Goal-Oriented Mesh Adaptation Based on Machine Learning Techniques

Guodong Chen   Krzysztof J. Fidkowski

Department of Aerospace Engineering

WCCM-ECCOMAS 2020

UNIVERSITY OF MICHIGAN

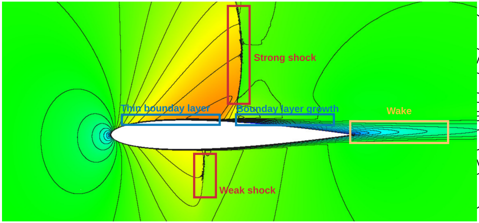
Outline

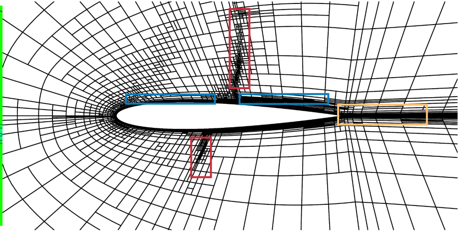
Outline

1. Motivation
2. Adjoint-based error estimation and mesh adaptation
3. Neural network surrogate model
4. Application to rectangular/square computational domains
5. Extension to irregular computational domains
6. Conclusions and future work

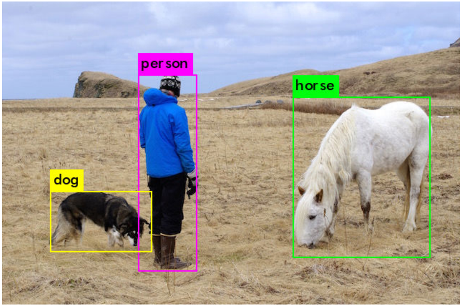
Mesh adaptation ⇔ Feature/object detection

Turbulent transonic airfoil: NACA 0012, $M = 0.8$, $\alpha = 1.25°$, $Re = 10^5$





Computer vision object detection tasks [Redmon et al. 2015]



Outline

Output error estimation

Suppose the governing PDE is parameterized by N_μ parameters, for a given discretization H, *i.e.*, mesh and approximation order, we can calculate the scalar output of interest, J.

$$\text{current space } H :\rightarrow \underbrace{\boldsymbol{\mu}}_{\text{parameters } \in \mathbb{R}^{N_\mu}} \rightarrow \underbrace{\mathbf{R}_H(\mathbf{U}_H;\boldsymbol{\mu}) = \mathbf{0}}_{N_H \text{ equations}} \rightarrow \underbrace{\mathbf{U}_H}_{\text{state } \in \mathbb{R}^{N_H}} \rightarrow \underbrace{J_H(\mathbf{U}_H)}_{\text{output (scalar)}}$$

Output error: $\delta J = J_H(\mathbf{U}_H) - J(\mathbf{U})$

This is the difference between J computed with the discrete system solution, $\mathbf{U}_H$, and J computed with the *exact* solution, $\mathbf{U}$.

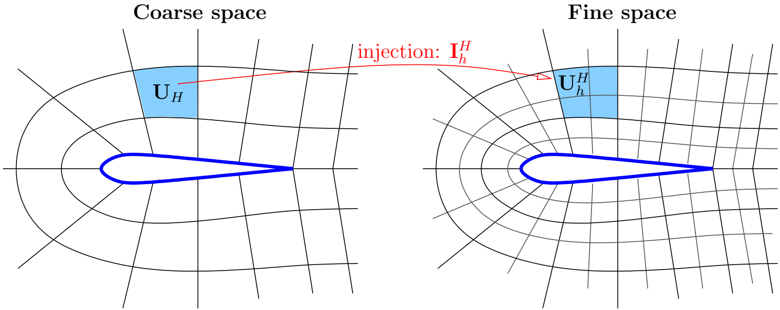
Error estimate: $\delta J = J_H(\mathbf{U}_H) - J_h(\mathbf{U}_h)$

This is the difference in J relative to a finer discretization h.

$$\text{finer space } h :\rightarrow \underbrace{\boldsymbol{\mu}}_{\text{parameters } \in \mathbb{R}^{N_\mu}} \rightarrow \underbrace{\mathbf{R}_h(\mathbf{U}_h;\boldsymbol{\mu}) = \mathbf{0}}_{N_h \text{ equations}} \rightarrow \underbrace{\mathbf{U}_h}_{\text{state } \in \mathbb{R}^{N_h}} \rightarrow \underbrace{J_h(\mathbf{U}_h)}_{\text{output (scalar)}}$$

Fine-space state injection

- The fine space can arise from spatial or order refinement
- We do not solve the fine space discretized equations $\mathbf{R}_h(\mathbf{U}_h; \boldsymbol{\mu}) = \mathbf{0}$
- We use fine space adjoint $\boldsymbol{\Psi}_h$ and the fine space residual $\mathbf{R}_h(\mathbf{U}_h^H)$
- Define an injection of the coarse space state into the fine space



- The injected states $\mathbf{U}_h^H$ will generally not satisfy the fine space discretized equations,

$$\mathbf{R}_h(\mathbf{U}_h^H; \boldsymbol{\mu}) \neq \mathbf{0}.$$

The adjoint-weighted residual

- The fine space adjoint $\mathbf{\Psi}_h$ is defined as the *sensitivity* of the output to the residual perturbation,

$$\left[\frac{\partial \mathbf{R}_h}{\partial \mathbf{U}_h}\right]^T \mathbf{\Psi}_h + \left[\frac{\partial J_h}{\partial \mathbf{U}_h}\right]^T = \mathbf{0}$$

- The injected states produce a residual perturbation, $\mathbf{R}_h(\mathbf{U}_h^H; \boldsymbol{\mu}) \neq \mathbf{0}$
- The adjoints then transfer the residual perturbation to an output perturbation

$$\delta J \approx J_h(\mathbf{U}_h^H) - J_h(\mathbf{U}_h) \approx \underbrace{\frac{\partial J_h}{\partial \mathbf{U}_h}\delta\mathbf{U} = \overbrace{-\mathbf{\Psi}_h^T \frac{\partial \mathbf{R}_h}{\partial \mathbf{U}_h}\delta\mathbf{U}}_{\text{adjoint definition}} \approx -\mathbf{\Psi}_h^T \delta\mathbf{R}_h}^{\text{residual linearization}}$$

$$= \underbrace{-\mathbf{\Psi}_h^T \mathbf{R}_h(\mathbf{U}_h^H; \boldsymbol{\mu})}_{\text{adjoint-weighted residual}}$$

Mesh adaptation

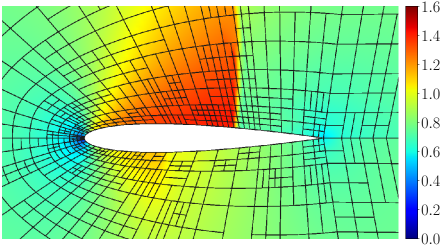
- The adjoint-weighted residual involves a sum of the local errors over elements

$$\delta J = -\boldsymbol{\Psi}_h^T \mathbf{R}_h(\mathbf{U}_h^H; \boldsymbol{\mu}) = \sum_e -\boldsymbol{\Psi}_{h,e}^T \mathbf{R}_{h,e}(\mathbf{U}_h^H; \boldsymbol{\mu})$$

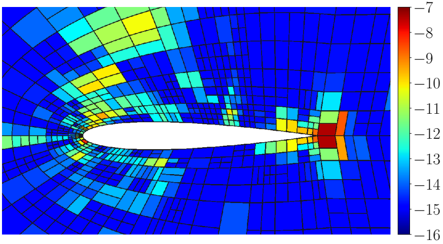
- The absolute value of each element's contribution can serve as the adaptive error indicator on that element

$$\epsilon_e = \left|\boldsymbol{\Psi}_{h,e}^T \mathbf{R}_{h,e}(\mathbf{U}_h^H; \boldsymbol{\mu})\right|$$

- Elements with large adaptive indicators are targeted for adaptation



Mach contours



Adaptive indicator contours (log scale)

- Can we directly build a map from the states to the adaptive indicators?

Outline

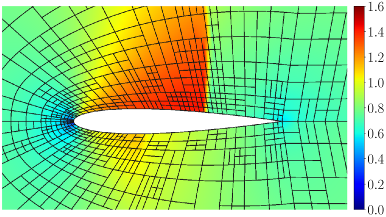
Corresponding computer vision tasks

Output error prediction $\Longleftrightarrow$ Image classification

Image Classification →

Person in the image?

Yes, label 1



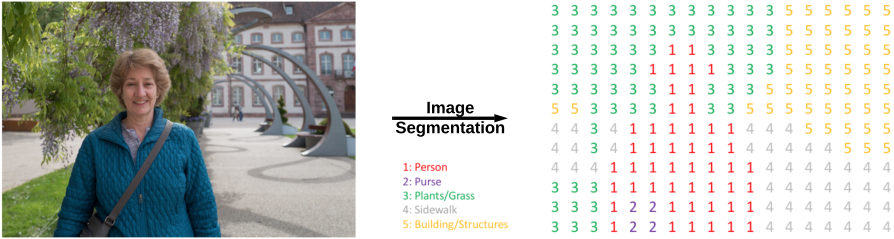
Error Prediction →

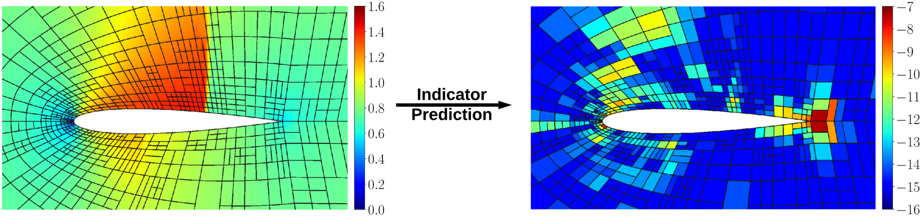
How much error in the drag?

$\delta J = 3.85 \times 10^{-4}$

Corresponding computer vision tasks

Adaptive indicator prediction $\Longleftrightarrow$ Image segmentation [Jordan, 2018]





- Difference: integer-valued vs. real-valued
- State of the art technique is the convolutional neural network (CNN)

Encoder-decoder CNN

- Adaptive error indicator prediction $\Longleftrightarrow$ Image segmentation
- Challenge: high-dimensional inputs and outputs
- Go through low-dimensional representations
- Paradigm: encoder-decoder type CNN

  Encoding: High-dimensional input $\xrightarrow{Convolution}$ Low-dimensional codes

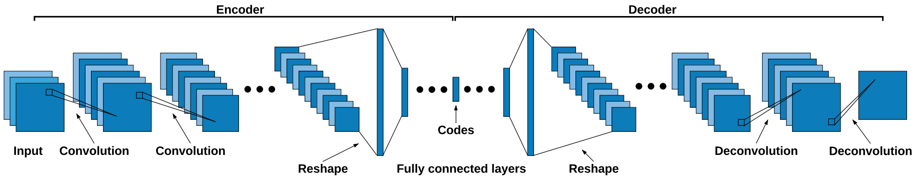
  Decoding: Low-dimensional codes $\xrightarrow{Deconvolution}$ High-dimensional output

Encoder-decoder CNN

- Adaptive error indicator prediction $\Longleftrightarrow$ Image segmentation
- Challenge: high-dimensional inputs and outputs
- Go through low-dimensional representations
- Paradigm: encoder-decoder type CNN

  Encoding: High-dimensional input $\xrightarrow{Convolution}$ Low-dimensional codes

  Decoding: Low-dimensional codes $\xrightarrow{Deconvolution}$ High-dimensional output

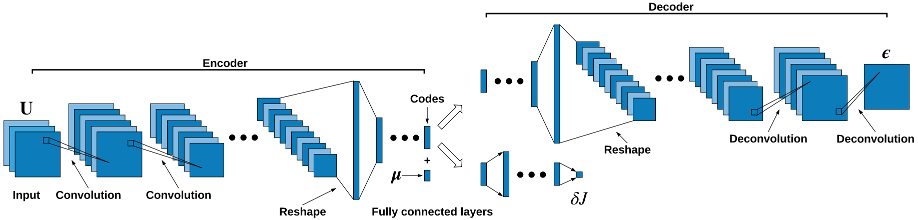


Encoder-decoder CNN

- Adaptive error indicator prediction $\Longleftrightarrow$ Image segmentation
- Challenge: high-dimensional inputs and outputs
- Go through low-dimensional representations
- Paradigm: encoder-decoder type CNN

  Encoding: High-dimensional input $\xrightarrow{Convolution}$ Low-dimensional codes

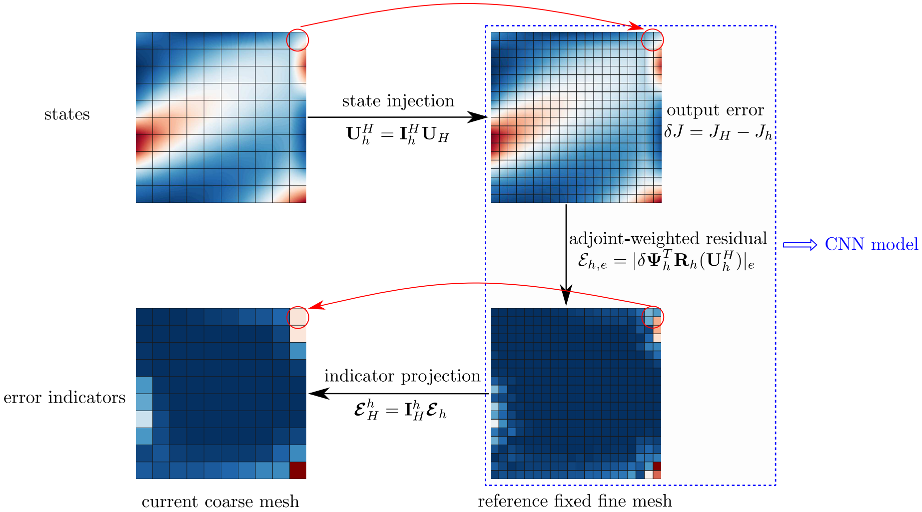
  Decoding: Low-dimensional codes $\xrightarrow{Deconvolution}$ High-dimensional output
- Incorporate within error estimation task



Outline

Fixed network for adaptive simulations

- In adaptive simulations, state and indicator dimensions are changing
- For a fixed network, both input and output dimensions are fixed
- Train the network on a fixed reference mesh



2D scalar advection-diffusion problem

- 2D advection-diffusion system in a square domain $\Omega = [0, 1]^2$

$$\vec{V} \cdot \nabla u - \nu \nabla^2 u = 0, \quad (x, y) \in \Omega;$$

$$u = \exp(0.5 \sin(-4x + 6y) - 0.8 \cos(3x - 8y)), \quad (x, y) \in \partial\Omega.$$

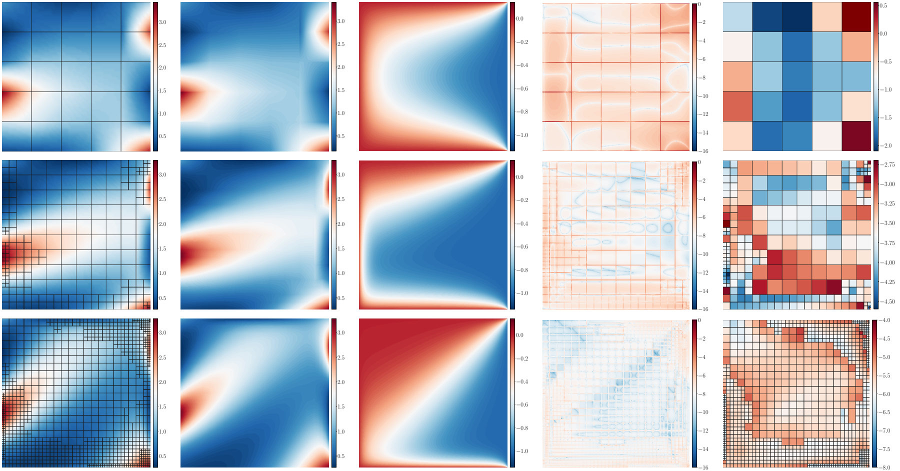
$\vec{V} = [\cos\alpha, \sin\alpha]$: unit advection velocity, ν : viscosity
u : scalar state, $Pe \equiv |\vec{V}|L/\nu$: Péclet number

- Parametrized discretized form

$$\mathbf{R}(\mathbf{U}; \boldsymbol{\mu}) = \mathbf{0}, \quad \boldsymbol{\mu} = \{Pe, \alpha\}.$$

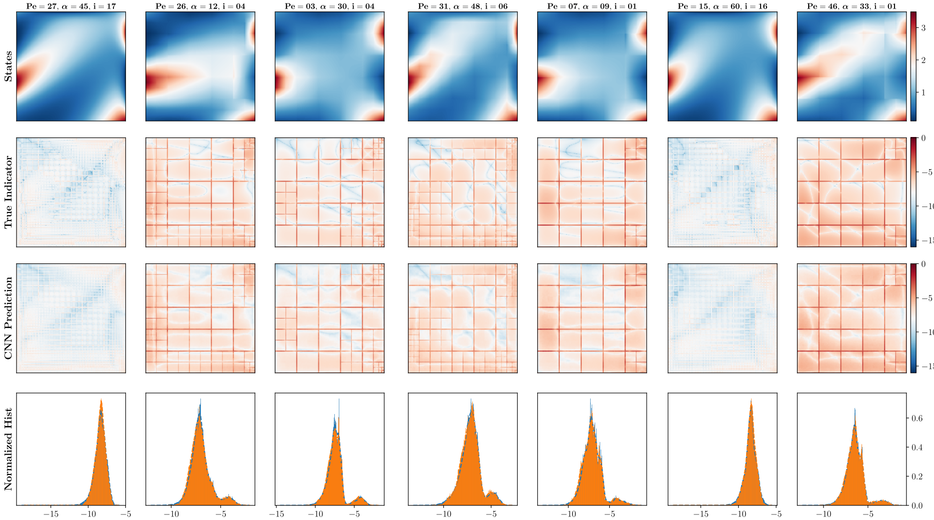
- Output of interest J: integral of flux, $-\nu\nabla u$, on the right boundary

Three samples from the dataset:

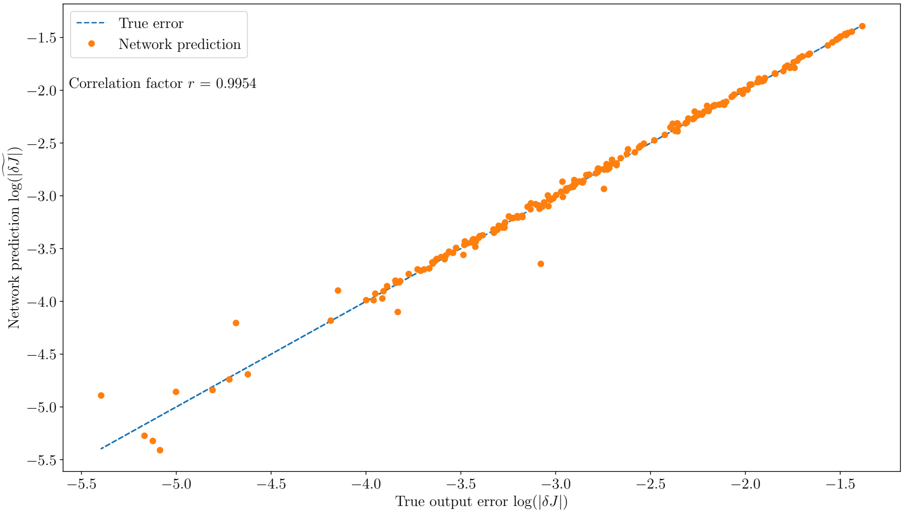


$\mathbf{U}_H$	$\mathbf{U}_h^H$	$\mathbf{\Psi}_h$	$\log(\boldsymbol{\epsilon}_h)$	$\log(\boldsymbol{\epsilon}_H^h)$
	Network input		Network output	

Adaptive indicator predictions on the testing set



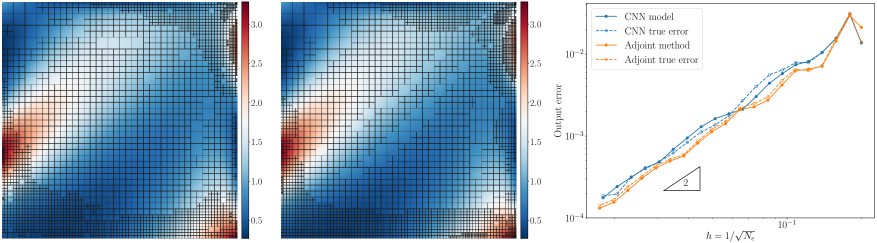
Output error predictions on the testing set



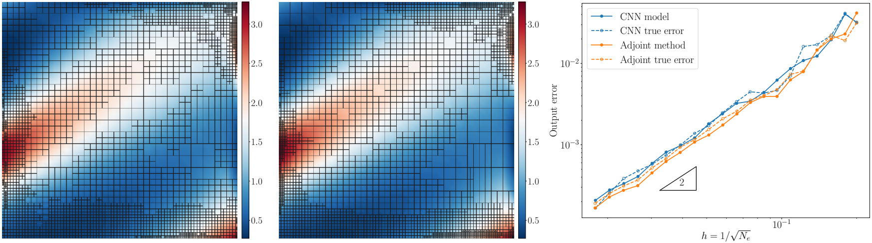
Model deployment on the testing set

Model deployment in real-time simulations:

CNN meshes | Adjoint meshes | Output error



$(Pe, \alpha) = (31, 48^\circ)$

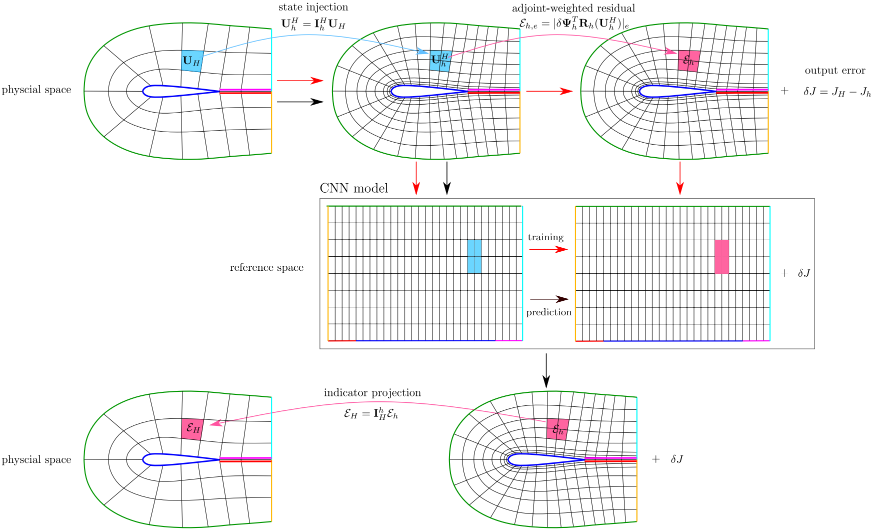


$(Pe, \alpha) = (46, 33^\circ)$

Outline

Irregular computational domain/geometry

- Current mesh ⇒ Fixed reference mesh ⇒ Cartesian mesh (reference space)



Inviscid transonic flow over NACA 4-digit airfoils

- Euler equations (inviscid fluid flow) in a C-shaped computational domain

$$\nabla \cdot \vec{\mathbf{F}}(\mathbf{u}) = \mathbf{0},$$

$\vec{\mathbf{F}}$ is the convective fluxes and $\mathbf{u}$ is the state vector, $\mathbf{u} = [\rho, \rho u, \rho v, \rho E]$

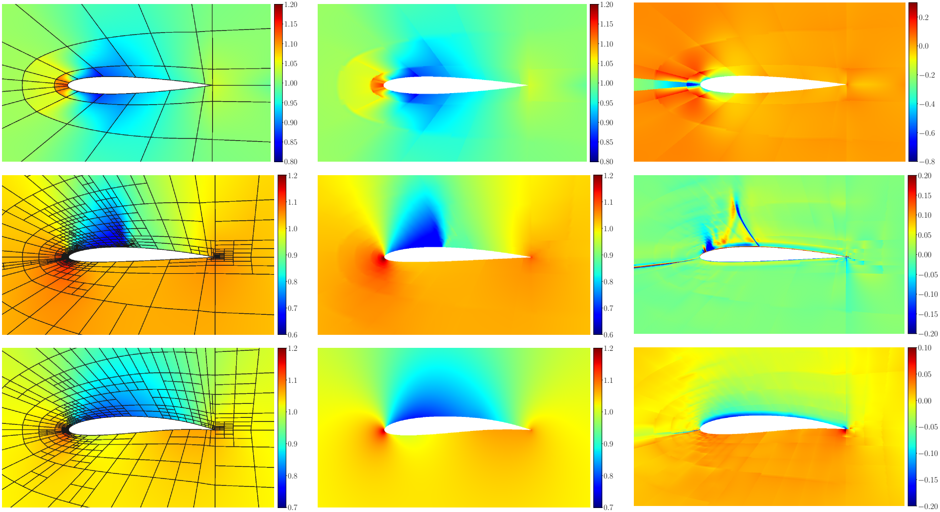
- Parametrized discretized form

$$\mathbf{R}(\mathbf{U}, \boldsymbol{\mu}) = \mathbf{0}, \quad \boldsymbol{\mu} = \{M, \alpha, S\},$$

M: free-stream Mach number, α: angle of attack, S: airfoil shape

- Output of interest: drag over the airfoil

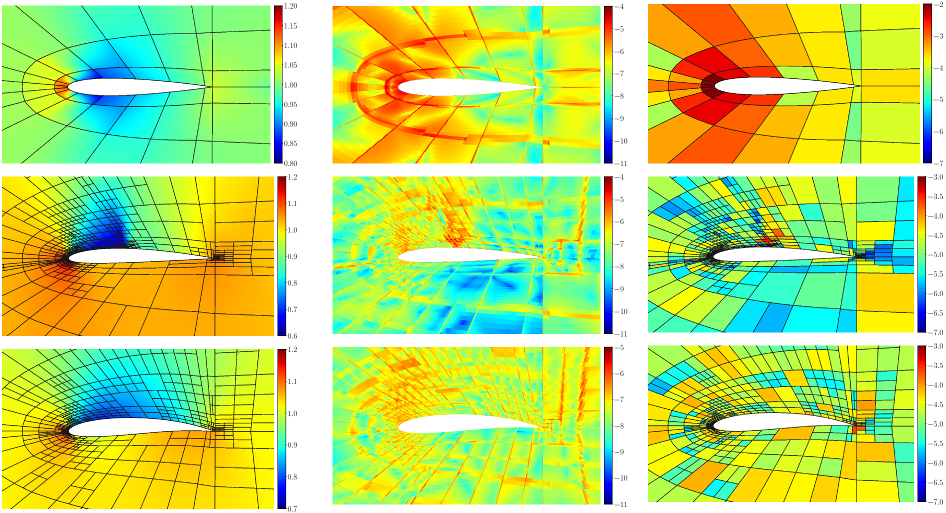
Training data samples



$\mathbf{U}_H$ $\mathbf{U}_h^H$ $\mathbf{\Psi}_h$

Network inputs

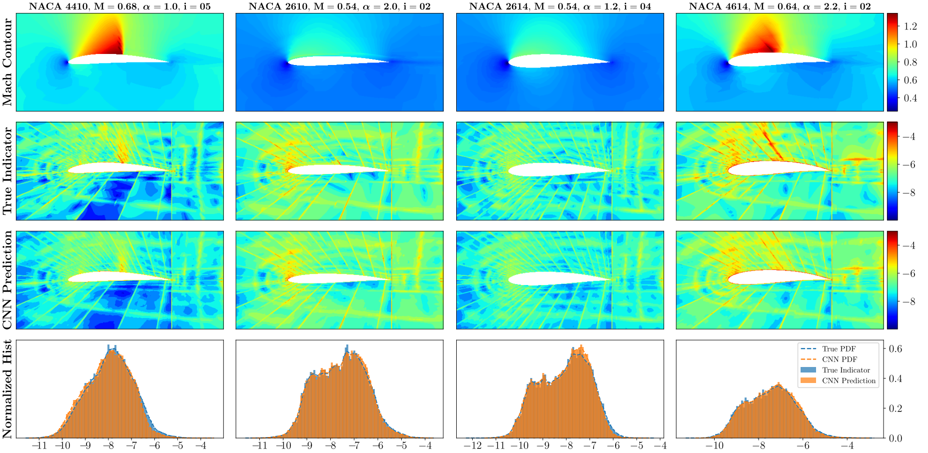
Training data samples



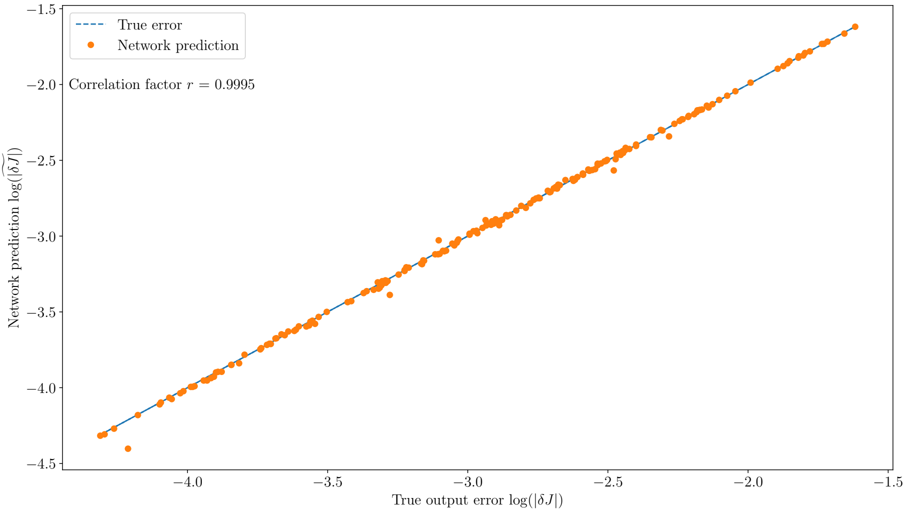
$\mathbf{U}_H$ $\log(\boldsymbol{\epsilon}_h)$ $\log(\boldsymbol{\epsilon}_H^h)$

Network outputs

Adaptive indicator predictions on the testing set

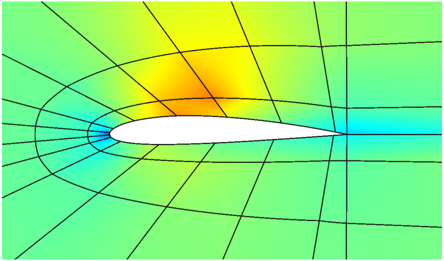


Output error predictions on the testing set

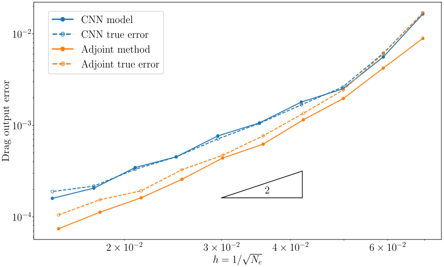


Model deployment (unsampled M)

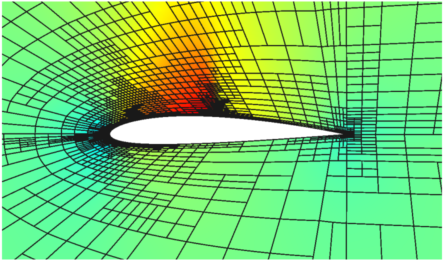
NACA 2412, $M = 0.70, \alpha = 1.0°$



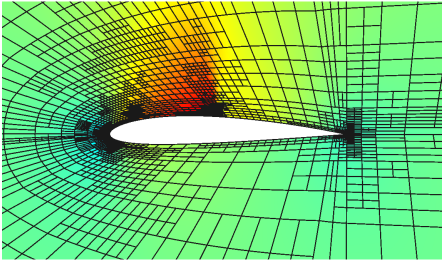
initial mesh



drag error convergence



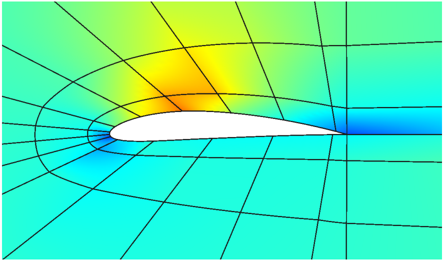
CNN mesh



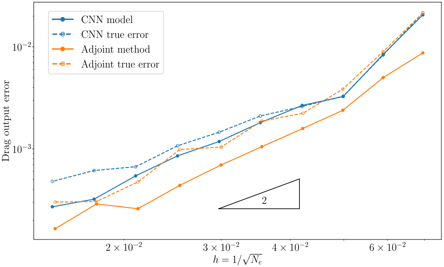
adjoint mesh

Model deployment (unsampled α)

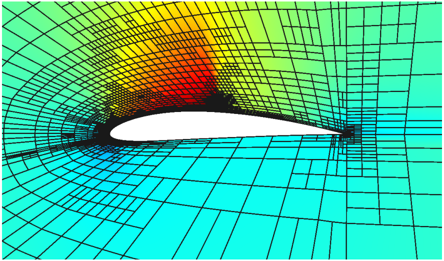
NACA 4412, $M = 0.62, \alpha = 4.0°$



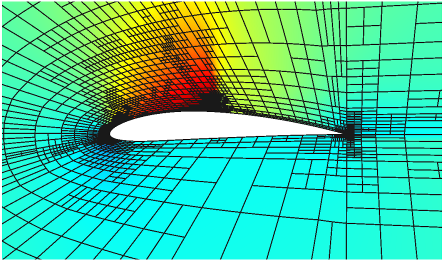
initial mesh



drag error convergence



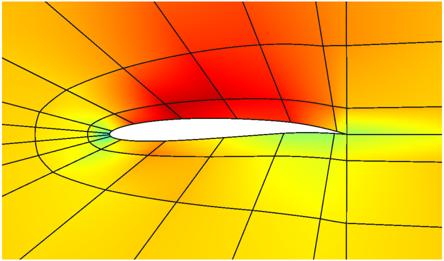
CNN mesh



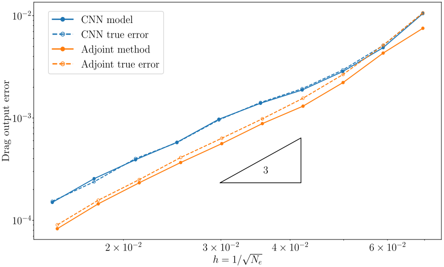
adjoint mesh

Model deployment (unsampled shape)

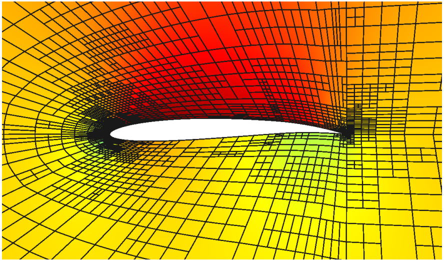
NACA 3709, $M = 0.66, \alpha = 0^\circ$



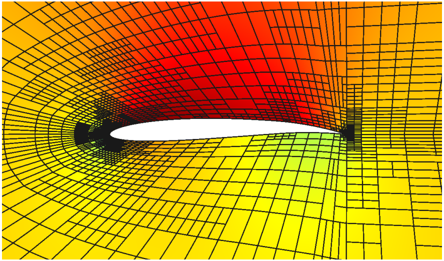
initial mesh



drag error convergence



CNN mesh



adjoint mesh

Outline

Conclusions and future work

Conclusions:

- Adjoint-based method: analytical, requires additional adjoint solutions
- Proposed CNN-based method: non-intrusive, generalizes adaptation knowledge from data
- Encoder-decoder type CNN is capable of predicting both the adaptive error indicator and the output error
- Physical-reference mapping provides a way to generalize CNN model to irregular computational domains
- For more detailed analysis/implementation, checkout my thesis at www.gdchen.me

Future work:

- Advanced training techniques and fine tuning
- Improve the efficiency: share encoder-decoder parameters (symmetric)
- Sparsity constraints in the latent layer: enforce independent codes
- Including physical-reference mapping (Jacobian) into the model to resolve multi-scale physics

Acknowledgments

Department of Energy
DE-FG02-13ER26146/DE-SC0010341
Boeing Company, with technical monitor Dr. Mori Mani
— Thank you —