Machine learning for the interface reconstruction of two-phase flow simulation on the cooling of electric motors

T. Nakano *,1 , M. A. Bucci 1 , J-M. Gratien 2 , T. Faney 2 , B. Serigne Malick 1

1. TAU, INRIA et LISN, Université Paris-Saclay et CNRS, 91405 Orsay Cedex
2. IFPEN, Computer Science Department, 1 et 4 av Bois Préau, 92500 Rueil-Malmaison

(*) corresponding author: tamon.nakano@inria.fr

Keywords: Machine learning, Graph neural network, Interface reconstruction, Multiphase flow

Introduction

- A major bottleneck limiting the performance of electric motors is the cooling system.
- The coolant induces a complex multiphasic flow structure which can degrade the performance of the motor and can even be harmful to its structure.
- Reliable multiphase flow simulations are necessary to elucidate these phenomena and further to improve the cooling system.
- One way to understand those complex phenomena is the numerical simulation based on the Navier-Stokes equations.

Additional terms in the Navier-Stokes equations to deal with multiphase flows:

Momentum equation

$$\frac{\partial \rho \mathbf{U}}{\partial t} + \nabla \cdot (\rho \mathbf{U}\mathbf{U}) = -\nabla p + \rho \mathbf{g} + \nabla \cdot (\mu \nabla \mathbf{U}) + \rho \mathbf{S} + \sigma \kappa \hat{\mathbf{n}}_S \delta(\mathbf{x} - \mathbf{x}_s)$$

Advection, Pressure, Gravity, Viscosity, Extra forces, Surface tension force

Continuity equation

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{U}) = 0$$

Density

$$\rho(\mathbf{x}, t) = \begin{cases} \rho_A, & \text{if } \mathbf{x} \text{ in fluid A} \\ \rho_B, & \text{if } \mathbf{x} \text{ in fluid B} \end{cases}$$

Different densities

The surface tension depends on **local geometrical features** of the interface:

Note that:
$\kappa = \nabla \cdot \hat{\mathbf{n}}_S$

$$F_S = \sigma \kappa \hat{\mathbf{n}}_S \delta(\mathbf{x} - \mathbf{x}_s)$$

Surface tension coefficient
Constant and depends on the liquid/gas at play

Mean curvature
Local curvature of the interface:
$\frac{1}{2}\left(\frac{1}{R_1} + \frac{1}{R_2}\right)$

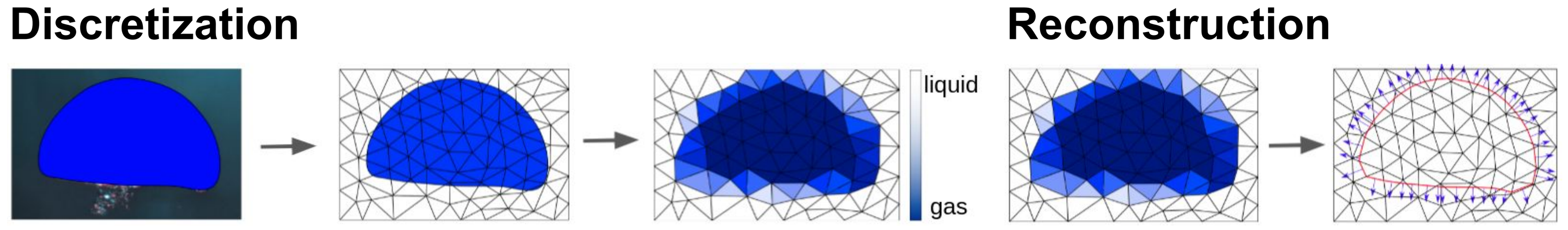
Normal
Normal direction to the interface in each point of the interface

kronecker delta
The surface tension is active only on the interface

The mean curvature and the normal vector to the surface are evaluated at each step of simulation. To do so, discretized surfaces in the simulation need to be reconstructed. In the finite volume method,

- Volume fraction, pressure, velocity, etc. are stored in grid cells.
- Geometrical information (normal, curvature etc.) will be lost after the discretization
- The volume of fluid (VoF) method is widely used to reconstruct the surface.
- Bottlenecks of the VoF are the cost of computation and a bad compatibility with the unstructured grid (while it's compatible with structured grid)
- Previous studies addressed this cost issue with machine learning based method (e.g., Qi *et.al.*[1], Patel *et al.*[2]), but they were based on structured grid.
- Graph Neural Networks (GNNs) are good candidates to handle this issue since they can consider graph-structured inputs such as elements on an unstructured grid.
- Our first attempt has been done with a dataset generated on spherical surfaces. A weakness related to its simple geometry was found. A dataset on more generalized geometries, that is, paraboloidal surfaces are currently being tested.

Discretization **Reconstruction**



Objective
We propose a machine learning-based method which is accurate enough to replace conventional reconstruction methods, less time-consuming and less resource-consuming. To that end, we study the use of GNNs (Graph Neural Networks) architectures to recover interface properties (e.g., normal vector, curvature, face center, etc.) from the discretized concentration field.

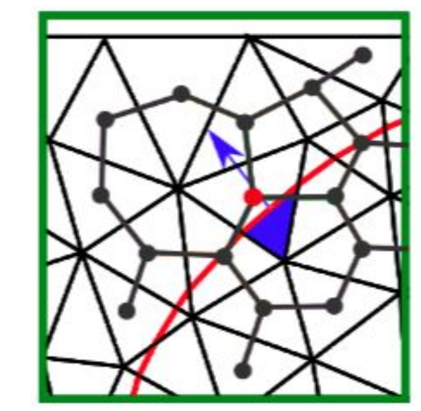
Methods

Our model based on GNNs learn,

- Features (volume fraction, volume of cells etc.) at all nodes
- Labels (normal, curvature and center) at the node of interest
- Edges: connections between nodes.

And it predicts the labels on a new (unseen) features

$$\hat{n}_i, H_i = f(\alpha_{ij})$$

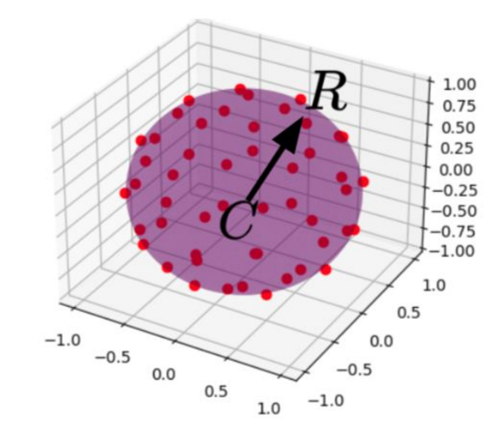


Schematic of a graph
red line: interface
red node: node of interest
black nodes: neighboring nodes

Dataset

Firstly, a synthetic dataset on a spherical surface was generated to the training and the validation.

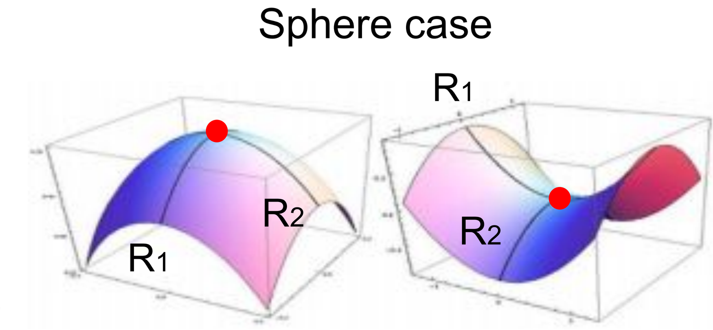
1. Create tetrahedral meshes around the red points in the figure
   - Center point = [0 + random[0,1]*Δx]
   - Uniform distribution: R/Δx = [10, 1000]
   - Grid elements: tetrahedron
   - Random curvature sign and random rotation
2. Collect data
   - Input: features in nodes, edges
   - Output: normal, curvature
3. Add random scaling to vary the curvature



Sphere case

As mentioned later, the result of spherical case showed the necessity of a more generalized geometry. Another dataset on paraboloidal surfaces are generated with the following conditions:

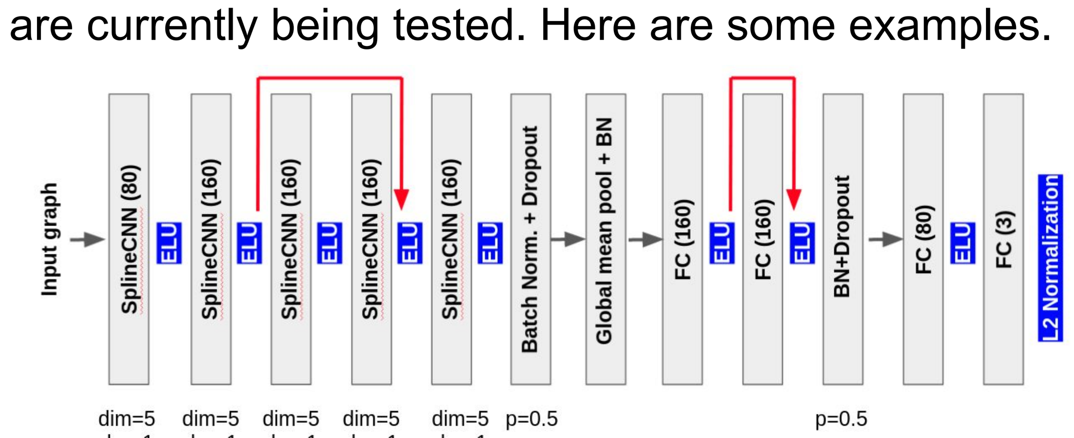
- 1 point at the center point (the red points in the figue), $R_{1,2}/\Delta x=\pm[10, 1000]$
- The others processes are same as the sphere dataset.



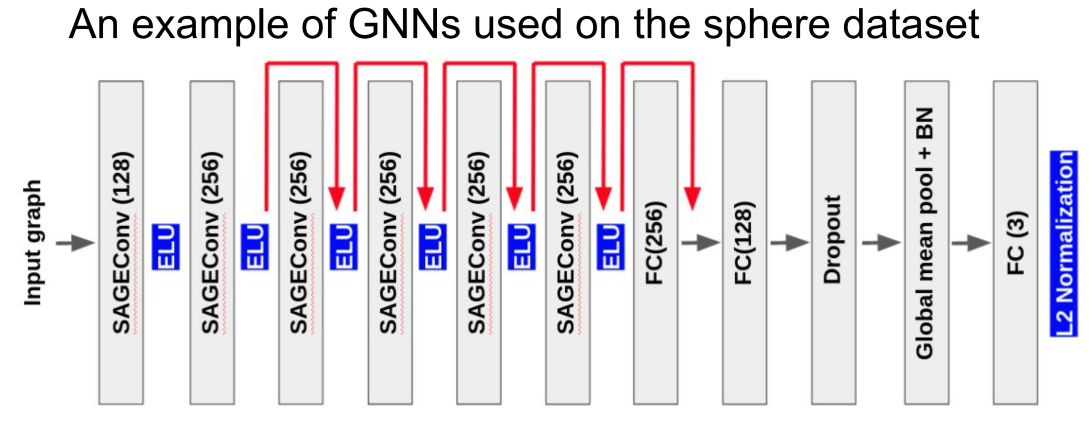
Paraboloid case[3]

GNN Model training

Different architectures based on convolutional GNNs are currently being tested. Here are some examples.



An example of GNNs used on the sphere dataset



An example of GNNs tested on the paraboloid dataset

MSE	Validation loss	Training loss	Batch size
Sphere	5.10e-4	3.53e-4	100
Paraboloid	5.42e-3	1.28e-4	1024

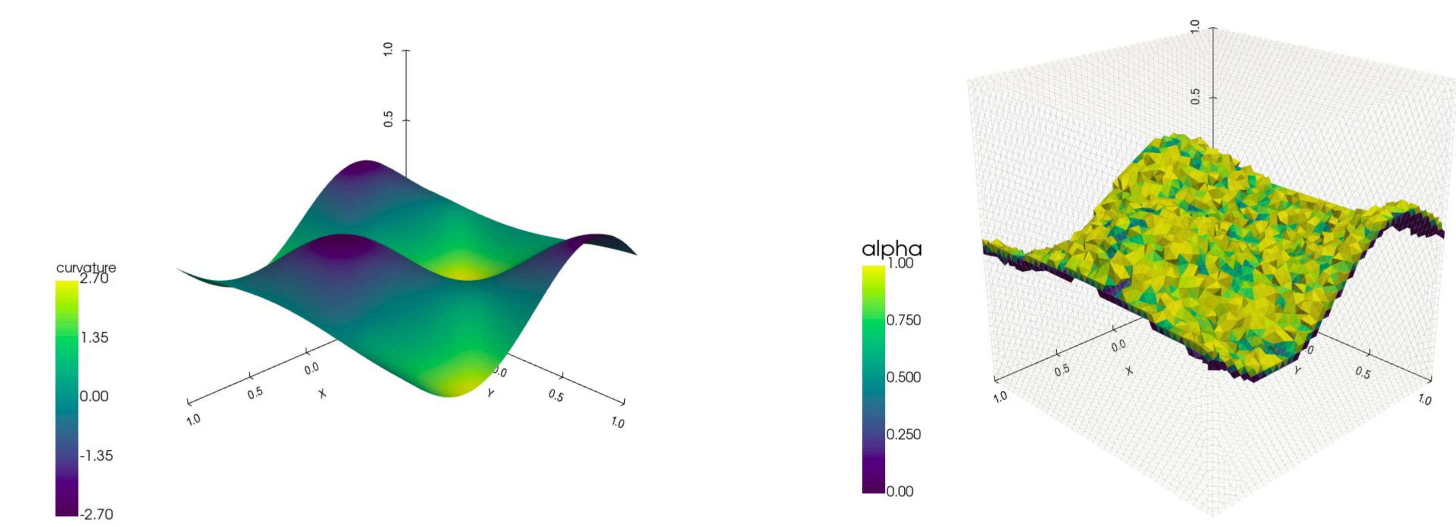
- Currently focusing on the normal prediction
- 100k graphs
- (Training, Validation)=8:2
- Mean Square Error
- ADAM optimizer

Generalization

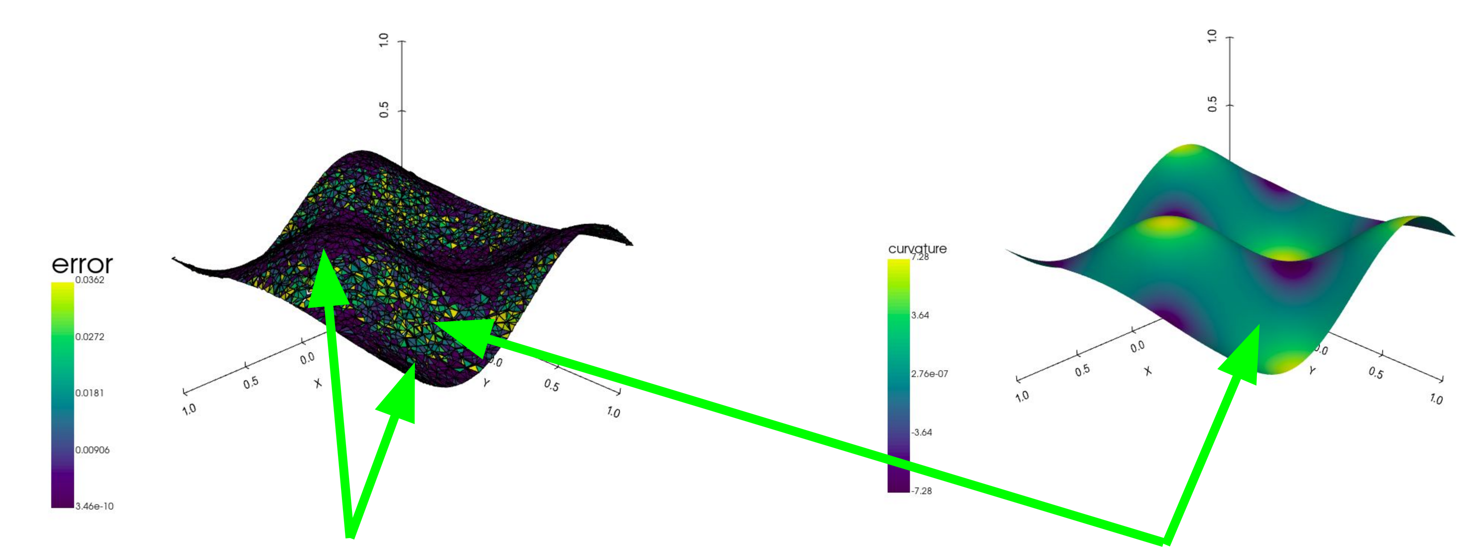
The trained model was tested on a sinusoidal surface which is unseen by the model in the training. A tetrahedral mesh was generated around the surface. At each voxel, the prediction was performed and compared to analytical solutions. This test has been done only with the model trained by the sphere case dataset so far. Only the prediction on the normal is discussed here.

Surface information:

- $f(x,y)=0.3\sin(3x)\cos(3y)$
- Test surface has R/Δx=8.33 at most, Training: R/Δx=[10, 1000]



Mean curvature Voxelized surface



Good prediction on convex/concave points but also on **saddle points**

High error (~10e-1) where **Gaussian curvature** equal to **zero**

Error on the normal by the GNN prediction Gaussian curvature $\kappa_g = 1/(R_1R_2)$

Conclusion & Perspective

Conclusion

- GNNs can be employed to predict the normal and the plane location, which can replace the conventional VoF method.
- The model generalises also beyond the training curvature range allowing coarser grid for the spatial discretization.
- Being trained on spherical shapes, the model has a weakness for the region where Gaussian curvature equal to zero.

Perspective

- Train the model on a new dataset which includes Gaussian curvature equal to zero.
  => Paraboloidal surfaces being tested at the present.
- Employ the GNN model in CFD code (i.e. OpenFOAM) to evaluate performances in a real multiphase flow simulation

[1]Qi, Yinghe, et al. "Computing curvature for volume of fluid methods using machine learning." Journal of Computational Physics 377 (2019): 155-161.
[2]Lörstad, Daniel, et al. "Assessment of volume of fluid and immersed boundary methods for droplet computations." International journal for numerical methods in fluids 46.2 (2004): 109-125.
[3] Connections between mathematics, bionic and architecture through minimal surfaces, Kalantari *et.al.,* Conference: 2nd international congress on science & engineering