ADVANCED FLUID SIMULATION USING GRAPH NEURAL NETWORKS

Enhancing Computational Fluid Dynamics with Deep Learning Techniques

[1]Umesh

[1]Senior Grade Lecturer
[1]Department of Science,
[1]Government polytechnic, Aurad (B), India.

Abstract:

Traditional computational fluid dynamics (CFD) relies on solving partial differential equations to compute flow field characteristics. However, this approach is computationally intensive and time-consuming. To address these limitations, we propose a fluid simulation system based on a graph neural network (GNN) framework. Our simulator offers high computational efficiency while significantly reducing resource consumption. By representing the computational domain as a structured graph, the model identifies neighboring nodes through adaptive sampling. We leverage deep learning techniques, incorporating attention-based graph neural networks, to enhance predictive accuracy. The simulator is trained on flow field data around a cylinder with varying Reynolds numbers. Once trained, it not only achieves high accuracy within the training set but also generalizes well to unseen flow conditions. Compared to conventional CFD solvers, our approach accelerates computations by 2-3 orders of magnitude. This advancement paves the way for faster optimization and design of fluid mechanics models, as well as real-time control of intelligent fluid systems.

Index Terms - ***Graph Neural Networks, Fluid Simulation, Computational Fluid Dynamics, Deep Learning, Flow Field Prediction, Adaptive Sampling.***

I. INTRODUCTION

Solving partial differential equations derived from physical models is a fundamental approach to understanding, simulating, and exploring real-world phenomena. Over the past century, advancements in solving these equations have transitioned from analytical approximations to numerical methods. Since the 1970s, Computational Fluid Dynamics (CFD), driven by high-precision numerical algorithms, has been instrumental in addressing complex nonlinear flow problems, including shock waves, material discontinuities, and turbulence. CFD allows researchers to analyze intricate fluid dynamics that are difficult to observe experimentally while providing quantitative predictions for fluid behavior. It has become an essential tool in various engineering applications.

Numerical methods in CFD convert differential governing equations into algebraic equations using discretization techniques such as finite difference and finite volume methods. The accuracy of these computations largely depends on the number of discrete nodes and their spatial relationships. A finer node distribution enhances accuracy, but also increases computational complexity.

In the past two decades, advances in hardware have significantly expanded CFD applications. However, relying solely on hardware improvements to enhance CFD performance has reached a plateau. Algorithmic innovations have stagnated, with the last major breakthrough, Total Variation Diminishing (TVD) methods, introduced in 1978. While the Essentially Non-Oscillatory (ENO) scheme has expanded interpolation capabilities, traditional finite-volume and finite-element algorithms have struggled to surpass second- or third-order accuracy. Meanwhile, meshless methods, such as particle algorithms, excel in handling high-speed solid collisions and crack propagation but lack mathematical proof of reliability.

From an application perspective, traditional CFD methods provide quantitative flow analysis through offline computations, aiding in system design, optimization, and control. However, emerging intelligent fluid engineering applications—such as biomimetic fluid power systems—require real-time processing and adaptation to complex, uncertain environments. Consequently, new techniques are needed to enhance real-time flow field prediction and adaptive control.

Traditional CFD relies on solving physical equations to uncover scientific principles, whereas machine learning enables direct data-driven analysis to extract meaningful patterns. By leveraging probability theory, statistical methods, and approximation techniques, machine learning can process vast datasets to identify correlations and make rapid predictions. This complements traditional CFD approaches and has the potential to surpass physics-based computational methods.

Since the early 2000s, machine learning—particularly deep learning—has gained traction in fluid simulation research. Some studies have trained neural networks using extensive observational data, incorporating physical properties through supervised learning. Ali Kashefi, for example, developed a deep learning model for flow field prediction in irregular domains using the PointNet architecture. However, this data-driven approach relies on costly experimental and large-scale simulation datasets.

To mitigate excessive data dependency and improve model interpretability, researchers have introduced inductive biases into neural networks. These biases embed domain-specific knowledge into the learning process, improving performance on fluid simulation tasks. Alvaro Sanchez-Gonzalez proposed a data-driven simulation framework, **Graph-based Network Simulator**, which integrates strong inductive biases by representing physical states as interacting particle graphs and approximating complex dynamics using multi-layer perceptrons (MLPs). Similarly, Julia Ling enhanced turbulence modeling accuracy by embedding Galilean invariance into network structures using an invariant tensor basis.

Another technique to improve neural network performance involves **learning bias**, where prior knowledge is incorporated through modifications to the loss function. M. Raissi introduced the **Hidden Fluid Mechanics model**, which penalizes deviations from simplified Navier-Stokes and transport equations to enforce physically consistent predictions. Observational bias, inductive bias, and learning bias can be applied separately or in combination, reinforcing each other when integrated.

While conventional deep learning models such as MLPs and CNNs struggle with irregularly structured data, researchers have turned to **graph theory and graph convolutional networks (GCNs)** to process non-Euclidean spatial structures. Graph convolutions introduce inductive biases tailored for structured data, making them well-suited for CFD applications. In engineering, fluid systems involve multi-physics and multi-scale dynamics, often requiring computational domains with unstructured meshes (e.g., airflow over an aircraft wing). Inspired by these challenges, we propose a **graph-based machine learning model** to handle high-dimensional, nonlinear flow field problems. By reformulating Euclidean space connections into graph-based representations, we can efficiently model complex fluid interactions.

Graph machine learning enables function definitions in non-Euclidean spaces, facilitating high-dimensional information transfer across computational domains. This approach reduces the complexity of nonlinear data relationships and enhances predictive accuracy. Our fluid simulation system employs a **graph attention network (GAT) architecture**, optimizing performance across various conditions and Reynolds numbers.

Key Innovations in Our Fluid Simulation Model:

1. **Enhanced Gradient Detection with Graph Attention Networks**
   - In fluid dynamics, regions with steep physical gradients are critical for accurate predictions. To better capture these gradients, our model utilizes a **graph attention mechanism** that assigns importance to specific nodes in the computational domain.
   - Since physical quantities in a flow field exhibit complex interdependencies, we employ a **multi-headed attention mechanism** to represent diverse relationships within different feature subspaces.
2. **Adaptive Sampling for Multi-Scale Flow Field Analysis**
   - Traditional CFD methods often use a fixed number of neighboring nodes for interpolation, limiting the ability to capture large-scale physical phenomena.
   - Our model introduces an **adaptive sampling mechanism**, dynamically selecting appropriate neighbor nodes based on scale-sensitive criteria. This allows the graph neural network to capture multi-scale physical information, improving accuracy in flow field predictions.
3. **Super-Resolution Training for Efficient Computation**
   - Fluid simulations involve vast spatial and temporal dimensions, making full-resolution simulations computationally expensive.
   - To improve efficiency, we implement a **sampling-aggregation mechanism** within the graph neural network, applying **super-resolution techniques**. This approach enables the model to be trained on low-resolution datasets while maintaining high-resolution predictive capabilities.

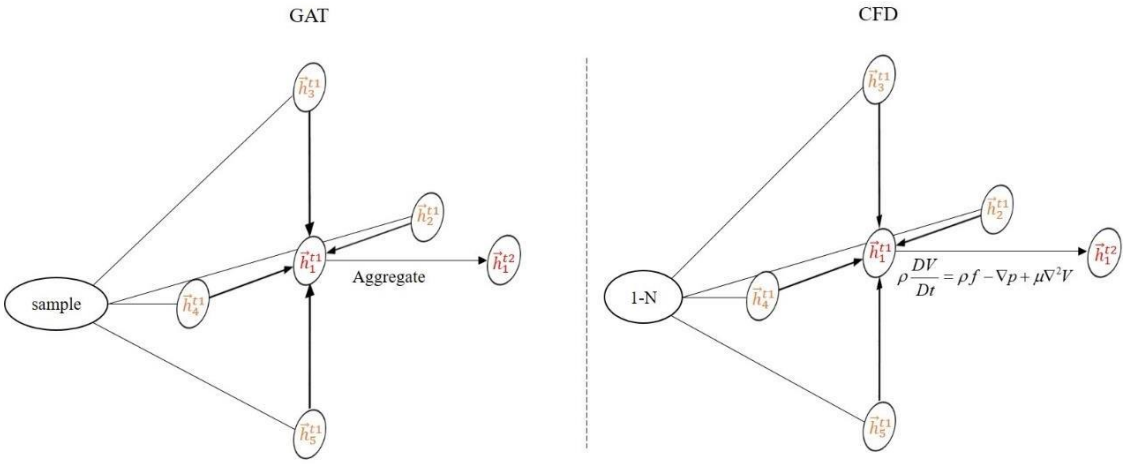
Through these innovations, our graph-based fluid simulation system significantly enhances computational efficiency and predictive accuracy, offering a transformative alternative to traditional CFD solvers.

II. THE NETWORK ARCHITECTURE

2.1 Connection between Neural Networks and Computational Fluid Dynamics

Computational fluid dynamics (CFD) relies on the Navier-Stokes equations to determine the physical properties of fluid flow, where variables exhibit strong interdependence. The nonlinear and complex coupling of these physical quantities makes solving the equations highly challenging.

According to the **Universal Approximation Theorem**, a sufficiently deep and wide neural network can approximate any continuous function. Graph Neural Networks (GNNs), with their intricate parameter structures, share fundamental similarities with CFD in terms of information aggregation. This inherent compatibility makes GNNs a promising approach for enhancing CFD models. The relationship between neural networks and computational fluid dynamics is illustrated below.



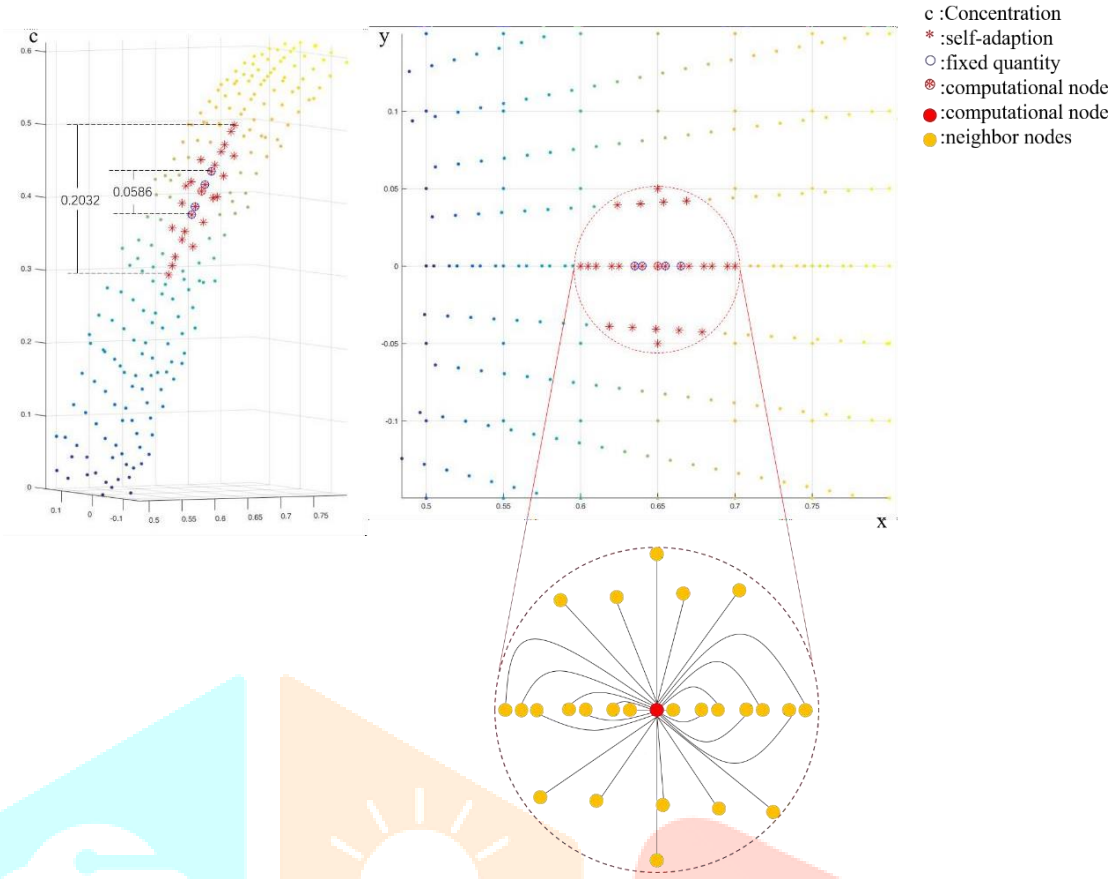
In Figure 1, the left section represents computational fluid dynamics (CFD), while the right section illustrates graph neural networks (GNNs). Both approaches operate on irregularly structured data. GNNs process local information using a sampling and aggregation mechanism, whereas CFD employs numerical methods to solve the Navier-Stokes equations and propagate physical information between nodes.

In this study, the graph neural network transmits messages by leveraging sampling and aggregation techniques. The key question is: **Which specific sampling and aggregation methods should be chosen?**

Fluid systems are inherently multi-physical and multi-scale in nature. In computational fluid mechanics, maintaining the balance of physical information exchange between nodes is crucial for improving calculation accuracy. To achieve this, a finer grid resolution is applied near walls or obstacles, where significant variations in gradients occur, compared to stable flow regions.

If first-order neighboring nodes or a fixed number of neighbors are selected, the computed gradient difference between a node and its surroundings becomes overly uniform. This prevents the neural network from capturing sharp gradient variations in those regions, leading to instability in flow field calculations where gradients change rapidly. To address this, we propose an **adaptive sampling strategy** that selects neighboring nodes dynamically based on gradient variation patterns.

By gathering multi-scale gradient information and sampling a greater number of neighboring nodes in regions with significant gradient changes, we ensure that finer details are captured without losing global information. **Figure X** compares adaptive sampling with fixed neighbor node selection, demonstrating how adaptive sampling better preserves complex gradient variations in the flow field.



Node Representation

Using the example of fluid flow around a cylinder, we apply a fully supervised learning approach to train the model. This enables the network to learn the physical properties of the flow field over successive time steps, ultimately allowing for end-to-end prediction of physical information. The relationship between nodes is established in the **spatiotemporal domain**, with the coordinate information of each node implicitly embedded into the input vector.

Each node's input state vector consists of **concentration, pressure, velocity in the X-direction, and velocity in the Y-direction** over six consecutive time steps. The input vector for a single node is represented as:

xt1−t6=[ct1−t6,pt1−t6,ut1−t6,vt1−t6]x_{t1 - t6} = [c_{t1 - t6}, p_{t1 - t6}, u_{t1 - t6}, v_{t1 - t6}]

The overall input vector is:

xt1∼t6=[ct1∼t6,pt1∼t6,ut1∼t6,vt1∼t6,e]x_{t1 \sim t6} = [c_{t1 \sim t6}, p_{t1 \sim t6}, u_{t1 \sim t6}, v_{t1 \sim t6}, e]

where:

- **e** represents the connection between nodes,
- **c** denotes concentration,
- **p** represents pressure,
- **u** is velocity along the X-axis,
- **v** is velocity along the Y-axis.

Similarly, the output vector for an individual node at the next time step t7t7 is:

yt7=[ct7,pt7,ut7,vt7]y_{t7} = [c_{t7}, p_{t7}, u_{t7}, v_{t7}]

The overall output vector is:

yt7=[ct7,pt7,ut7,vt7,e]y_{t7} = [c_{t7}, p_{t7}, u_{t7}, v_{t7}, e]

In the simulation model, the input vector xt1∼t6x_{t1 \sim t6} is used to predict the output vector at t7t7. The simulation function is expressed as:

yt7=f(xt1−t6,∑j∈ne[i]xt1−t6,eij,ω)y_{t7} = f (x_{t1 - t6}, \sum_{j \in ne[i]} x_{t1 - t6}, e_{ij}, \omega)

Where:

- **ne[i]** represents the set of neighboring nodes of node **i**,
- **e_ij** is the connection edge between nodes **i** and **j**,
- **ω** denotes the learnable parameters of the model.

This formulation enables the model to capture spatiotemporal dependencies in the flow field, facilitating accurate prediction of fluid dynamics.

Network Architecture

The parameterized function f(ω)f(\omega) is responsible for computing the physical properties of the fluid, mapping input xx to output yy. To design the network structure, we employ a **graph attention layer**, and the complete architecture is illustrated below.

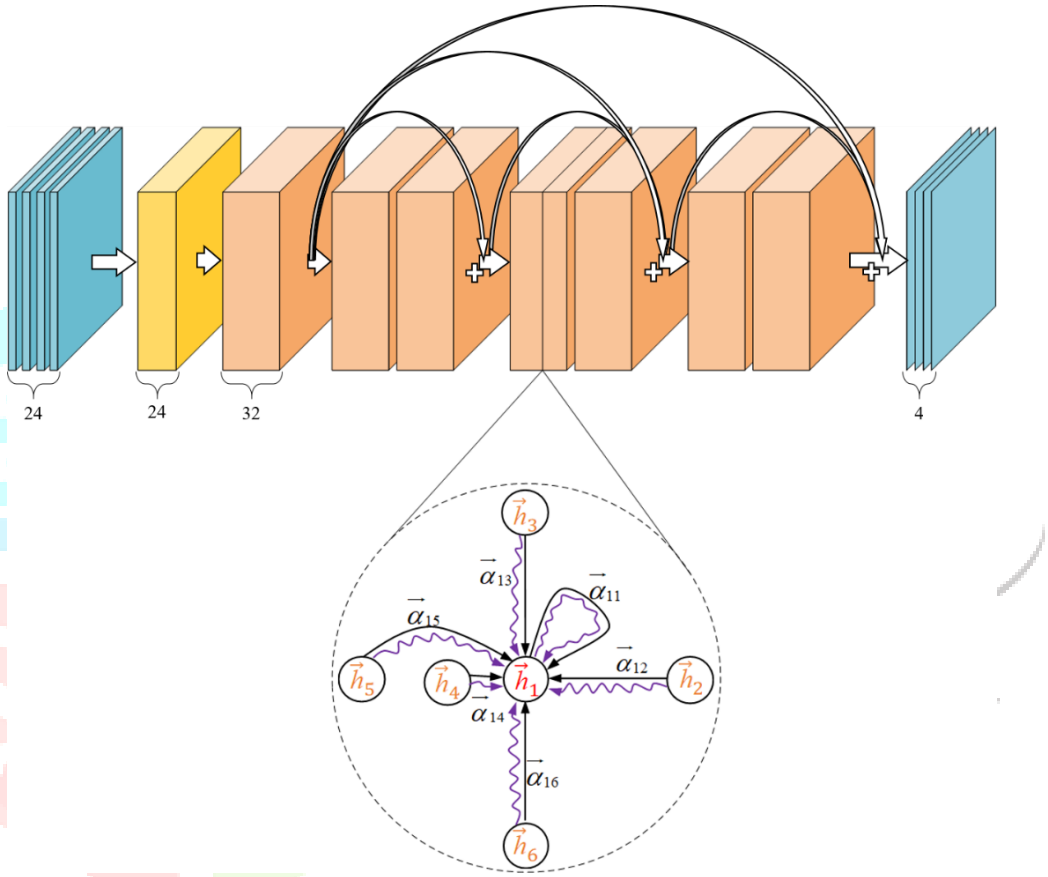


Figure 3: Graph Attention Network-Based Architecture

The network consists of **single-headed attention layers** and **multi-headed attention layers**. The attention mechanism assigns a weight matrix to capture relationships between computational nodes and their neighboring nodes while aggregating physical data from the neighborhood. As depicted in the figure, physical quantities such as **pressure, velocity, and density** are highly interdependent in fluid dynamics.

The input and output of our model include **concentration, pressure, and velocity**, which align with the nonlinear interactions described by the **Navier-Stokes equations**. The attention coefficient's parameter matrix effectively models the nonlinear correlations between node features. The attention weights between nodes are determined by the following equation:

q=∑j∈ne[i]kiq = \sum_{j \in ne[i]} k_i vi=v_i = αij=exp⁡(s(σ(ki),σ(qj))/d)∑k∈ne[i]exp⁡(s(σ(ki),σ(qk))/d)\alpha_{ij} = \frac{\exp(s(\sigma(k_i), \sigma(q_j)) / d)}{\sum_{k \in ne[i]} \exp(s(\sigma(k_i), \sigma(q_k)) / d)}

where:

- Wq,Wk,WvW_q, W_k, W_v are linear transformation matrices corresponding to the **query vector, key vector, and value vector**, respectively.
- ne[i]ne[i] represents the set of neighboring nodes of node **i**.
- dd is the dimension of the query and key vectors.
- ss denotes the **dot product function**.
- σ\sigma is the **LeakyReLU activation function**.

- αij\alpha_{ij} represents the **attention coefficient**, determining the weight between nodes.

Since our model incorporates **multiple inputs and multiple outputs**, there exist **complex interdependencies** between them. To better capture these relationships, we integrate **single-headed and multi-headed attention mechanisms** within the network architecture.

- **Single-headed attention** focuses on learning feature dependencies between nodes.
- **Multi-headed attention** provides a more comprehensive representation by capturing diverse node relationships.

A visualization of the multi-head attention mechanism is provided in the following section.

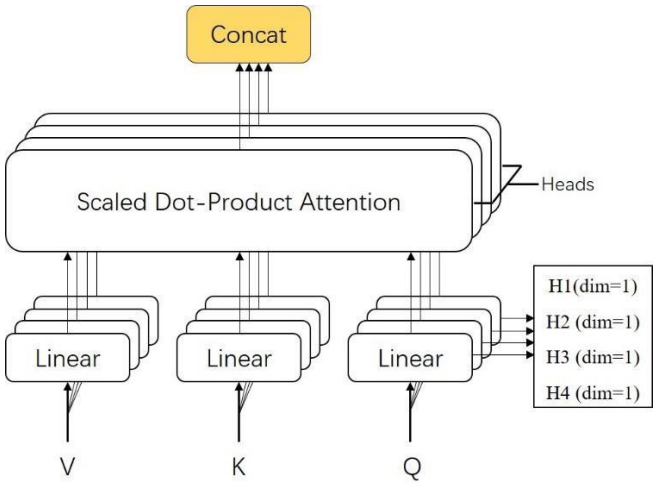


Figure 4: Multi-head attention mechanism

The **multi-headed attention mechanism** distributes the **V (value), K (key), and Q (query) vectors** into four distinct **representation subspaces**, where the dimensionality of the **K and Q subspaces** is set to **1**.

To enhance the **network's complexity**, two primary approaches are considered:

1. **Increasing the network width** – Expanding the dimensionality of each network layer.
2. **Deepening the network** – Adding more layers to increase depth.

Additionally, a **k-layer graph convolutional neural network (GCN)** enables the integration of information from **k-order neighboring nodes**. This deeper architecture provides a **broader receptive field**, allowing the model to capture **long-range dependencies** and more distant **physical information** in the flow field.

The figure below illustrates how a **two-layer graph neural network** aggregates information from neighboring nodes.

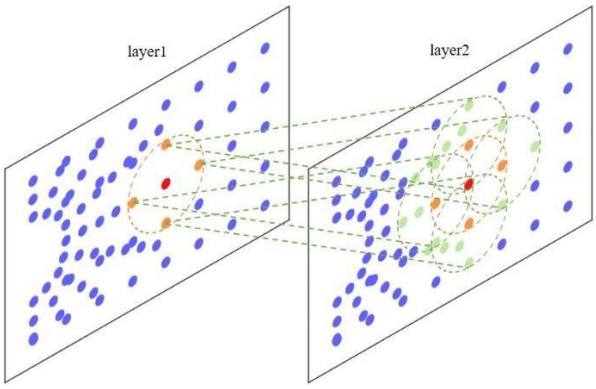


Figure 5: The receptive field of the network continues to expand through the superposition of multiple layers.

III. RESULTS

To mitigate the issue of gradient explosion, we incorporated a Batch Normalization (BN) layer before the attention layer. Additionally, we set the dropout rate to 0.6 and the learning rate to 0.001 between layers. The mean squared error (MSE) was chosen as the loss function for model optimization.

For **2D flow field computations**, the hidden layer dimension was configured to 32, while the output layer had a dimension of 4. In contrast, for **3D flow field computations**, the hidden layer dimension was increased to 64, and the output layer dimension was set to 5.

The network architecture comprises a total of **eight graph attention layers**, each with specific configurations. The number of attention heads per layer, along with the dimensional settings and activation functions for the Q and K vectors, is detailed in the following table.

Layer	Heads	Dimensionality	Activation Function
1	4	4	Leaky_ReLU
2	1	16	Leaky_ReLU
3	4	4	-
4	1	16	Leaky_ReLU
5	4	4	-
6	1	16	Leaky_ReLU
7	4	4	-
8	1	1	-

Table 1: Network Architecture Details

Our model was implemented using **TensorFlow**, **tf_Geometric22**, and **SKLearn**. The hardware setup consisted of an **NVIDIA TITAN RTX 2070 GPU** and an **Intel(R) Xeon(R) CPU E5-2660 V4 @ 2.00GHz**.

To evaluate performance, we compared the accuracy of various network architectures using **2D circular cylinder flow data** under conditions of **Re = 100 and Pec = 100**. Additionally, we tested **3D circular cylinder flow data** for **Re = 100 and Pe = 100**. Further assessments were conducted on:

- **2D circular cylinder flow data** at **Re = 200 and Pec = 2000** under **Neumann boundary conditions**
- **2D circular cylinder flow data** at **Re = 200 and Pec = 2000** under **Dirichlet boundary conditions**
- **3D aneurysm flow data** at **Pec = 1.0/0.0101822**
- **Flow data with Re = 5 and Pec = 15**, used to evaluate the generalization capability of the model

The dataset utilized for these experiments was sourced from **Brown University's Department of Applied Mathematics**.

Fluid Simulation Simulator

The design principles and network structure of the **fluid simulation simulator** are detailed in **Section 2**. Through experimental analysis, we validate the simulator's ability to predict fluid dynamics and assess how different network configurations impact the accuracy of the results.

The **fluid simulation model** predicts the flow of fluid data by extrapolating physical quantities from six consecutive time steps to determine the **seventh moment**, as illustrated in the figure below.

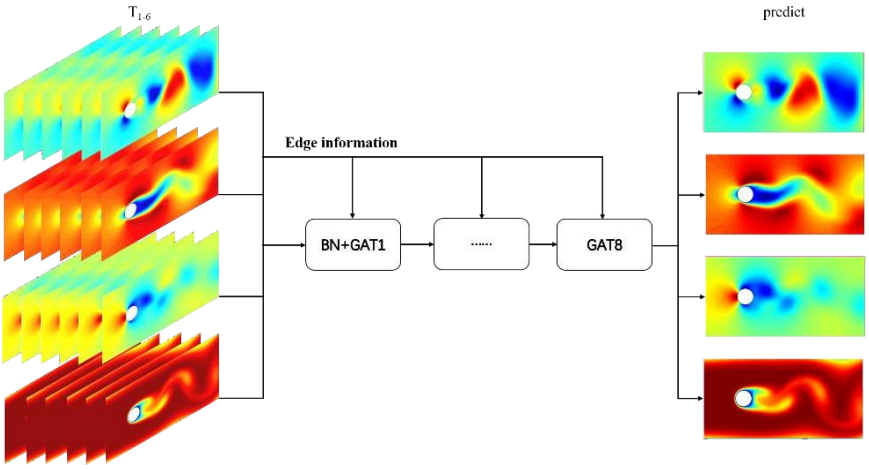


Figure 6: The simulator predicts the physical properties at time step $\mathbf{t_7}$ based on the physical characteristics observed from $\mathbf{t_1}$ **to** $\mathbf{t_6}$.

Computational Accuracy Analysis of Different Network Architectures

We evaluate the **computational accuracy** of various network architectures by measuring their **relative error**. The figure above illustrates the **prediction accuracy** of different models in simulating the flow field around a **cylinder** over a **16-second duration**.

A comparison of the results in the table highlights the **superior accuracy** of our proposed **network architecture** compared to other models.

- **GraphSAGE**, which uses **average aggregation** without an **attention mechanism**, performs the worst.
- **Sampling strategy:** The performance of models using a **fixed number of neighboring nodes** is **inferior** to those that dynamically adapt to their neighbors.
- **Network architecture:** Models **without residual connections** and those using a **single-headed attention mechanism** show **lower accuracy** than our approach. This confirms that incorporating **residual connections** and a **multi-headed attention mechanism** significantly enhances performance.

In comparison with **HFM**, our model demonstrates **higher accuracy** in predicting **velocity and pressure**, although **concentration prediction** is slightly less accurate. Additionally, **GCN models** incorporating **residual connections** and **adaptive sampling** also yield promising results.

Methods	L2 relative error			
	C	U	V	P
Ours	0.0223	0.0129	0.0128	0.0148
No Residual	0.0291	0.0194	0.0251	0.0215
Self-Attention	1.1823	0.7319	0.3870	0.1351
Fix-Point	0.0376	0.0215	0.0208	0.0184
GCN	0.0507	0.0404	0.0349	0.0430
GraphSAGE	4.7874	0.0606	1.0016	1.0000
HFM	0.0153	0.0680	0.0701	0.0987

IV. CONCLUSION

In this study, we developed a **fluid simulation simulator** incorporating **inductive bias**, utilizing a **graph attention neural network** to construct the model. For **multi-physical and multi-scale** fluid systems, we applied **adaptive sampling** and **attention mechanisms** to efficiently aggregate physical information from both **compute nodes** and **neighboring nodes**. Experimental comparisons between **adaptive sampling** and a **fixed-neighbor sampling approach** demonstrate that **adaptive sampling** yields **superior performance**.

To assess the significance of the **attention mechanism**, we compared **average aggregation** with **attention-based aggregation** in **cylindrical flow experiments**. The results indicate that **attention-based aggregation** significantly outperforms average aggregation in capturing **physical interactions** between adjacent nodes. Given the **complex relationships** between nodes in fluid dynamics, we employed a **multi-head attention mechanism** to learn **node interactions** across different **representation subspaces**, enabling a more **comprehensive description** of the physical relationships between adjacent nodes.

As the **graph neural network** deepens, it gains the ability to extract **broader-scale physical information**, thereby enhancing its **receptive field**. To achieve this, we implemented an **8-layer graph attention network**, which was experimentally verified to deliver the **highest prediction accuracy**. However, deeper networks may lose some **shallow-level information** during information transfer. To mitigate this, we incorporated **residual connections**, ensuring that critical **shallow information** is effectively propagated to deeper layers, ultimately improving prediction accuracy. Experimental results confirm that our **network architecture** surpasses **GCN layers** in predicting **concentration, pressure, and velocity**. When compared with **HFM**, our model excels in **pressure and velocity predictions** but falls slightly behind in **concentration prediction**.

We also validated the **generalization ability** of our **fluid simulation simulator** under various conditions, including **Dirichlet and Newman boundary conditions** in **2D flow around a cylinder**, **2D half-cylinder flow fields**, **3D cylinder flow fields**, and **3D aneurysm flow fields**. **Time-series analysis** of prediction accuracy revealed a **negative correlation** between **flow field gradients** and **prediction accuracy**—indicating that higher gradients result in lower prediction accuracy, aligning with conventional **computational fluid dynamics** principles.

Currently, our fluid simulation simulator integrates **inductive bias**, and in future work, we aim to incorporate **learning bias** to further **enhance prediction accuracy**, **reduce training time**, and **improve the interpretability** of the simulator.

REFERENCES

1. **Mi, Y., Ishii, M., & Tsoukalas, L. H. (2001).** Flow regime identification methodology with neural networks and two-phase flow models. *Nuclear Engineering and Design, 204*(1-3), 87–100.
2. **Noack, B. R., Afanasiev, K., Morzynski, M., Tadmor, G., & Thiele, F. (2003).** A hierarchy of low-dimensional models for the transient and post-transient cylinder wake. *Journal of Fluid Mechanics, 497*, 335–363.
3. **Monaghan, J. J. (1988).** An introduction to SPH. *Computer Physics Communications, 48*(1), 89–96.
4. **Müller, M., Charypar, D., & Gross, M. (2003).** Particle-based fluid simulation for interactive applications. *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 154–159.
5. **Koshizuka, S., & Oka, Y. (1996).** Moving-particle semi-implicit method for fragmentation of incompressible fluid. *Nuclear Science and Engineering, 123*(3), 421–434.
6. **Becker, M., & Teschner, M. (2007).** Weakly compressible SPH for free surface flows. *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 209–217.
7. **Solenthaler, B., & Pajarola, R. (2009).** Predictive-corrective incompressible SPH. *ACM Transactions on Graphics, 28*(3), 40.
8. **Bender, J., & Koschier, D. (2015).** Divergence-free SPH for incompressible and viscous fluids. *IEEE Transactions on Visualization and Computer Graphics, 21*(3), 248–262.
9. **Smagorinsky, J. (1963).** General circulation experiments with the primitive equations: I. The basic experiment. *Monthly Weather Review, 91*(3), 99–164.
10. **Hamilton, W. L., Ying, R., & Leskovec, J. (2015).** Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems, 30*, 1024–1034.