FYP Final Presentation

LIU, Ran; XIAO, Ziruo; ZENG, Lingqi; ZHANG, Rushan

Hong Kong University of Science and Technology

April 27, 2024

Table of Contents

Introduction

Overview

Task: Flow Simulation

- The prediction of velocity field and pressure field of flow around an object

Applications

- Weather forecasting
- Aircraft design
- Building design

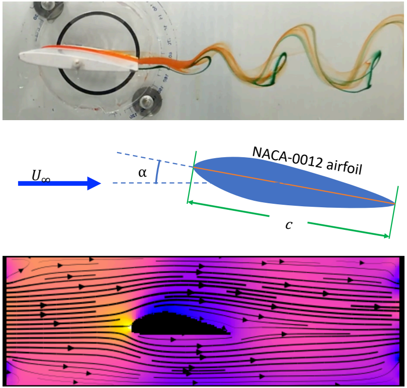


Figure: An example fluid simulation problem

Modelling of a fluid problem

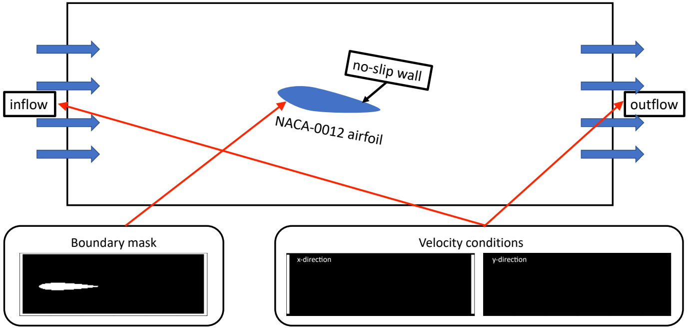


Figure: Schematic of a case setup

Governing Equations

Conservation of Momentum

$$\rho \dot{\mathbf{u}} = \rho(\partial_t \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u})$$

Conservation of Mass

$$\Delta \cdot \mathbf{u} = 0$$

Boundary Conditions

No-slip condition

$$\mathbf{u} = \mathbf{u}_{boundary}$$

Numerical Simulation

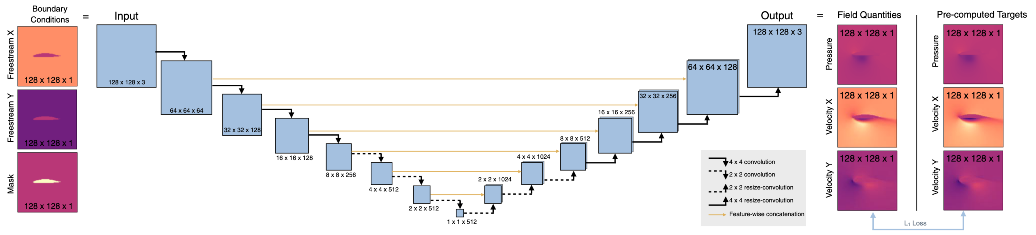
OpenFOAM

Ansys Fluent

SU2

Data Driven Neural Network

Deep learning methods for reynolds-averaged navier–stokes simulations of airfoil flows (Thuerey et al., 2020)



Physics Informed Neural Network

Learning Incompressible Fluid Dynamics from Scratch – Towards Fast, Differentiable Fluid Models that Generalize (Wandel et al., 2020)

- Physical Equations
  - Conservation of Momentum

$$\rho \dot{\mathbf{u}} = \mu \nabla^2 \mathbf{u} - \nabla p + \mathbf{f}$$

  - No-slip Condition

$$\mathbf{u} = \mathbf{u}_{boundary}$$

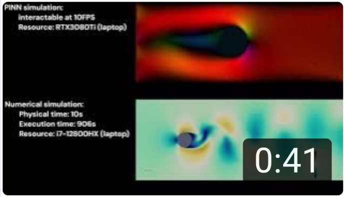
- PINN Loss

$$L_{PINN} = \alpha_{fluid} \sum_{fluid} ||\rho \dot{\mathbf{u}} - \mu \nabla^2 \mathbf{u} + \nabla p||^2 + \alpha_{boundary} \sum_{boundary} ||\mathbf{u} - \mathbf{u}_{boundary}||^2$$

Pros and Cons of Typical Methods for Simulation

- **Numerical Simulation**
  - **Pros** well-studied
  - **Cons** computationally costly
- **Data Driven Neural Networks**
  - **Pros** rely on large amount of expensive data
  - **Cons** high online inference speed
- **Physics Informed Neural Networks**
  - **Pros** relieve the dependence on data
  - **Cons** potential challenge with generalizability

Video Demonstration



Introduction

Literature Survey

Definition of a fluid problem

The principle of training

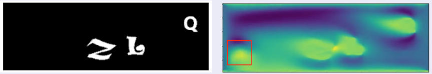
- Exposing the PINN simulator to a wide range of fluid problems

Vanilla implementation

- Training the PINN simulator with simple fluid problems

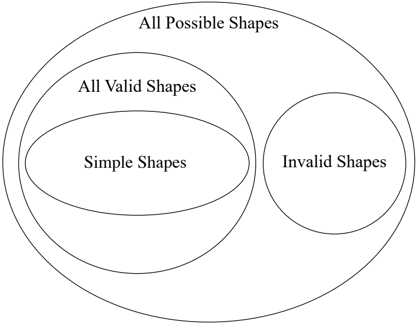
Simple problems used by Baseline

Baseline on complex problems



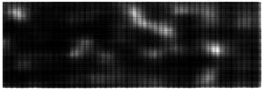
Cases unseen in fluid problems for training

- Miscellaneous objects in the flow field
- Indicating the lack of generalizability



(a) All Valid Shapes

(b) Simple Shapes



(c) Invalid Shapes

Figure: Left: Schematics of the Problem Space; Right: Samples in the three problem spaces. Object shapes are used as examples. A fluid problem also involves other parameters like inflow velocity.

A fluid problem is defined by

- Boundary shapes
- Velocity condition on the boundaries

Our idea

- Designing a closed-loop training process to fully explore the problem space

- Objectives
  - **Generalizability.** Develop an unsupervised method that comprehensively explores the problem space to enhance the sample space's completeness.
  - **Accuracy.** Guarantee the generalizability of the PINN solver is improved without the expense of the PINN solver's prediction accuracy on the common problems.
  - **Efficiency.** Optimize the training pipeline such that the model is able to more effectively explore the problem space.

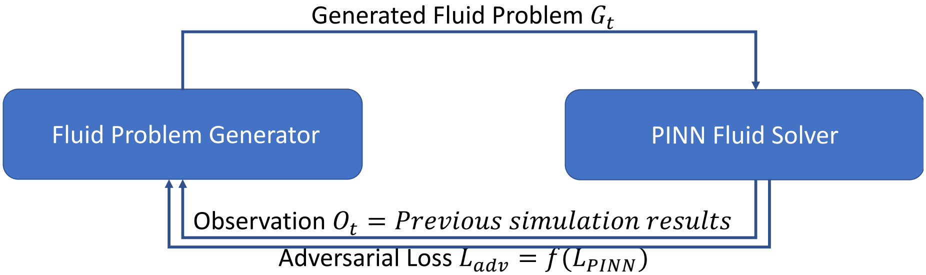
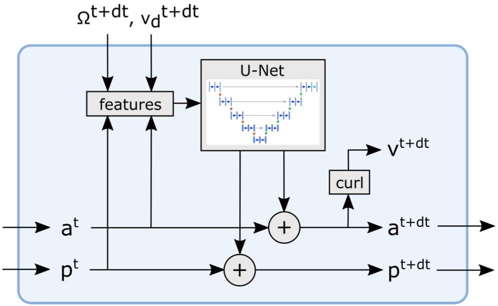


Figure: Overview of the proposed method.

Methodology

PINN Fluid Solver



- Flow Field
  - *a*, 2D tensor, velocity field
  - *p*, 2D tensor, pressure field
- Boundary Conditions
  - Ω, 2D tensor, the shape of boundaries as binary mask
  - v_d, 2D tensor, velocity condition on the boundaries

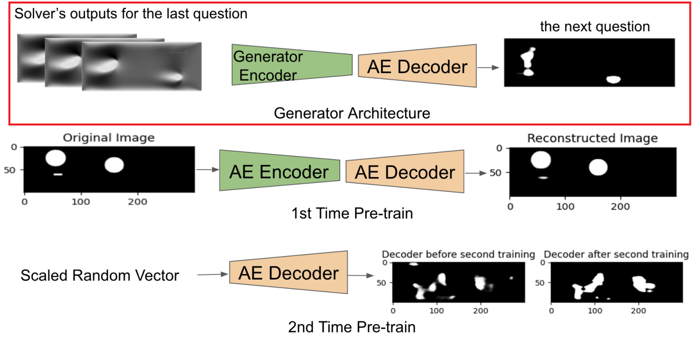


Figure: Generator Design and its Pre-training Pipeline.

Generator structure

- Inputs:
  - 32 consecutive frames of the solver's output, i.e., velocity (v_x and v_y) and pressure fields (p) generated by the solver in a 32 consecutive time steps.
- Outputs:
  - A newly generated problem, i.e., mask and condition velocity.

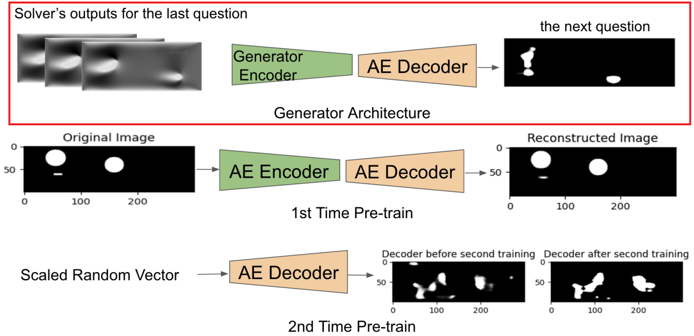


Figure: Generator Design and its Pre-training Pipeline.

Generator structure

- Generative encoder:
  - Several encoder-like convolutional layers.
  - Generates a latent vector.
- Pretrained decoder:
  - Translates the latent vector into a rational CFD problem.
  - Freezed during the adversarial training process.

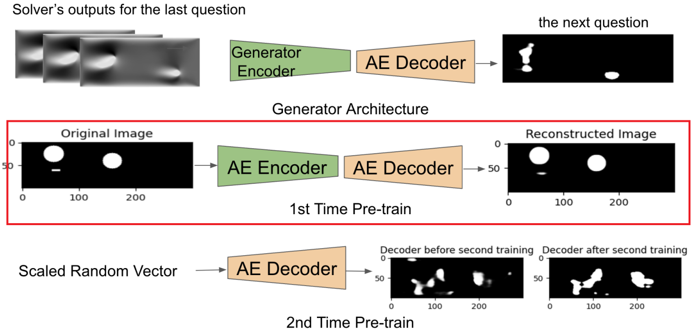


Figure: Generator Design and its Pre-training Pipeline.

Generator pretraining

- Training decoder.
- 1^{st} pretraining: Reconstruction
  - Train an autoencoder.
  - Extract the latent vector (encoder's output) distribution, i.e., mean and variance.

Methodology

Fluid Problem Generator

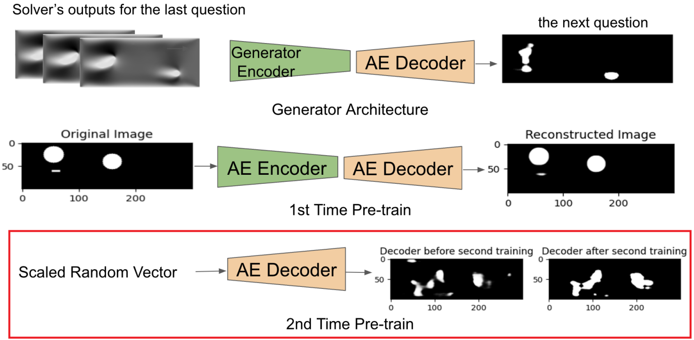


Figure: Generator Design and its Pre-training Pipeline.

Generator pretraining

- Training decoder.
- 2^{nd} pretraining: Sharpening
  - Train the decoder D by taking a scaled random vector.
  - Using customized loss:

$$X = D(v)$$

$$X_{\text{original}} = D_{\text{original}}(v)$$

$$L = \text{mean}\left((X(1-X))^2 + \text{mean}(X - X_{\text{original}})^2\right.,$$

where v is a random vector.

Methodology

Training Pipeline

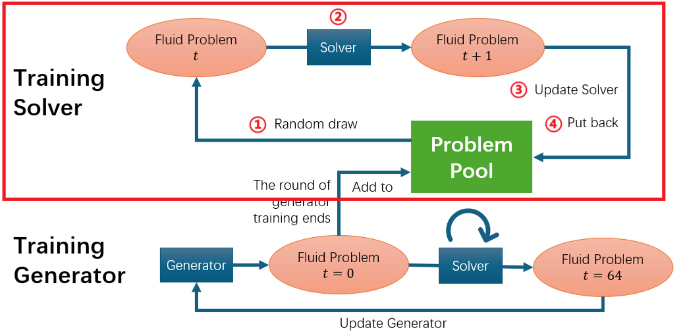


Figure: Training workflow.

- The generator and solver are iteratively trained.
- Problem pool:
  - Stores all generated problems.
- Training solver:
  1. Pick a random problem from problem pool.
  2. The solver predicts the next time frame for the problem.
  3. Calculate PINN loss and update solver.
  4. Update the problem.

Methodology

Training Pipeline

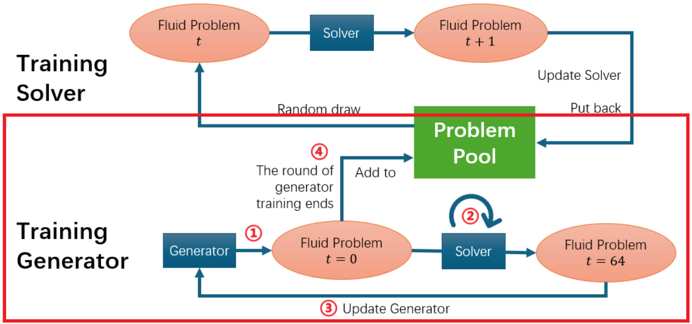


Figure: Training workflow.

- The generator and solver are iteratively trained.
- Training generator:
  1. Generator generates a batch of problems.
  2. Solver predicts 64 consecutive time frames.
  3. Calculate and negate PINN loss, add rationality loss and update generator.
  4. In the last iteration, push new problems to the problem pool.

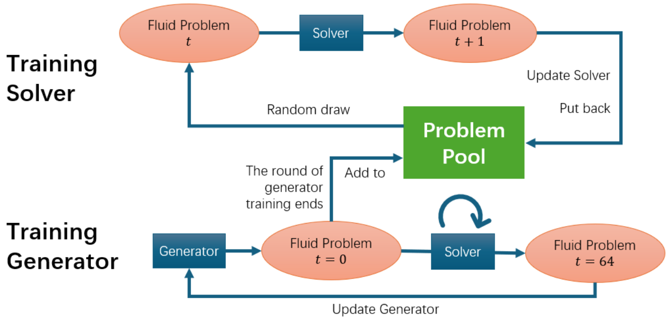


Figure: Training workflow.

- Generator reinitialization:
  - The generator is reinitialized periodically to add diversity.
- Problem renewal:
  - Problems in the problem pool are randomly renewed to initial state.

Methodology

Training Pipeline

Algorithm 1 `TrainAL-PINN`

```
Initialize problem pool P, setting the timestamp for each problem to 0
Initialize generator G
Initialize solver S
for epoch = 1 to 200 do
    Reinitialize G
    Renew a random subset of P
    for i = 1 to 5000 do
        j = Random index from 1 to len(P)
        t ← p_j.timestemp
        p_j ← S(p_j)
        p_j.timestemp ← t + 1
        Compute loss and update S
        if UpdateConditionSatisfied() then
            for k = 1 to 10 do
                G generates a batch of problems
                S solves the batch of problems from time step 0 to 64
                Calculate and negate loss
                Update G
            end for
            G generates a new batch of problems
            Append new problems to P
        end if
    end for
end for
```

Evaluation

Method Overview

Data for evaluation

- Ground truth data: **overfitted** results of the PINN solver for each case.
- Inferred data: the solver infers 64 time steps, and the **latter 32** time steps are used.

Metrics

- Peak Signal-to-Noise Ratio (PSNR)
- Structural Similarity Index Measure (SSIM)
- Frequency and Magnitude Difference (FMD)

Questions

- Cylinder
- Airfoil
- Complex shapes

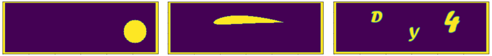


Figure: Examples of validation dataset.

Evaluation

Validation Datasets

Figure: Examples of validation dataset. Left: **Cylinder**; Middle: **Airfoil**; Right: **Complex shapes**.

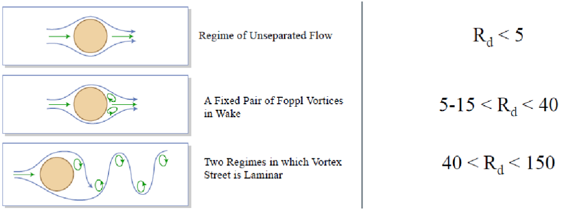


Figure: Flow topology behind a cylinder under different Reynolds numbers.

Equations for airfoil generation

$$y_t = 5t\Big[0.2969\sqrt{x} - 0.1260x - 0.3516x^2 + 0.2843x^3 - 0.1015x^4\Big]$$

Peak Signal-to-Noise Ratio (PSNR)

$$\mathrm{MSE} = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2$$

$$\mathrm{PSNR} = 10 \cdot \log_{10}\left(\frac{\mathrm{MAX}_I^2}{\mathrm{MSE}}\right)$$

Structural Similarity Index Measure (SSIM)

$$\mathrm{SSIM}(I, K) = \frac{(2\mu_I\mu_K + C_1)(2\sigma_{IK} + C_2)}{(\mu_I^2 + \mu_K^2 + C_1)(\sigma_I^2 + \sigma_K^2 + C_2)}$$

$$C_1 = (K_1 L)^2, C_2 = (K_2 L)^2$$

Frequency and Magnitude Difference (FMD)

$$St = \frac{f \cdot D}{U}$$

$$\mathrm{FMD} = |f_i - f_{\mathrm{ref}}| + |m_i - m_{\mathrm{ref}}|$$

PSNR, SSIM vs FMD

- Range
- Structural information, luminance, and contrast.
- Periodic information.

	Model	PSNR	SSIM	FMD
Cylinder	**Baseline**	80.184	0.660	2.857
	Ours	77.680	0.541	2.927
Airfoil	**Baseline**	83.419	0.763	3.349
	Ours	75.433	0.569	3.334
Complex	**Baseline**	60.972	0.333	0.175
	Ours	64.651	0.505	0.226

Table: Comparison of Different Metrics Across Categories

Scores

Simple cases:

- PSNR and SSIM: The baseline performs better.
- FMD: Mixed performance.

Complex cases:

- PSNR and SSIM: Ours performs better.
- FMD: Small for both models. Not significant.
  - Turbulent state. No significant frequency response.

Evaluation

Quantitative Result

	Model	PSNR	SSIM	FMD
Cylinder	**Baseline**	80.184	0.660	2.857
	Ours	77.680	0.541	2.927
Airfoil	**Baseline**	83.419	0.763	3.349
	Ours	75.433	0.569	3.334
Complex	**Baseline**	60.972	0.333	0.175
	Ours	64.651	0.505	0.226

Table: Comparison of Different Metrics Across Categories

Discussions

- Our model has better generalizability.
- Baseline is trained with simpler problems, and there might be overfitting.

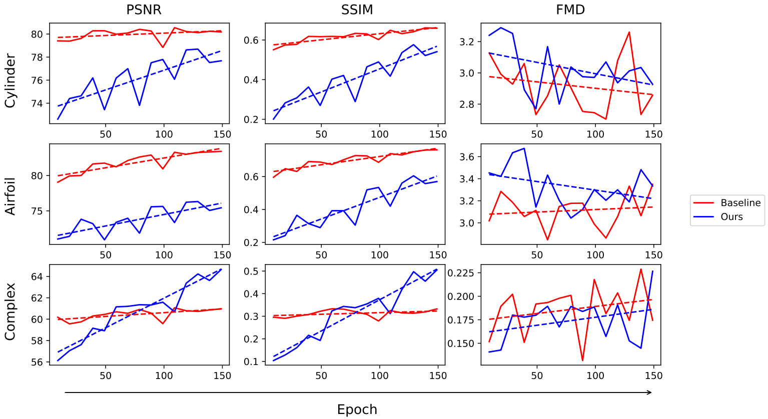


Figure: Tracking the model performance during iterations of training

Insights

- Complex problems:
  Our performance quickly surpassed Baseline's with the problem pool size increased.
- Simple problems:
  Our performance traces Baseline's.

We explored a few Encoder-Decoder-like architectures and their pre-train strategies for the fluid problem generator.

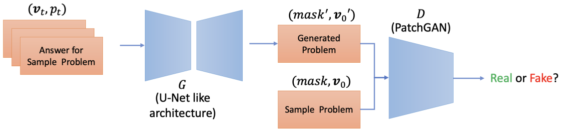


Figure: Pix2Pix-based Generator Pre-train Pipeline

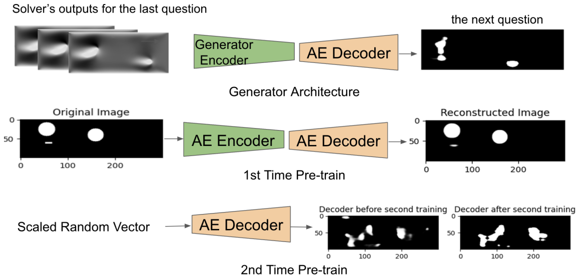


Figure: AE-based Generator Pre-train Pipeline

Generator	Performance
G1	Generated irrational problem, the pre-trained pattern cannot maintain
G2	Generated rational problems, problems are lack of diversity
G3	Generated rational problems with satisfactory diversity

G1: Pix2Pix

G2: ResNet-AE

G3 (Chosen): Fully Convolutional-AE

Table: Generator Attributes and Performance Metrics

Test ID	Generator	Modules			
		A	B	C	D
Full	G3	√	√	√	√
w/o GR		√	√	√	
w/o PR		√	√		√
w/o GMU		√		√	√

A: **P**roblem **P**ool
B: **G**enerator **R**einitialization
C: **P**roblem **R**enewal
D: **G**enerator **M**ultiple **U**pdates

Table: Ablation study setups

Experiments

Ablation Study - Results

	Test ID	PSNR	SSIM	FMD
Cylinder	Full	76.221	0.382	2.937
	w/o GR	76.040	0.364	3.000
	w/o PR	73.444	0.263	3.088
	w/o GMU	73.440	0.267	2.799
Airfoil	Full	72.268	0.287	3.288
	w/o GR	73.880	0.365	3.182
	w/o PR	71.080	0.254	3.400
	w/o GMU	71.244	0.252	3.429
Complex	Full	58.443	0.188	0.193
	w/o GR	57.855	0.199	0.180
	w/o PR	57.357	0.158	0.201
	w/o GMU	55.740	0.109	0.117

Best ; Second Best

A: **P**roblem **P**ool
B: **G**enerator **R**einitialization
C: **P**roblem **R**einitialization
D: **G**enerator **M**ultiple **U**pdates

Insights

- Top-2 performances: w/ and w/o GR.
- The top-2, while having a comparable performance, significantly outperformed the rest.

Experiments

Ablation Study - Results

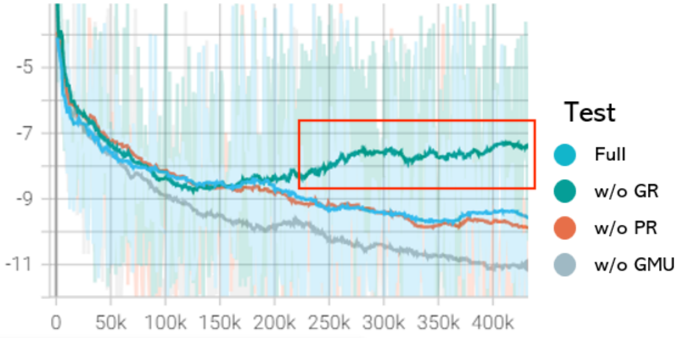


Figure: The learning loss during the training of different test settings, presented on a logarithmic scale.

The Full (w/ GR) is preferred over w/o GR considering the risks of w/o GR to converge to a local minimum and to lose generalizability.

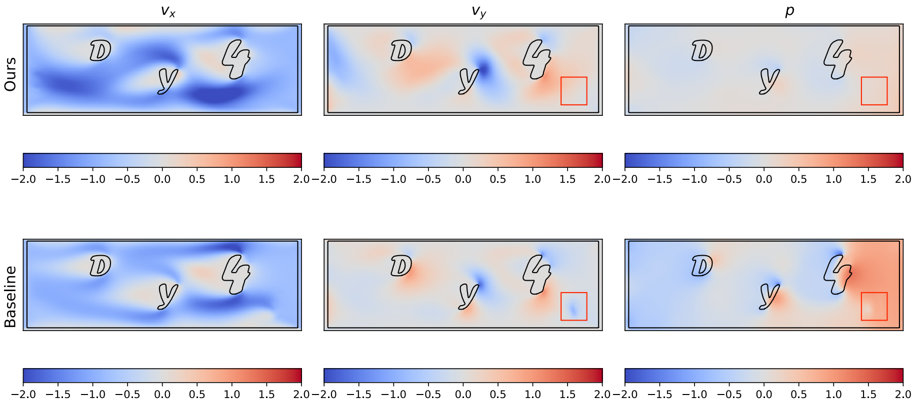


Figure: Qualitative comparison between ours and baseline on complex shapes on the steady state.

Experiments

Qualitative Results

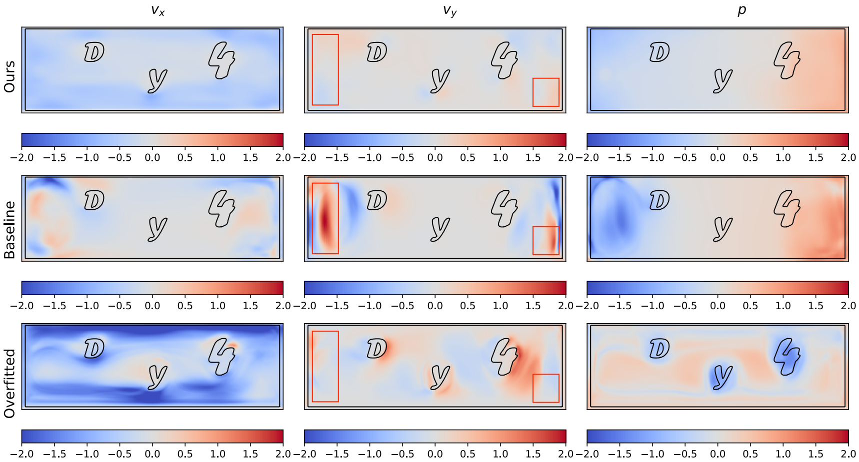


Figure: Qualitative comparison between ours and baseline on complex shapes on the unsteady state.

Applications

Interactive Interface Demonstration

- Demoing the **real-time** simulation ability.
- Demoing the **application potentials**, and can be further developed into real-world products.

Conclusion

- Goal achievements:
  - Generalizability.
  - Accuracy.
  - Efficiency.
- Contributions:
  - **Training strategy**
    - ★ We proposed a novel adversarial-learning-based PINN training strategy.
  - **Model design**
    - ★ We explored several designs for the problem generator.
    - ★ We also explored different components within the training pipeline to improve performance.
  - **Evaluation**
    - ★ We offer novel insights into evaluating different training strategies by using overfitted model to identify the upper bound of performance of solver with given architecture.

Conclusion

- Future plan:
  - **Training** Conduct more training tests with longer epochs and different seeds to examine its stability and full potential on both simple and complex problems.
  - **Evaluation** More comprehensive datasets and matrices.
  - **Application** Incorporate into industrial applications like active flow control.

Thank you!

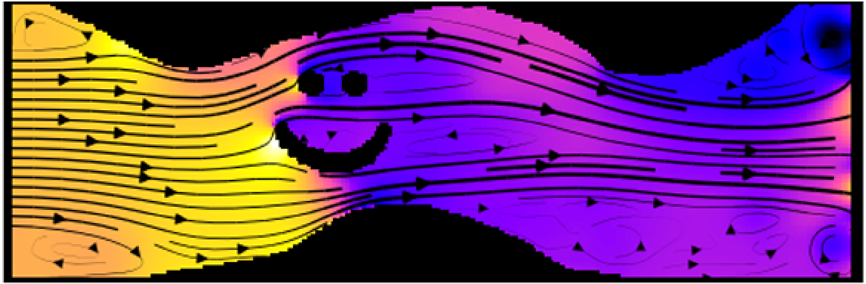


Figure: This is a thank you figure :)