} = \hat{\boldsymbol{u}} + \mathcal{D}(\boldsymbol{u})\Delta t$

したがって解くべき非線形代数方程式は:

$$\underbrace{\boldsymbol{A}^h(\boldsymbol{u}^h)}_{\boldsymbol{u}^h - \mathcal{D}^h(\boldsymbol{u}^h)\Delta t} = \underbrace{\boldsymbol{f}^h}_{\hat{\boldsymbol{u}}^h}$$

粗いグリッドでの方程式は:

$$\underbrace{\boldsymbol{A}^{2h}(\boldsymbol{u}^{2h})}_{\boldsymbol{u}^{2h} - \mathcal{D}^{2h}(\boldsymbol{u}^{2h})\Delta t} = \underbrace{\boldsymbol{f}^{2h}}_{\hat{\boldsymbol{u}}^{2h} + \boldsymbol{I}_h^{2h}\mathcal{D}^h(\boldsymbol{v}^h)\Delta t - \mathcal{D}^{2h}(\boldsymbol{v}^{2h})\Delta t}$$

補間操作の交換関係によって
生じる「外力項」

# マルチグリッド: 定式化 (Full Approximation Scheme)

解きたい偏微分方程式: $\dfrac{\partial}{\partial t}\boldsymbol{u} = \mathcal{D}(\boldsymbol{u})$

ただし、NS 方程式の場合は $\mathcal{D}(\boldsymbol{u}) := -\nabla \cdot [\boldsymbol{u} \otimes \boldsymbol{u} - \nu \nabla \otimes \boldsymbol{u} + p\boldsymbol{I}]$

既知の量 $\hat{\boldsymbol{u}} := \boldsymbol{u}(t)$ とし陰的 Euler 法で時間離散化: $\boldsymbol{u} = \hat{\boldsymbol{u}} + \mathcal{D}(\boldsymbol{u})\Delta t$

したがって解くべき非線形代数方程式は:

$$\underbrace{\boldsymbol{A}^h(\boldsymbol{u}^h)}_{\boldsymbol{u}^h - \mathcal{D}^h(\boldsymbol{u}^h)\Delta t} = \underbrace{\boldsymbol{f}^h}_{\hat{\boldsymbol{u}}^h}$$

粗いグリッドでの方程式は:

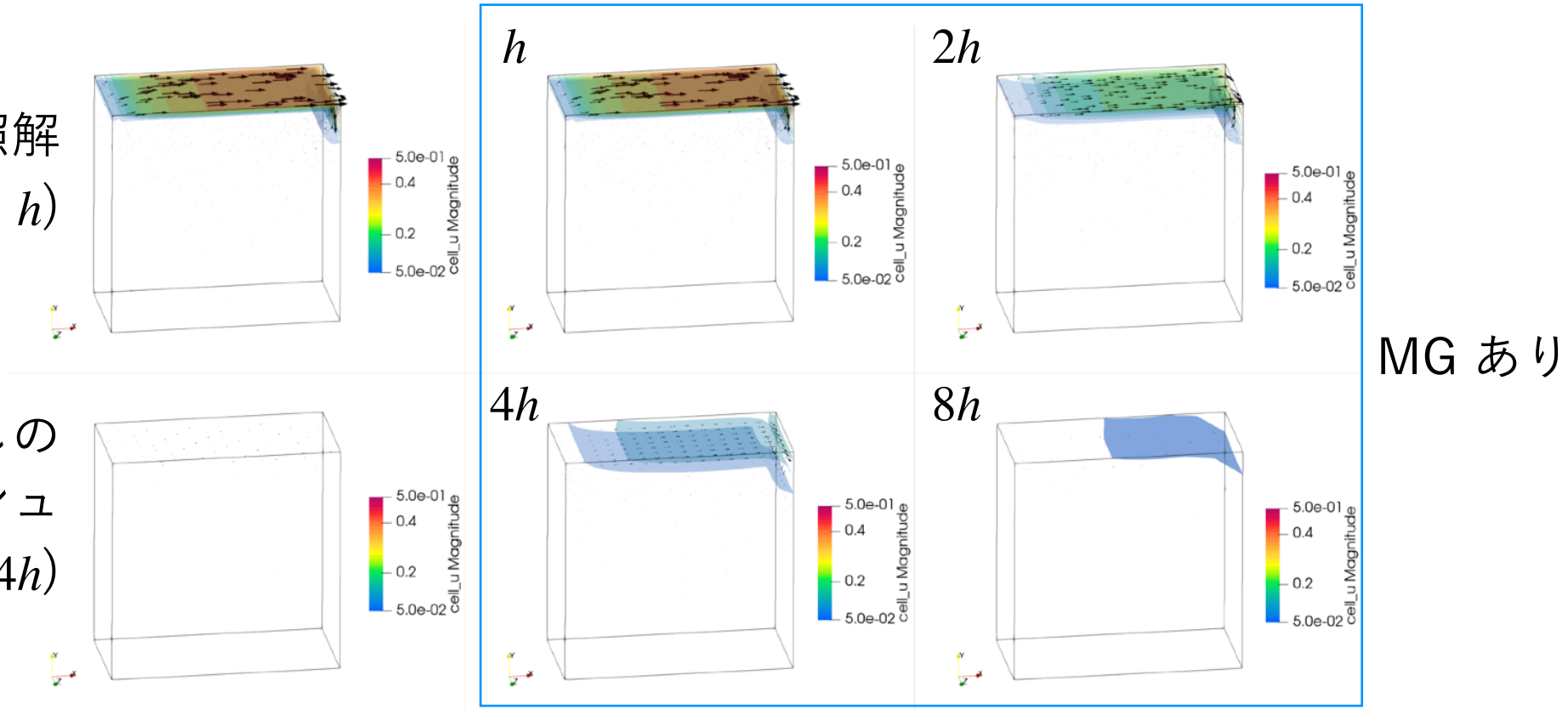
$$\underbrace{\boldsymbol{A}^{2h}(\boldsymbol{u}^{2h})}_{\boldsymbol{u}^{2h} - \mathcal{D}^{2h}(\boldsymbol{u}^{2h})\Delta t} = \underbrace{\boldsymbol{f}^{2h}}_{\hat{\boldsymbol{u}}^{2h} + \boldsymbol{I}_h^{2h}\mathcal{D}^h(\boldsymbol{v}^h)\Delta t - \mathcal{D}^{2h}(\boldsymbol{v}^{2h})\Delta t}$$

Flux $\mathcal{D}^h$, $\mathcal{D}^{2h}$の近似に同一の機械学習モデルを用いることで、さまざまな解像度に対応できる

補間操作の交換関係によって生じる「外力項」

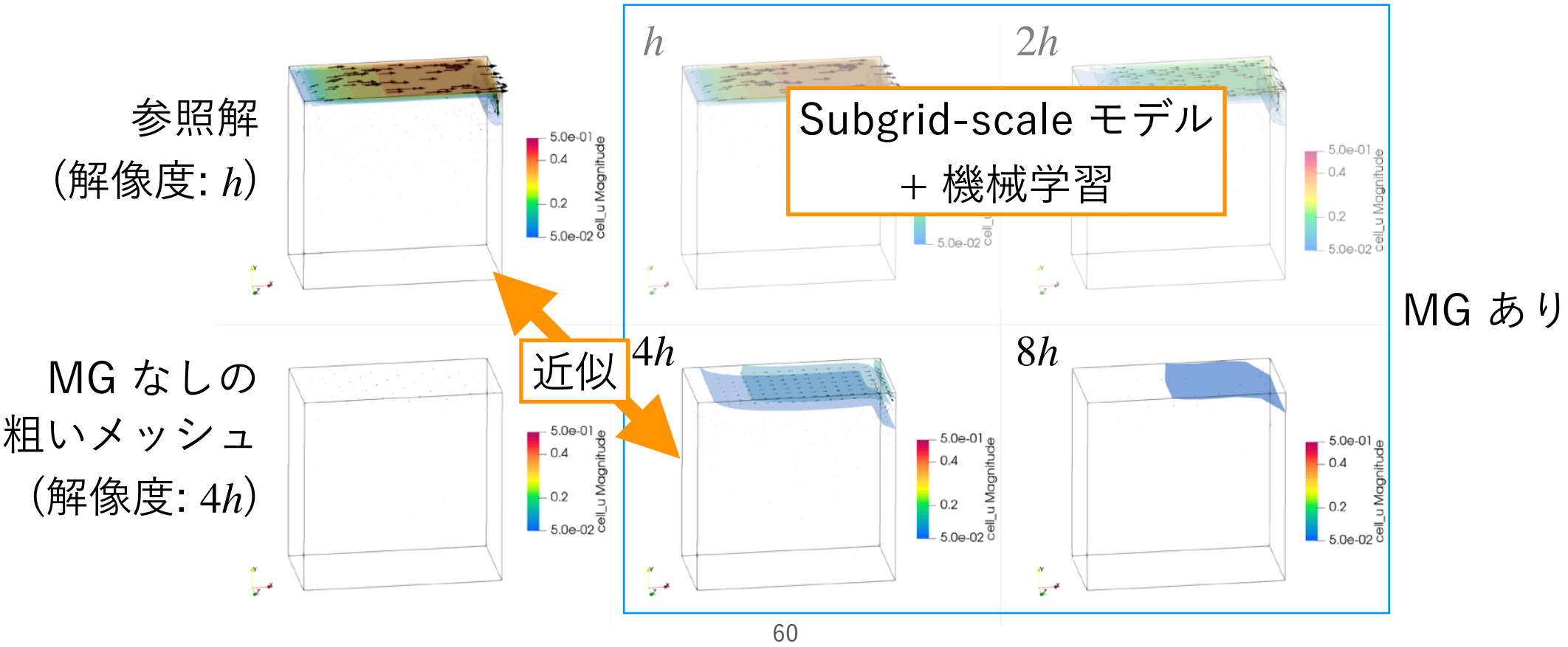
# マルチグリッド: 機械学習なしでの予備検討

- 機械学習を用いない予備検討において MG 法による外力の補正項により粗いグリッドでも細かいグリッドの数値解と近い結果が得られた



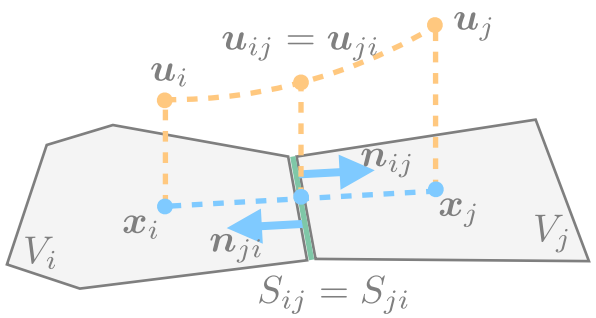
# マルチグリッド: 機械学習なしでの予備検討

- 機械学習を用いない予備検討において MG 法による外力の補正項により
  粗いグリッドでも細かいグリッドの数値解と近い結果が得られた
- 外力を SGS (subgrid-scale) モデルとして導入した乱流モデルの構築が進行中

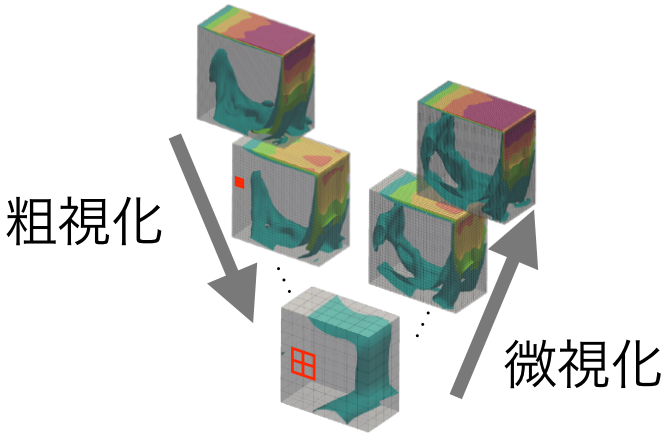


# 目次

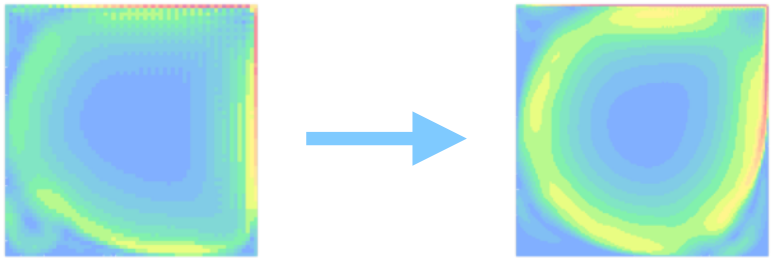
FluxGNN:
有限体積法ベースの
汎用的機械学習モデル




非線形マルチグリッド:
複数の解像度を協働させた
シミュレーション手法

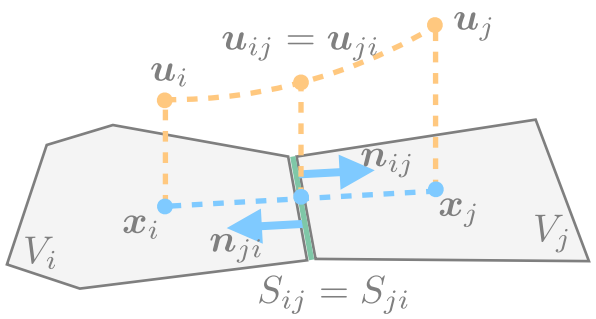



数値実験:
超解像シミュレーションと
Reynolds 数への汎化性能

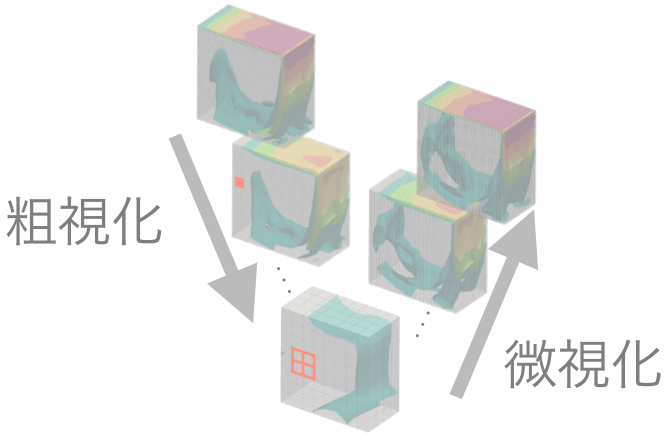


# 目次

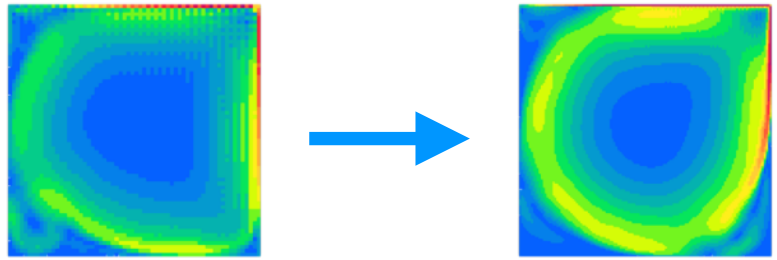
FluxGNN:
有限体積法ベースの
汎用的機械学習モデル




非線形マルチグリッド:
複数の解像度を協働させた
シミュレーション手法



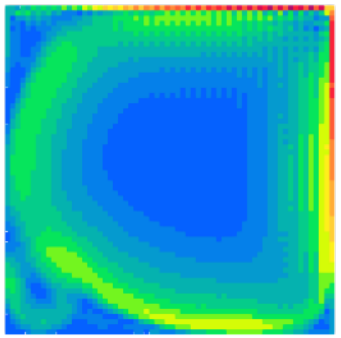

数値実験:
超解像シミュレーションと
Reynolds 数への汎化性能



# 数値実験: 超解像シミュレーション

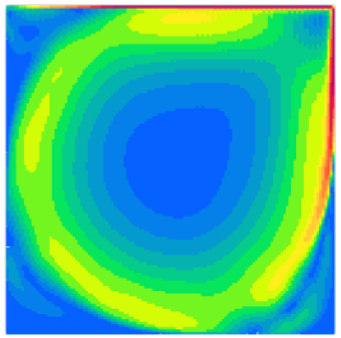
入力

流速



予測したい

参照解

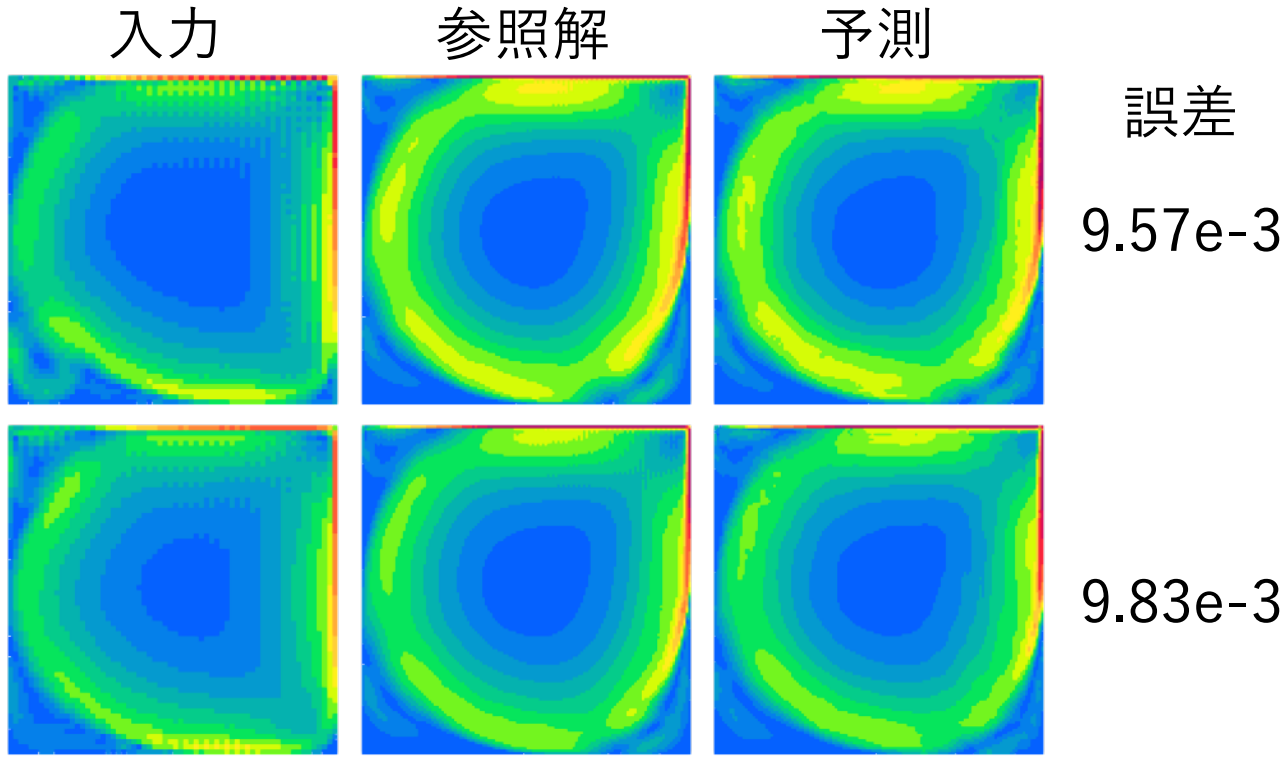


粗い解像度での
統計的定常状態

細かい解像度での
統計的定常状態

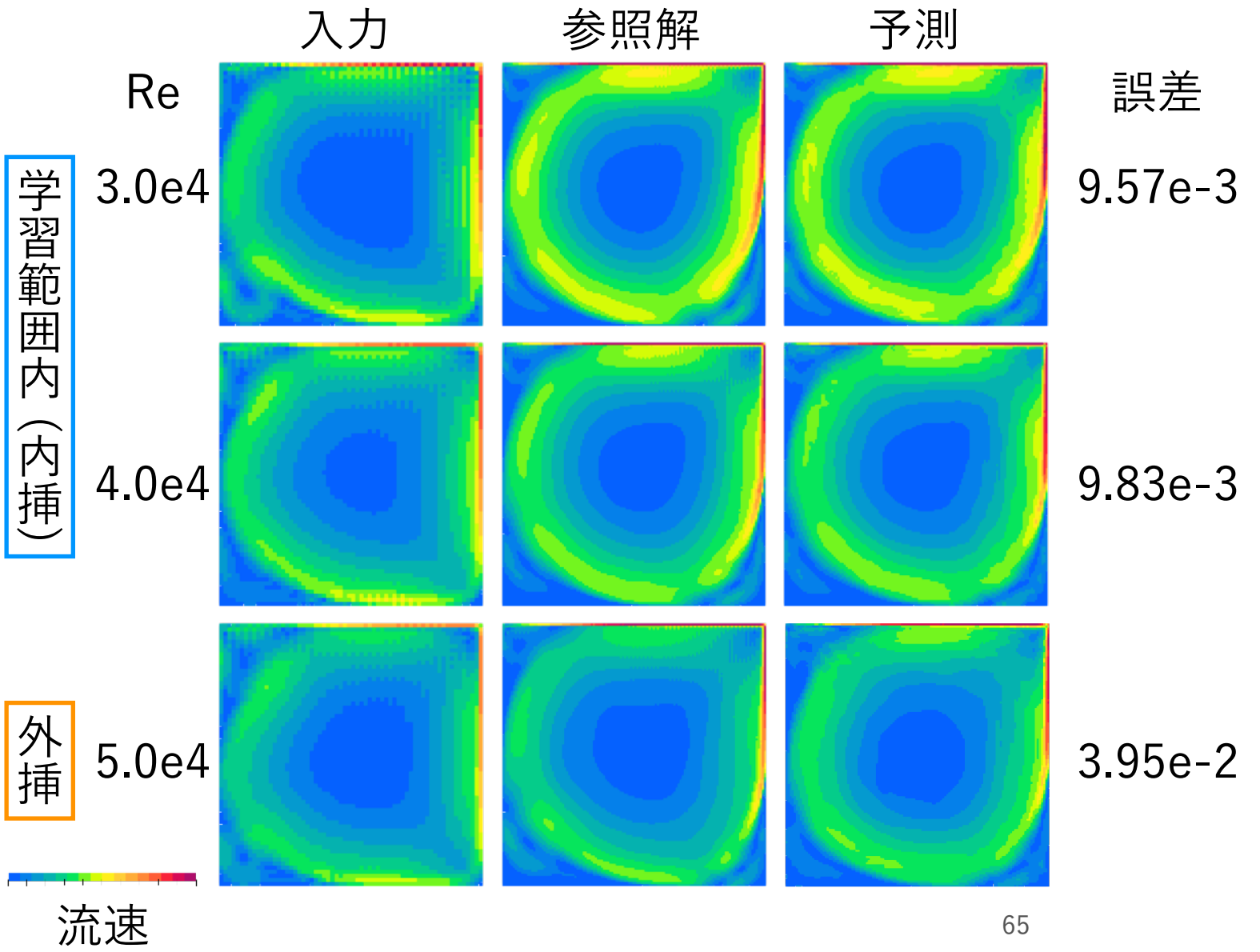
- マルチグリッドにより
  粗い解像度のシミュレーションを
  入力として細かい解像度の
  シミュレーションを出力する
  ことが可能 → 超解像

# 数値実験: 超解像シミュレーション




- マルチグリッドにより
  粗い解像度のシミュレーションを
  入力として細かい解像度の
  シミュレーションを出力する
  ことが可能 → 超解像

# 数値実験: 超解像シミュレーション

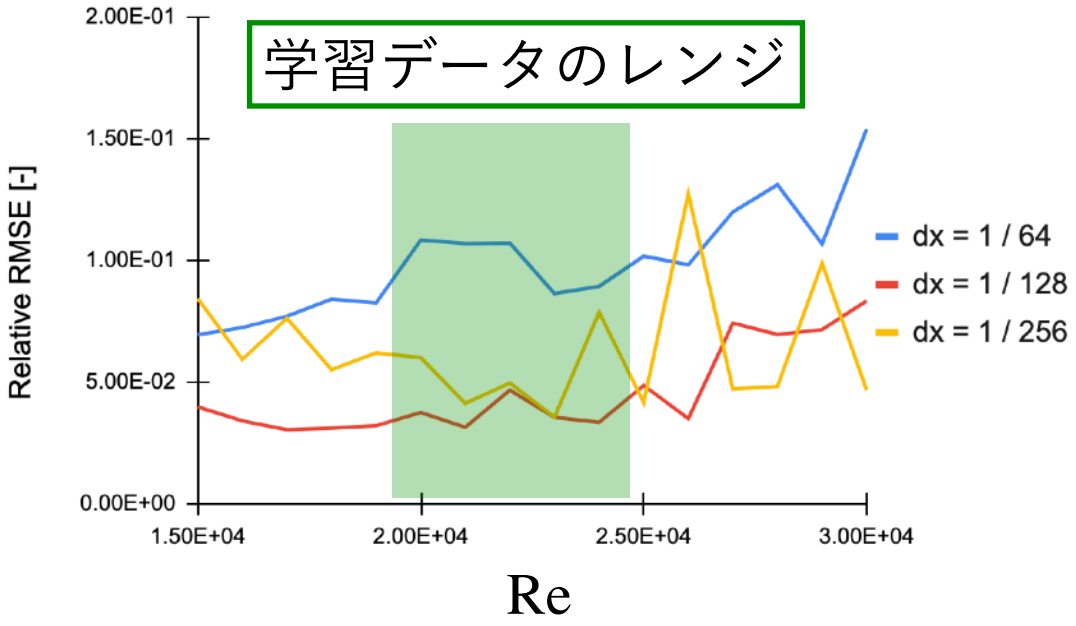



- マルチグリッドにより
  粗い解像度のシミュレーションを
  入力として細かい解像度の
  シミュレーションを出力する
  ことが可能 → 超解像

- 誤差は拡大するものの
  Re の外挿に対しても
  ある程度良好な精度の予測が実現
- 同精度の比較ではないが
  DNS では最低数日かかる計算が
  10 秒程度にまで短縮

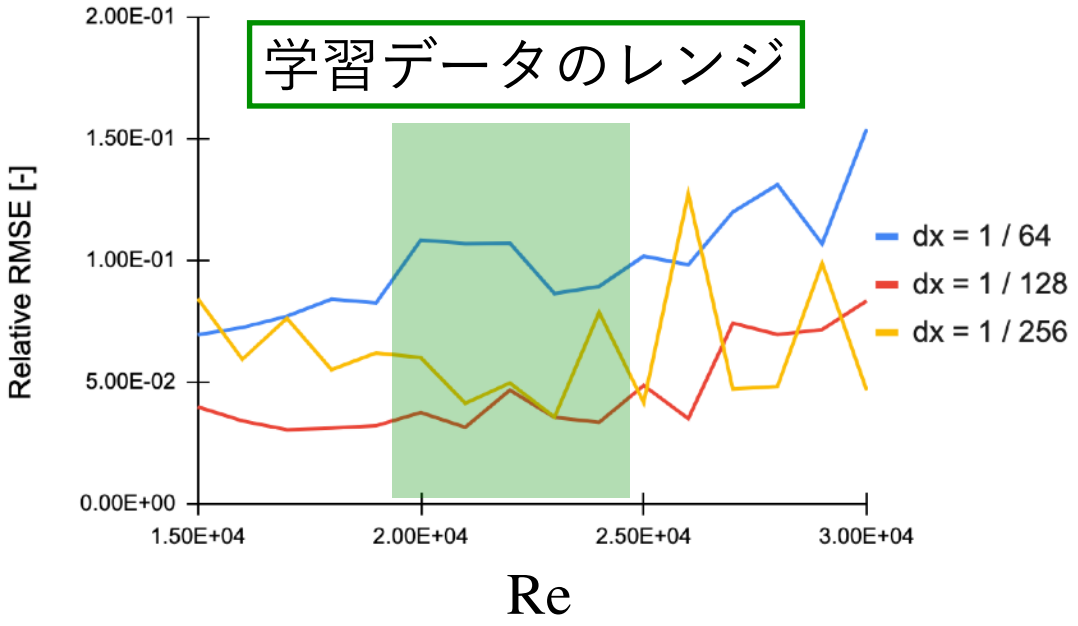
# 数値実験: Reynolds 数外挿性能

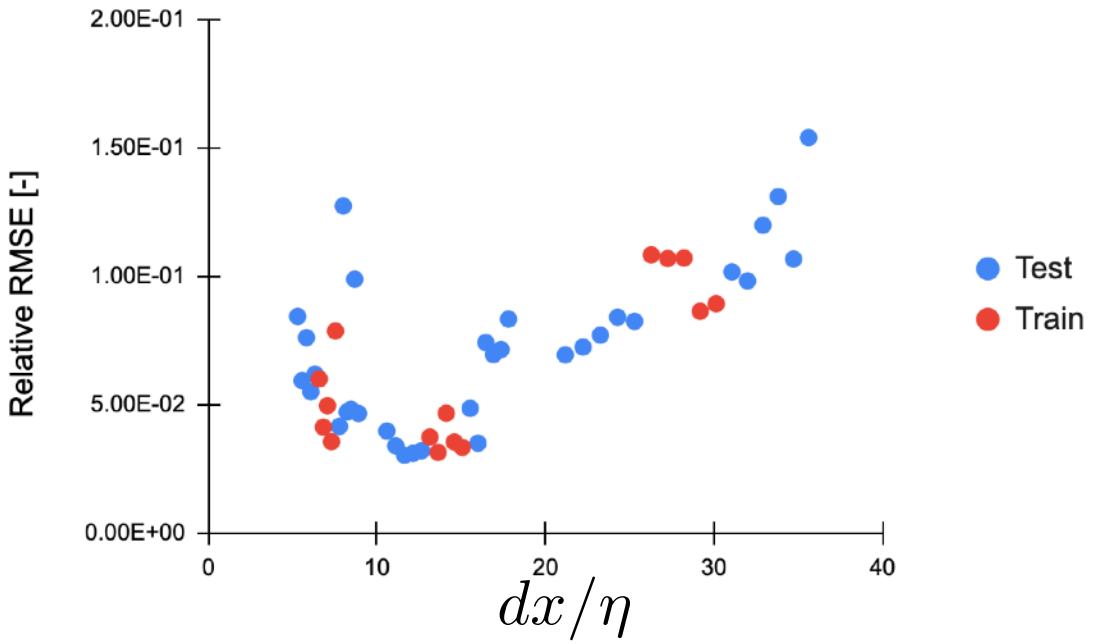
- 複数の解像度を同一の機械学習モデルで学習
  - Reynolds 数についての外挿性能は
    入力の解像度が高いほうが高い



# 数値実験: Reynolds 数外挿性能

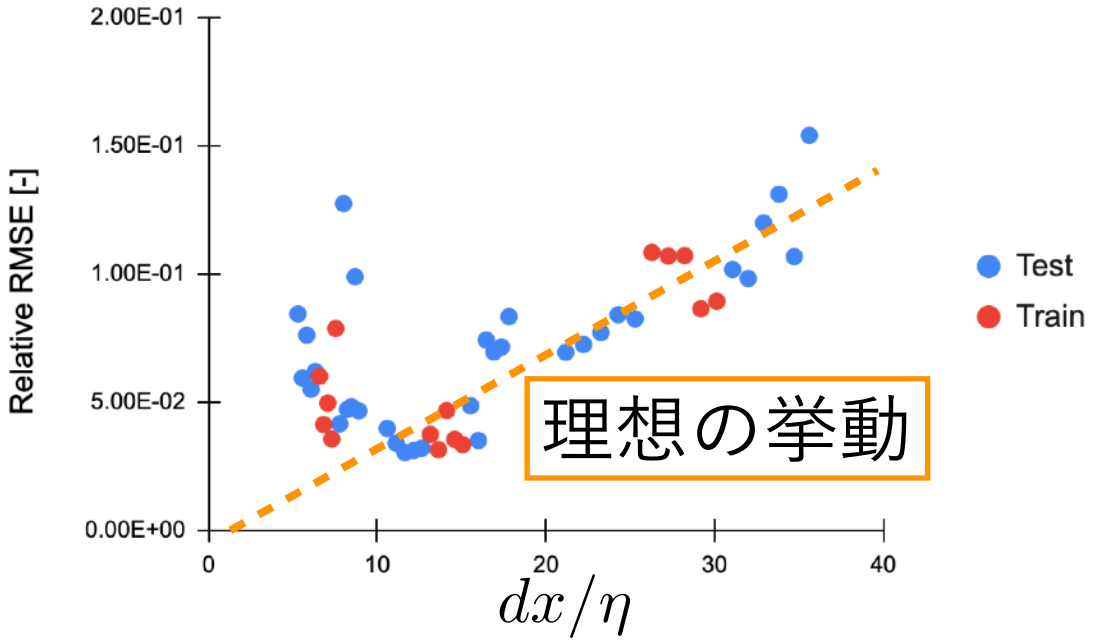
- 複数の解像度を同一の機械学習モデルで学習
  - Reynolds 数についての外挿性能は
    入力の解像度が高いほうが高い
  - Kolmogorov 長 $\eta \sim L\,\mathrm{Re}^{3/4}$ と空間刻み幅の
    関係に依存して予測精度の傾向が決定
    - $dx/\eta$ 大: 入力の解像度が低すぎて
      予測が難しくなる
    - $dx/\eta$ 小: 解像度が小さすぎて
      予測計算が不安定化する



# 数値実験: Reynolds 数外挿性能

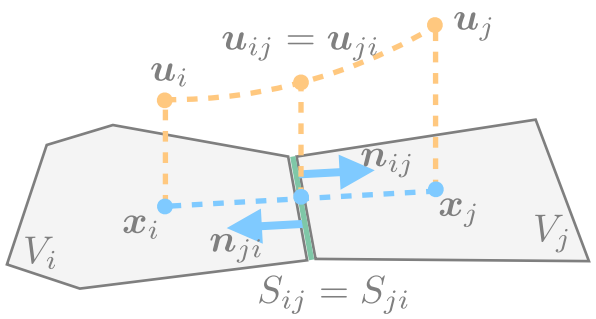
- 複数の解像度を同一の機械学習モデルで学習
  - Reynolds 数についての外挿性能は
    入力の解像度が高いほうが高い
  - Kolmogorov 長 $\eta \sim L\,\mathrm{Re}^{3/4}$ と空間刻み幅の
    関係に依存して予測精度の傾向が決定
    - $dx/\eta$ 大: 入力の解像度が低すぎて
      予測が難しくなる
    - $dx/\eta$ 小: 解像度が小さすぎて
      予測計算が不安定化する



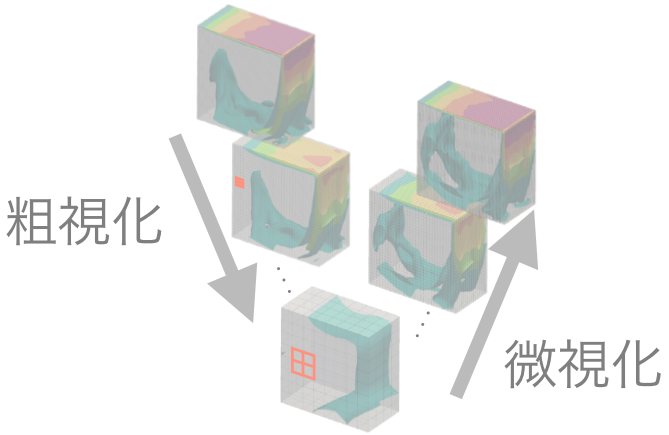

➡ 任意の解像度で安定して予測できるようにすれば任意の Reynolds 数で
超解像シミュレーションの機械学習による予測が可能？

# 目次

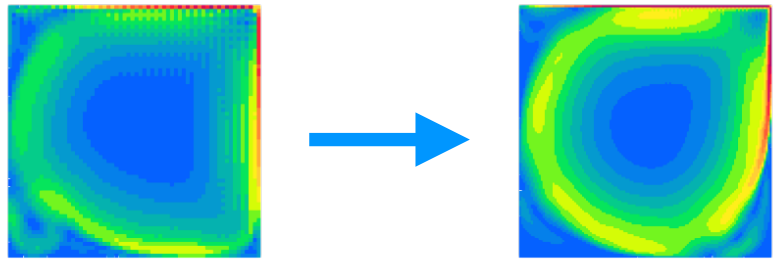
FluxGNN:
有限体積法ベースの
汎用的機械学習モデル




非線形マルチグリッド:
複数の解像度を協働させた
シミュレーション手法

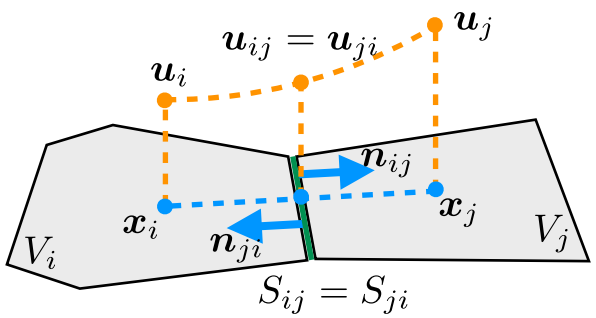



数値実験:
超解像シミュレーションと
Reynolds 数への汎化性能

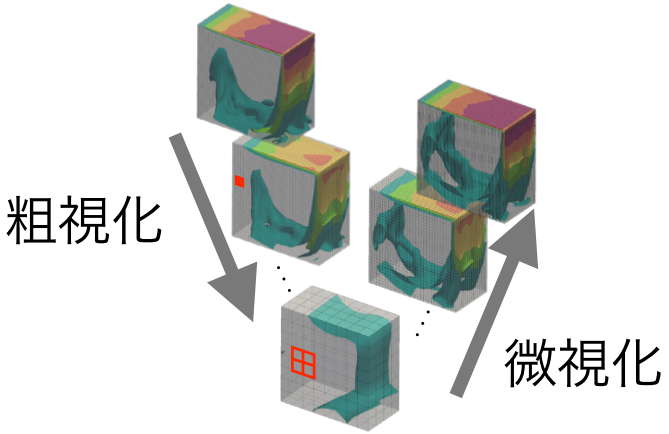


# 目次

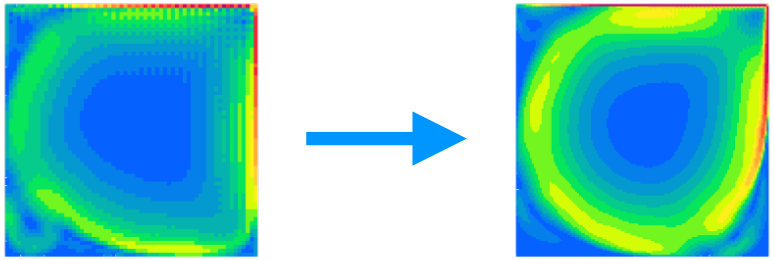
FluxGNN:
有限体積法ベースの
汎用的機械学習モデル




非線形マルチグリッド:
複数の解像度を協働させた
シミュレーション手法

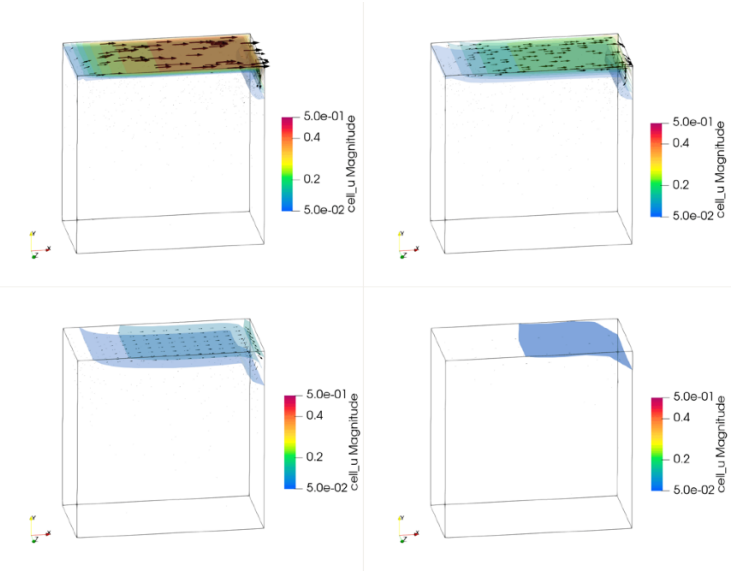



数値実験:
超解像シミュレーションと
Reynolds 数への汎化性能



# 研究の概要とまとめ

- 高 Reynolds 数 (～ $10^4$) 流れの予測に適した機械学習手法を構築した
  - 複数の解像度を協働させる非線形マルチグリッド法を
    導入することによりさまざまなスケールの相互作用を考慮
  - 任意の Reynolds 数に対して汎化性能を持つ機械学習モデルの手がかりを得た




非線形
マルチグリッド法