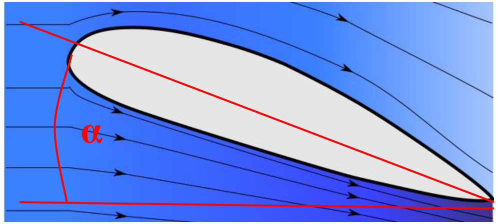
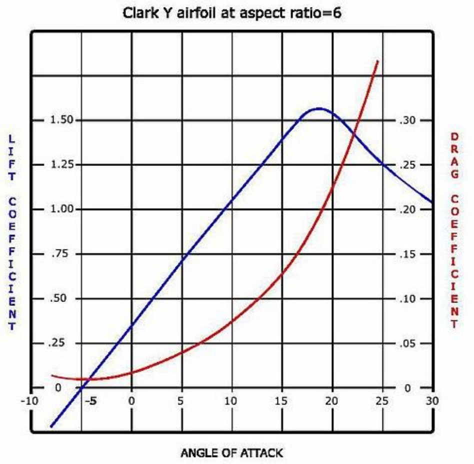
Artificial neural networks for airfoil shape design in the subsonic regime

Xiangbei Liu, Marius Ruh, Mingyuan Wu

Background *[Marius]*



- An airfoil is the 2D cross-section of a wing or rotor blade
- Airfoil analysis provides insight into the following aerodynamic parameters:
    - **Lift coefficient C_l**
    - **Drag coefficient C_d**
    - Moment coefficient C_m
    - Pressure coefficient C_p

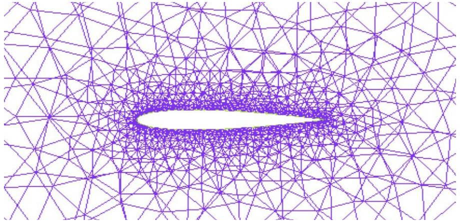


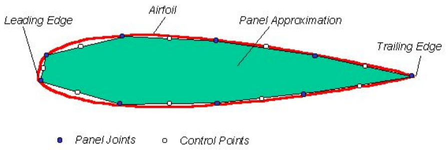
- Under subsonic conditions ($M < 1$) C_l and C_d mostly depend on:
    - Angle of attack α (AoA)
    - Reynolds number *Re*

Motivation *[Marius]*

There are two main methods to predict C_l and C_d:

1. CFD (computational fluid dynamics) solver
    - Pro: exact flow solution by solving Navier-Stokes equations
    - Con: Computationally expensive, long simulation time





2. (Vortex) panel method (Xfoil is state-of-art)
    - Pro: fast prediction by dividing airfoil into panels
    - Con: Inaccurate compared to solving NSE, not always robust

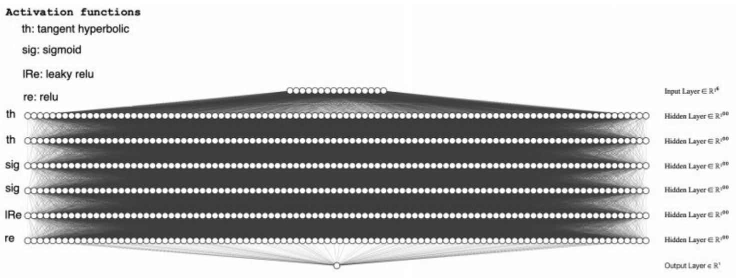
<u>**Goal**</u>: Develop ML model to predict C_l and C_d by using method (2) to generate database to ensure:

- Instantaneous and robust prediction of C_l and C_d
- Suitability for gradient based optimization

Literature survey *[Marius]*

Predicting airfoil performance using ML has gained popularity over the last two years

- Li et al. [2019]
    - Airfoil parameterization via singular value decomposition (SVD) to generate airfoil mode shapes for transonic and subsonic flow regimes → 2 separate models
    - Surrogate modelling techniques such as gradient enhanced kriging and partial least squares → Not actual ML
- Bouhlel et al. [2020]
    - Similar airfoil parameterization approach via SVD but combined for transonic and subsonic regime
    - Used gradient enhanced artificial neural network (ANN)
    - CFD solver uses adjoint method to calculate derivatives of Cl/Cd w.r.t to AoA and Mach number



Literature survey for neural network algorithm *[Xiangbei]*

1. Czarnecki et al.: *Sobolev Training for Neural Networks*
    a. It incorporates the gradient information in the loss function with the training samples while training artificial neural network.
    b. Improve the quality of our predictors, as well as the data-efficiency and generalization capabilities of function approximation.

2. Bouhlel et al.: *Scalable gradient–enhanced artificial neural networks for airfoil shape design in the subsonic and transonic regimes*
    a. Training gradient-enhanced artificial neural network (SANN and mSANN) to model the aerodynamic force coefficients of airfoils in both subsonic and transonic regimes.
    b. mSANN is used to to introduce the gradient information gradually during the learning process by incorporating a parameter to set the weight of gradient information.

Algorithm 1: SANN

Input: $(\mathbf{X}, \mathbf{y}, \mathbf{df}, N)$;
Define the ANN architecture;
Initialize a set of neural network model parameters θ;
for $i \leq N$ **do**

$$\min_{\theta} \bar{y}_{\text{loss}}(\mathbf{X}, \mathbf{y}, \mathbf{df} \mid \theta),$$
$$\bar{y}_{\text{loss}}(\mathbf{X}, \mathbf{y}, \mathbf{df} \mid \theta) := \sum_{i=1}^{n_t} l\left(m\left(\mathbf{x}^{(i)} \mid \theta\right), y^{(i)}\right) + \sum_{j=1}^{d} l_j\left(\frac{\partial m}{\partial x_j}\left(\mathbf{x}^{(i)} \mid \theta\right), \frac{\partial f}{\partial x_j}\left(\mathbf{x}^{(i)}\right)\right)$$

end
Output: $\hat{y}(\mathbf{x})$;

Algorithm 2: mSANN

Input: $(\mathbf{X}, \mathbf{y}, \mathbf{df}, N)$;
Define the ANN architecture;
Define λ_k;
Initialize a set of neural network model parameters θ;
for $i \leq N$ **do**
for $\lambda \in list\,(\lambda_k)$ **do**

$$\min_{\theta} \bar{y}_{\text{loss}}(\mathbf{X}, \mathbf{y}, \mathbf{df} \mid \theta),$$
$$\bar{y}_{\text{loss}}(\mathbf{X}, \mathbf{y}, \mathbf{df} \mid \theta) := \sum_{i=1}^{n_t} l\left(m\left(\mathbf{x}^{(i)} \mid \theta\right), y^{(i)}\right) + \lambda_k \sum_{j=1}^{d} l_j\left(\frac{\partial m}{\partial x_j}\left(\mathbf{x}^{(i)} \mid \theta\right), \frac{\partial f}{\partial x_j}\left(\mathbf{x}^{(i)}\right)\right)$$

end
end
Output: $\hat{y}(\mathbf{x})$;

Details on the dataset *[Marius]*

Output data

Run Xfoil for:
- -14°<AoA<14°
- Re = 6 Re

to obtain C_l and C_d

Smooth data:
- C_l: Linear fit
- C_d: Least squares with regularization

Extrapolate C_l and C_d to ±90° via Viterna model (airfoil is modeled as flat plate in stall regime)

Input data

Obtain B-spline control points for upper and lower airfoil surface

Pair control points with all combinations of AoA and Re

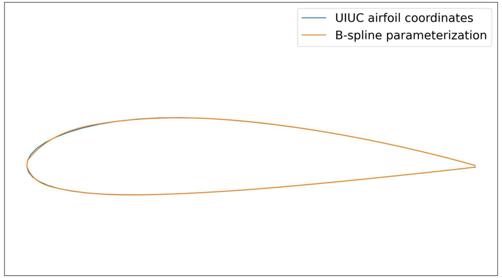
Details on the dataset *[Marius]*

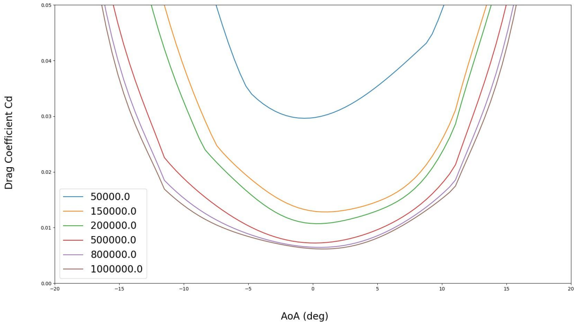
Our own dataset

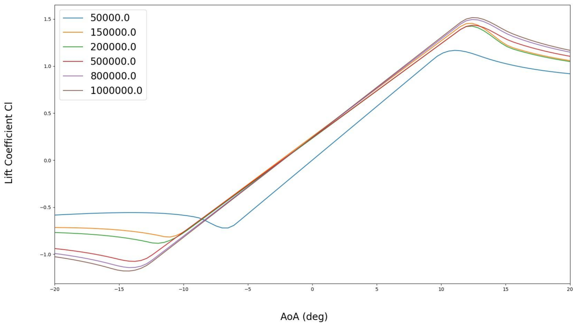
1. Pick 100 airfoils from the UIUC airfoil database (contains almost 1700 airfoils)
2. **Run Xfoil for 100 airfoils** at 6 different Reynolds numbers and 400 different AoA
   - n = 100 x 6 x 400 = 240 000 samples for C_l and C_d
   - **Output shape is n x 1 = 240,000 x 1** for C_l and C_d
3. Represent all airfoil shapes in the same way via B-spline parameterization
   - B-splines are piecewise polynomials defined by control points (cp), each having a pair of x and y coordinates
   - We chose 8 cp for the airfoil lower surface and 7 for the upper surface
   - d = 32 = number of input variables: 2 x 15 cp (30 x-y coordinates) + 1 AoA + 1 Reynolds number
   - **Input shape is n x d = 240,000 x 32**

Details on the dataset *[Marius]*

NACA 2414 airfoil



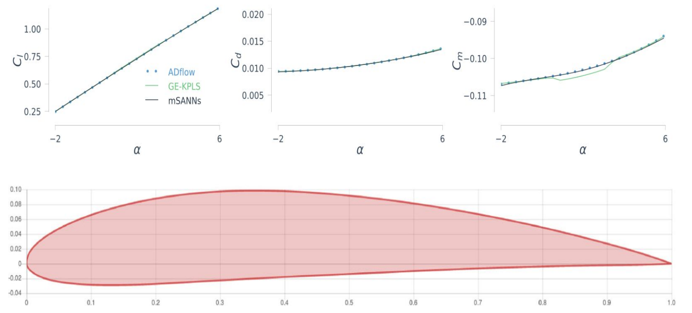




Details on feature extraction used *[Xiangbei]*

1. Our own data:
   a. B-spline representation to define the airfoil geometry.
   b. A larger range of attack angles with corresponding Reynolds numbers.

2. Bouhlel's mSANN model benchmark data:
   a. Using inverse distance weighting (IDW) to interpolate the surface function of each airfoil.
   b. Then applying singular value decomposition (SVD) to reduce the number of variables that define the airfoil geometry. It includes a total of 14 airfoil modes (seven for camber and seven for thickness) .
   c. Totally 16 input variables, two flow conditions of Mach number (0.3 ~ 0.6) and the angle of attack ($2^\circ \sim 6^\circ$) plus 14 shape coefficient.
   d. The output airfoil aerodynamic force coefficients and their respective gradients are computed using ADflow, which solves the RANS equations with a Spalart–Allmaras turbulence model.



(a) NACA 4412 airfoil

Details on the model used *[Mingyuan]*

Table of layers:

The structure of the dataset depends on the dimension of the input data and the number of data points in the set.

Layers	Activation function	Number of neurons
Input layer	N/A	16
Hidden layer 1	ReLu	120
Hidden layer 2	ReLu	120
Hidden layer 3	ReLu	120
Hidden layer 4	ReLu	120
Hidden layer 5	ReLu	120
Hidden layer 6	ReLu	60
Output layer	N/A	1

Three models for comparison:
1. ANN 2. Sobolev ANN (SANN) 3. Modified SANN (mSANN) *[Mingyuan]*

ANN:

$$\min_\theta \bar{y}_{loss}(\mathbf{X}, \mathbf{y}, \mathbf{df} \mid \theta)$$
$$\bar{y}_{loss}(\mathbf{X}, \mathbf{y}, \mathbf{df} \mid \theta) := l\left(m\left(\mathbf{x}^{(i)} \mid \theta\right), y^{(i)}\right)$$

SANN:

$$\min_\theta \bar{y}_{loss}(\mathbf{X}, \mathbf{y}, \mathbf{df} \mid \theta)$$
$$\bar{y}_{loss}(\mathbf{X}, \mathbf{y}, \mathbf{df} \mid \theta) := l\left(m\left(\mathbf{x}^{(i)} \mid \theta\right), y^{(i)}\right) +$$
$$\sum_{j=1}^{d} l_j\left(\frac{\partial m}{\partial x_j}\left(\mathbf{x}^{(i)} \mid \theta\right), \frac{\partial f}{\partial x_j}\left(\mathbf{x}^{(i)}\right)\right)$$

mSANN:

$$\min_\theta \bar{y}_{loss}(\mathbf{X}, \mathbf{y}, df \mid \theta),$$
$$\bar{y}_{loss}(\mathbf{X}, \mathbf{y}, \mathbf{df} \mid \theta) := \sum_{i=1}^{n_t} l\left(m\left(\mathbf{x}^{(i)} \mid \theta\right), y^{(i)}\right) +$$
$$\lambda_k \sum_{j=1}^{d} l_j\left(\frac{\partial m}{\partial x_j}\left(\mathbf{x}^{(i)} \mid \theta\right), \frac{\partial f}{\partial x_j}\left(\mathbf{x}^{(i)}\right)\right)$$

Why SANN?

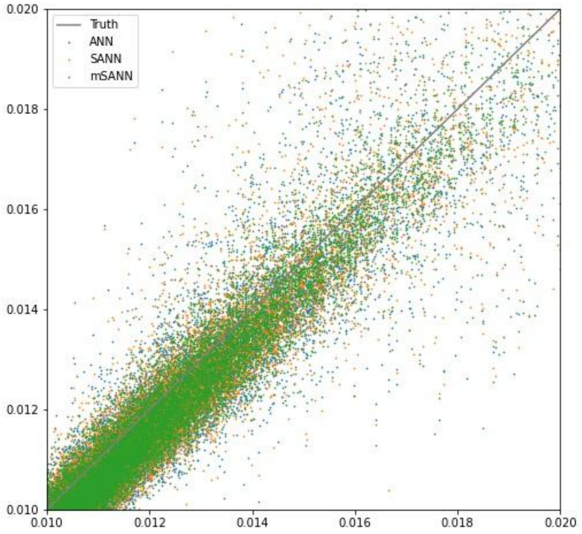
To exploit the derivative of the physical model when training a neural network. This can increase the accuracy of prediction.

Why mSANN?

Accelerate the training convergence by controlling how much information is used in the SANN model.

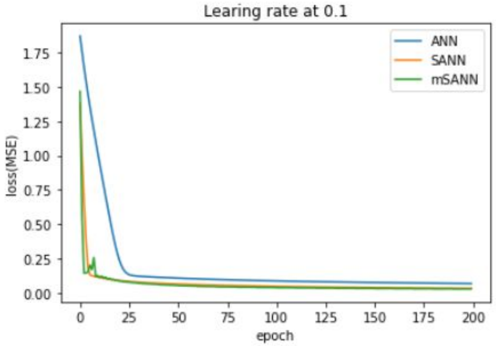
Results/Observations for Bouhlel's dataset *[Xiangbei]*

1. Prediction plot for each model



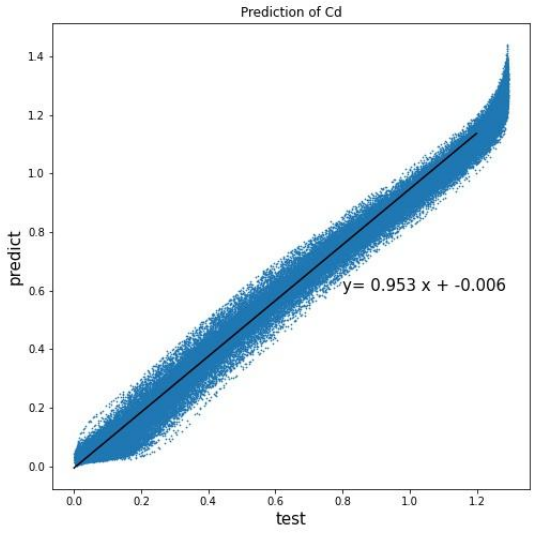
2. Evaluation methods: MSE, R2 score, loss

	MSE	R2 score
ANN	2.417e-6	0.828
SANN	2.250e-6	0.839
mSANN	2.165e-6	0.846



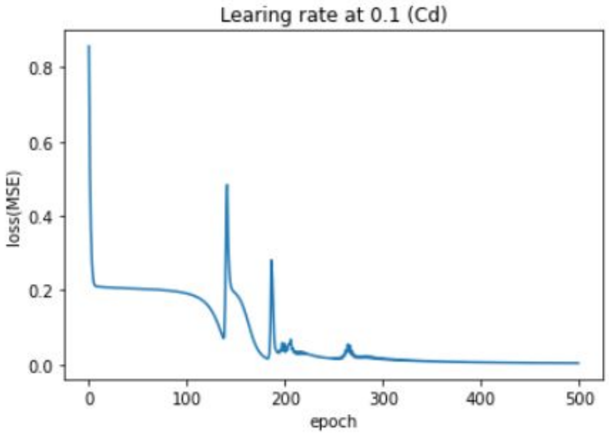
Results/Observations for our own dataset *[Xiangbei]*

1. Prediction plot for each model



2. Evaluation methods: MSE, R2 score, loss

	MSE	R2 score
ANN	0.0037	0.982



Further items to be completed *[Xiangbei]*

1. Cross Validation
2. Generate more dataset with more types of subsonic airfoils
3. Compute derivatives for our own dataset to train in SANN and mSANN

References

1. UIUC airfoil database: https://m-selig.ae.illinois.edu/ads/coord_database.html
2. Czarnecki, Wojciech Marian, et al. ”Sobolev training for neural networks.” arXiv preprint arXiv:1706.04859 (2017).
3. J. Li, M. A. Bouhlel, and J. R. R. A. Martins. Data-based approach for fast airfoil analysis and optimization. *AIAA Journal*, 57(2):581–596, February 2019. doi: 10.2514/1.J057129.
4. Bouhlel, Mohamed Amine, Sicheng He, and Joaquim RRA Martins. "Scalable gradient–enhanced artificial neural networks for airfoil shape design in the subsonic and transonic regimes." *Structural and Multidisciplinary Optimization* (2020): 1-14.
5. Bouhlel, Mohamed Amine; HE, Sicheng; Martins, Joaquim (2019), “mSANN model benchmarks”, Mendeley Data, V1, doi: 10.17632/ngpd634smf.1
6. Drela, Mark. "XFOIL: An analysis and design system for low Reynolds number airfoils." *Low Reynolds number aerodynamics*. Springer, Berlin, Heidelberg, 1989. 1-12.

Run down of the code