Oral presentation | Data science and AI
Data science and AI-IV
Fri. Jul 19, 2024 10:45 AM - 12:45 PM Room C

[13-C-01] Flow Prediction and Shape Optimization of Aerodynamic Bodies Using Physics-Informed Machine Learning

Prashant Kumar[1], *Rajesh Ranjan[1] (1. Indian Institute of Technology, Kanpur)

Keywords: Physics Informed Neural Network, NACA-2412, CFD

©Retained by Authors

Twelfth International Conference on Computational Fluid Dynamics (ICCFD12), Kobe, Japan, July 14-19, 2024

ICCFD12-2024-xxxx

Flow Predictions behind NACA 2412 Aerofoil Using Physics-Informed Machine Learning

Prashant Kumar and Rajesh Ranjan†
Corresponding author: †rajeshr@iitk.ac.in
Indian Institute of Technology Kanpur, India.

1 Introduction

Predictions of aerodynamic flows, such as those over airfoils and wings, are of great interest to the design community. Typically, computational fluid dynamics (CFD)-based tools are extensively used for these predictions during the design cycle. Although CFD has been successful in these predictions to a great extent, their usage in shape optimization to achieve a higher lift-to-drag ratio is rather limited due to the large cost involved [1]. Further, these CFD computations require an essential pre-processing step, generating a good-quality mesh for convergence, which is often challenging for highly cambered airfoils and complex turbomachinery blades. Further, the intermediate designs in a traditional shape optimization process can be a complex shape, for which it would be difficult to obtain convergence.

In this paper, we evaluate the ability to quantitatively predict flow over NACA 2412 airfoil using Physics-informed Neural Networks (PINN, [2]). Although the applications of PINN in simple configurations have shown the potential to predict fluid flows reasonably well, the literature showing its utility for practical configurations is sparse. Even where some results are reported [3], typically integrated quantities like total lift and drag are matched to the reference data, and rigorous comparisons of flow profiles, as done in CFD, are missing. This work attempts to fill this gap while also reporting the limitations and advantages of PINN over CFD.

Note that while training data can be used to make predictions faster and better in PINN, current predictions are made without employing any training data. Therefore, the predictions are made purely by training the neural networks to minimise losses in the underlying partial differential equations (PDEs) and boundary conditions, as shown in figure 1. Further, since PINN works on training networks on a given distribution of collocation points, no pre-processor is necessary to generate grids. Although the PINN approach discussed in this paper is computationally more expensive than the traditional CFD approach, the former approach is advantageous during shape optimization as the well-trained network can be readily used for parametric studies.

2 Numerical Approach

In this work, PINN has been used to solve the incompressible steady Navier-Stokes equation for flow over the airfoil NACA-2412 at $Re = 100$. The computational domain is taken as $[X \times Y] \rightarrow [[-5, 15] \times$

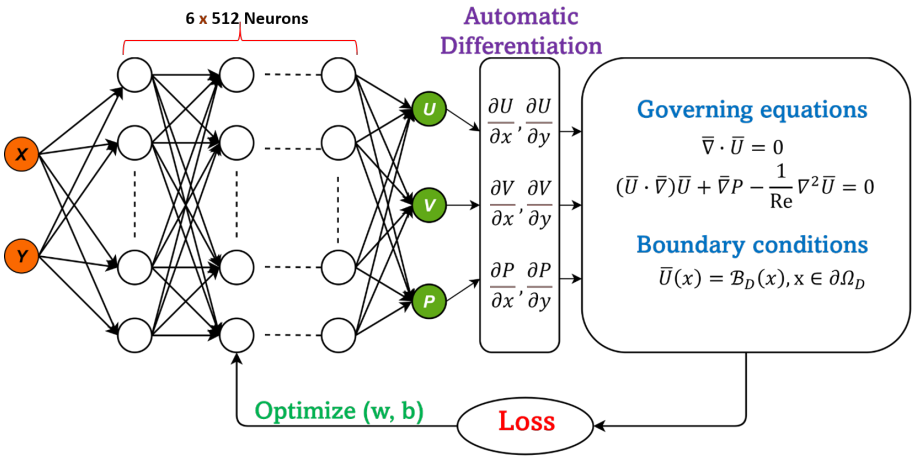
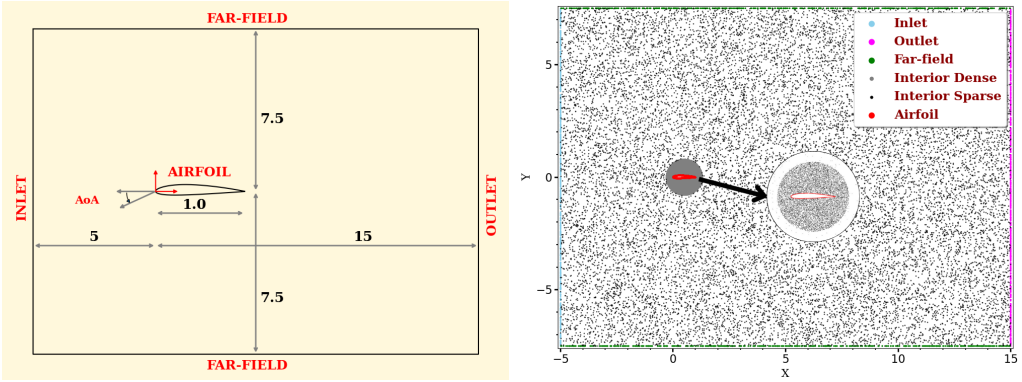


Figure 1: Neural Network Architecture



(a) Computational domain (b) Residual point distribution

Figure 2: Computational Domain for NACA 2412

$[-7.5, 7.5]]$ as shown in Fig. 2(a). The inlet and far-field are assigned uniform velocity in the angle of attack α-direction; the airfoil is set to no-slip boundary conditions; and the outlet is set to pressure boundary conditions.

Now we describe the network used for the current work. All the problems studied in the paper are solved using Forward Neural Network (FNN; [4]). The ML model for fluid flow computations has been solved in Nvidia-Modulus [5] - a library developed for Scientific Machine Learning. In FNN, residual points are the input, flowfields are the output, and there are several levels of hidden layers, also called deep layers. In equation 1, the mathematical structure of PINN is given for L number of deep layers.

$$z^1 = W^1 X + b^1 \tag{1a}$$
$$z^2 = W^2 \sigma_1 \left(z^1\right) + b^2 \tag{1b}$$
$$\cdots$$
$$z^{L-1} = W^{L-1} \sigma_{L-2} \left(z^{L-2}\right) + b^{L-1} \tag{1c}$$
$$z^L = W^L \sigma_{L-1} \left(z^{L-1}\right) + b^L \tag{1d}$$

Here X is the input, W^i is the weight matrix, σ_i is the activation function, b^i is the bias and z^i is the output of ith layer. Weightage W^i is applied to every residual point, which is then summed with a bias b^i and finally passed through an activation function σ_i to provide the desired non-linearity in the solution. The output of the activation function is passed to subsequent layer neurons and the same process is followed till it reaches the output layer. The output, $Y^L = \sigma_{L-1}\left(z^L\right)$, is then obtained from the values in the final layer.

Figure 1 shows the network used for the NACA 2412 flow predictions. The input vectors are the coordinates x and y, while u, v and p are desired output variables. This output value of the network is verified to satisfy governing equations, which for our problem is the steady incompressible Navier-Stokes equations. These equations are employed in the non-dimensional form so that the orders of magnitude of pressure and velocity are not very different. This helps in the training process of the weights and biases. The network also needs to satisfy the boundary conditions as mentioned earlier. Therefore, the network minimizes the total loss, given in equation (2), which is a weighted sum of the losses due to satisfying the governing equations ($\mathcal{L}_{\Omega_i}$) and the boundary conditions ($\mathcal{L}_{\Gamma_j}$). λ_i and λ_j are weights given to partial differential equation loss and boundary condition loss, respectively. These are hyperparameters which can be varied to get the desired solution.

$$\mathcal{L}_{Total} = \sum \lambda_i * \mathcal{L}_{\Omega_i} + \sum \lambda_j * \mathcal{L}_{\Gamma_j} \tag{2}$$

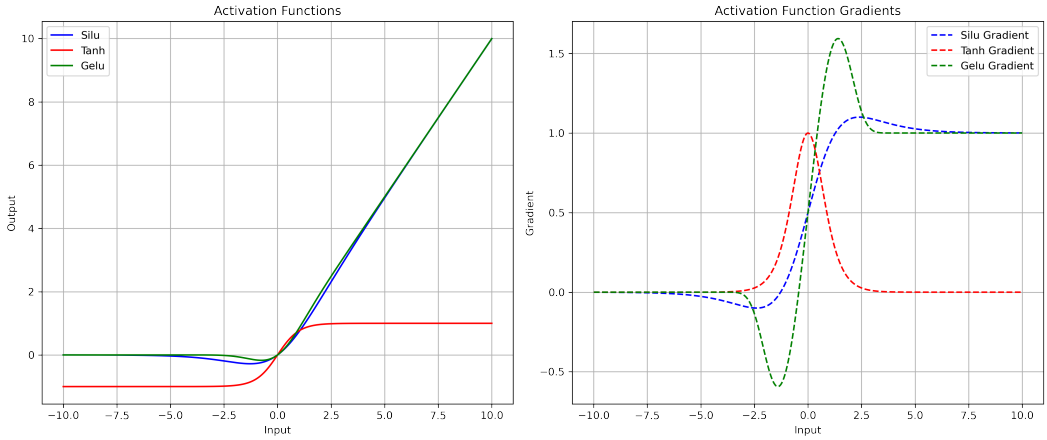


Figure 3: Activation functions and their gradients

$$
\begin{aligned}
\mathcal{L}_{\Omega 1} &= \frac{1}{|\hat{\Omega}|} \sum_{\mathbf{x} \in \hat{\Omega}} \lambda_r \left\| \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right\|^2 \\
\mathcal{L}_{\Omega 2} &= \frac{1}{|\hat{\Omega}|} \sum_{\mathbf{x} \in \hat{\Omega}} \lambda_r \left\| u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + \frac{\partial p}{\partial x} - \frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \right\|^2 \\
\mathcal{L}_{\Omega 3} &= \frac{1}{|\hat{\Omega}|} \sum_{\mathbf{x} \in \hat{\Omega}} \lambda_r \left\| u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + \frac{\partial p}{\partial y} - \frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) \right\|^2
\end{aligned}
\tag{3}
$$

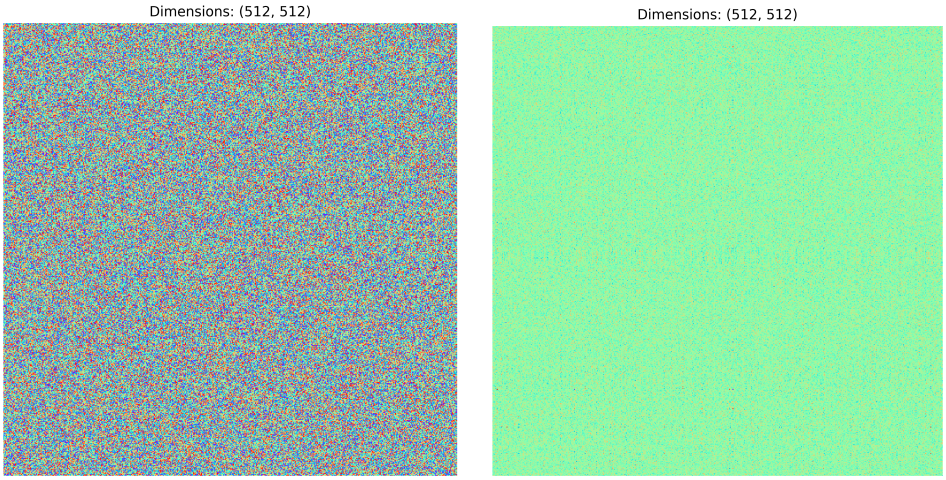
$$
\mathcal{L}_{\Gamma_i} = \frac{1}{\left|\hat{\Gamma}_i\right|} \sum_{\mathbf{x} \in \hat{\Gamma}_i} \left\| \mathcal{B}_i - (\mathcal{U}, \mathcal{P}) \right\|^2 \tag{4}
$$

PDE loss, eq. (3), consists of losses in the mass ($\mathcal{L}_{\Omega_1}$) and momentum ($\mathcal{L}_{\Omega_2}$, $\mathcal{L}_{\Omega_3}$) conservations. These losses are premultiplied by λ_r to avoid the conflict of the boundary conditions near discontinuities. In the current work, λ_r is based on the signed distance function (SDF), evaluated from the airfoil as well as the top and bottom boundaries of the domain.

Figure 2(b) shows the distribution of residual points used for PINN predictions. A total of $30,000$ points were used in the interior domain with $10,000$ points densely packed near the airfoil. To satisfy boundary conditions, 1000 residual points are used on the airfoil as well as the far-field, while 640 residual points are employed at both the inlet and outlet. Batch training has been used with 1000 random collocation points sent in batch per iteration.

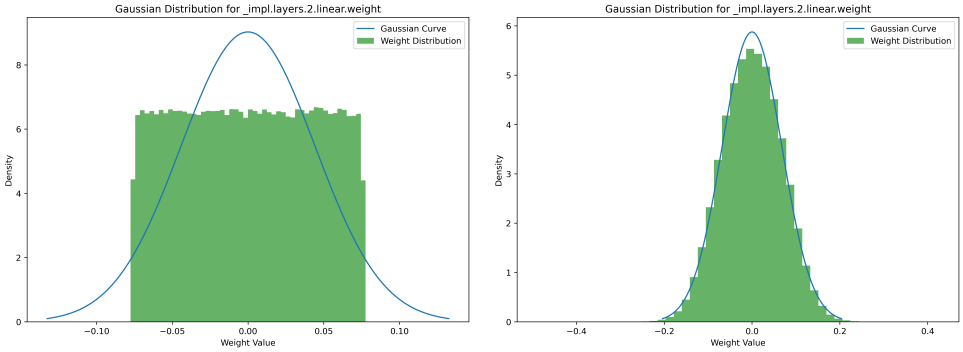
Six deep layers, each with 512 neurons, have been used for building the neural network. Three different activation functions, Sigmoid Linear Units(SILU; [6]), Tan-hyperbolic (TANH), and Gaussian Error Linear Units (GELU; [7]) are employed in different attempts to study the network performance to estimating the correct output. Figure 3 shows all three functions as well as their gradients. All three functions as well as their gradients are smooth. SILU tends to work better when networks are batch-normalized. GELU prevents the problem of vanishing gradients, it has a continuous derivative at 0, which can sometimes make training faster.

The Xavier random initialization was used to initialize weights. The main idea is to set the initial weights of the network in a way that allows the activations and gradients to flow effectively during both forward and backpropagation. In this approach, the numbers of input and output units in each layer are used to determine the scale of the random initialization. For training, an 'ADAM' optimizer with an adaptive learning rate of 0.001 was used for 0.2 million epochs. The number of epochs is estimated based on the experience to achieve a total mean-square error (MSE) loss to at least 10^{-3}.



(a) Untrained weight matrix (b) Trained weight matrix

Figure 4: Weight matrix in FNN



(a) Gaussian curve of untrained weight matrix (b) Gaussian curve of trained weight matrix

Figure 5: Gaussian Weight matrix in FNN

3 Results and Discussions

PINN Predictions are obtained for a wide range of angles of attacks, $0^0 \leq \alpha \leq 20^0$, and the results are compared with those from CFD simulations obtained using the commercial software, Ansys FLUENT.

Before discussing the results, we discuss the consistency of our ML approach. We select the $\alpha = 0^0$ prediction, with SILU function as well as equal weightage of governing equation and BC in loss calculation, for this discussion: similar observations were made for other cases also. Figures 4 and 5 show matrices of initial (untrained) and final (trained) weights. From figures 4(a) and 5(a), it can be observed that the untrained matrix is sparse and distribution seems random as expected from the Xavier Initialization. The final weight matrices, figures 4(b) and 5(b), however, show less sparse distributions which approximate the Gaussian curve. Figure 6 shows final weight distributions in successive layers. It is observed that all the layers are reasonably well trained and they all have similar sparsity qualitatively.

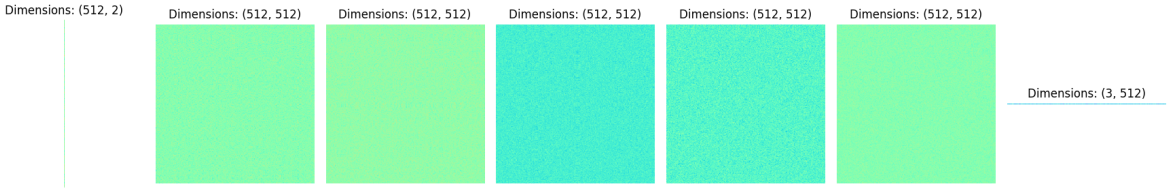


Figure 6: Weight matrix in each layer

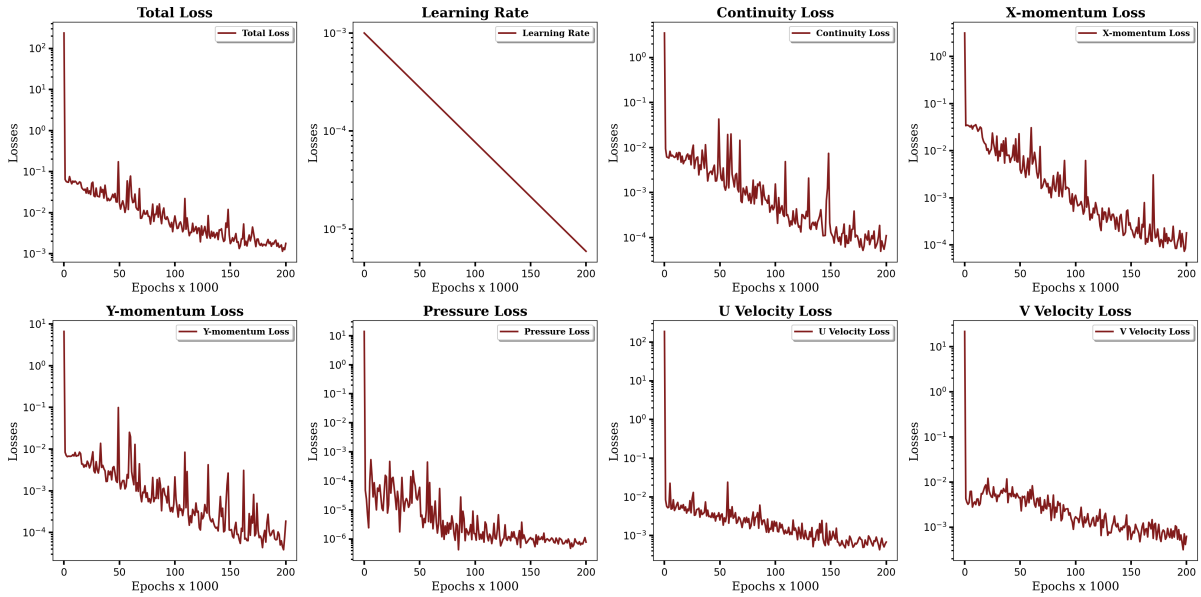


Figure 7: Change in losses with epochs

Figure 7 shows the change in losses with epochs. The adaptation in the learning rate with epochs is also presented. It is observed that total loss has dropped to the order of 10^{-3}. The PDE losses are reduced to 10^{-4} level. All the data losses at the boundary are represented in terms of pressure and velocity losses. While the pressure loss has reduced to below the 10^{-6} level, the losses in U and V velocities are around 10^{-3}. The learning rate has adapted logarithmically from 10^{-3} to 10^{-5}.

Now we focus on flow predictions. Figure 8 shows qualitative comparisons between PINN and CFD using contours of three primary flow variables for $\alpha = 0^0$ case. For all three variables, PINN predictions are very close to CFD estimates. The highest pressure obtained in PINN is slightly lower than that obtained in CFD. Note that the pressure reported is the gauge pressure and the negative pressure just indicates the expansion as the base pressure is set to zero value.

Using the predicted pressure and velocity values, aerodynamic forces are calculated. Equations (5) provide the formulae to compute forces in x and y-directions, while eqns (6) gives the net lift and drag forces by taking appropriate components of F_x and F_y.

$$\begin{aligned} F_x &= \int \left(-p \cdot n_x + \mu \left(\frac{\partial u}{\partial x} \cdot n_x + \frac{\partial u}{\partial y} \cdot n_y \right) \right) ds \\ F_y &= \int \left(-p \cdot n_y + \mu \left(\frac{\partial v}{\partial x} \cdot n_x + \frac{\partial v}{\partial y} \cdot n_y \right) \right) ds \end{aligned} \tag{5}$$

$$\begin{aligned} F_l &= F_y \cos\alpha - F_x \sin\alpha \\ F_d &= F_y \sin\alpha + F_x \cos\alpha \end{aligned} \tag{6}$$

The drag and lift coefficients for angles of attack between 0^0 and 20^0 are shown in figure 9 and 10 respectively. The distributions of lifts and drags due to pressure and viscous forces are also shown. For more granular comparison, these data are presented in tables 1 and 2 for PINN and CFD respectively.

Looking at the lift curve, PINN is able to predict lift values very close to CFD for angles less than 10^0. The departure between the two plots increases with the angle of attack. Further, both pressure and viscous lifts are estimated reasonably in PINN. Higher discrepancies are found in the case of drag force which is due to slight underprediction of viscous drag by PINN. Ref.[3] have a slightly better match in drag force, however their PINN and CFD estimations were done at a much lower Reynolds number ($Re = 20$) and a relatively simple Joukowski airfoil configuration. It is well known that prediction of flow with machine learning becomes difficult with an increase in Reynolds number.

Table 3 shows the percentage difference in lift and drag forces between PINN and CFD. The difference

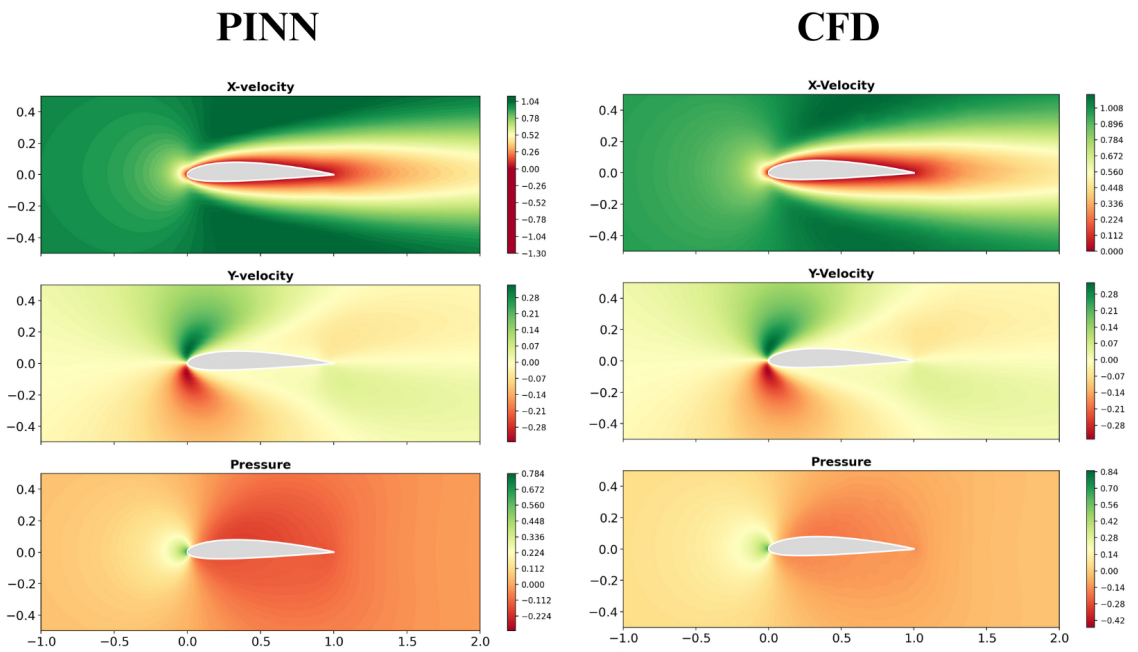


Figure 8: Comparison of PINN predictions and CFD estimates ($\alpha = 0^0$)

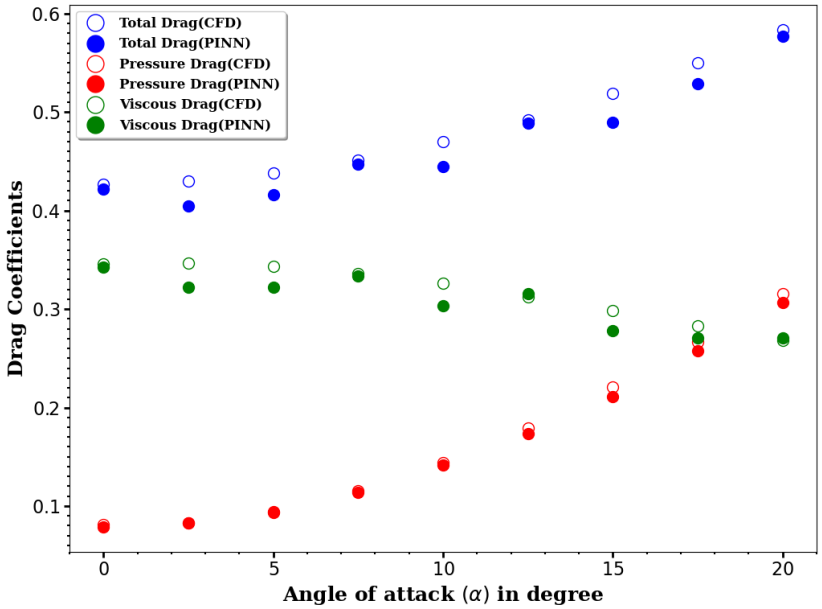


Figure 9: Drag Coefficients (C_d) using PINN and CFD

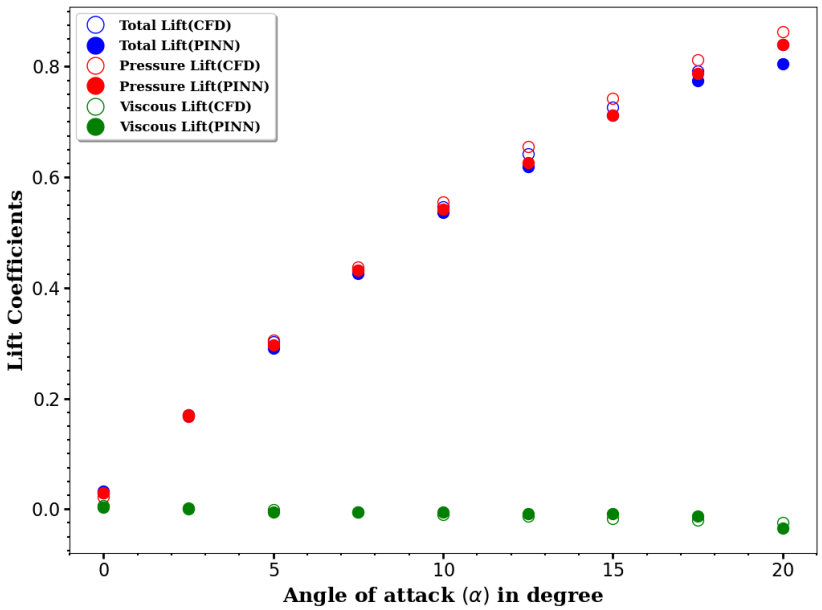


Figure 10: Lift Coefficients (C_l) using PINN and CFD

Table 1: Aerodynamic Coefficients calculated using PINN

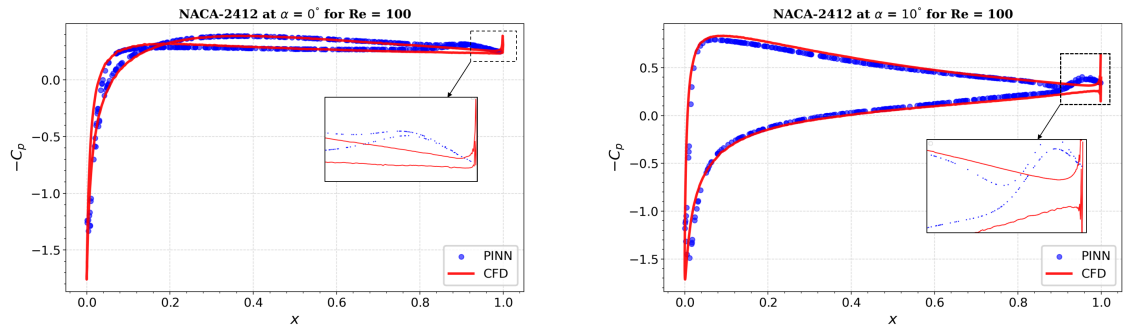
AoA	Pressure Drag	Viscous Drag	Total Drag	Pressure Lift	Viscous Lift	Total Lift
0.0	0.03952	0.17146	0.21098	0.01456	0.00181	0.01637
2.5	0.0414	0.16103	0.20244	0.08511	0.00001	0.08511
5	0.04663	0.1612	0.20784	0.14823	-0.00297	0.14526
7.5	0.05694	0.16678	0.22375	0.21571	-0.00245	0.21322
10	0.07083	0.15163	0.22247	0.27084	-0.00304	0.2678
12.5	0.08686	0.15771	0.24457	0.31347	-0.00404	0.30943
15	0.1057	0.13907	0.24478	0.35586	-0.00414	0.35586
17.5	0.12899	0.13545	0.26445	0.39399	-0.00664	0.38734
20	0.15334	0.13515	0.2885	0.41982	-0.01697	0.40285

Table 2: Aerodynamic Coefficients calculated using CFD

AoA	Pressure Drag	Viscous Drag	Total Drag	Pressure Lift	Viscous Lift	Total Lift
0.0	0.04045	0.17285	0.2133	0.01136	0.00303	0.01439
2.5	0.04148	0.17345	0.21493	0.0836	0.00112	0.08472
5	0.04718	0.17182	0.219	0.15242	-0.000827	0.151593
7.5	0.05761	0.1681	0.22571	0.21861	-0.00282	0.21579
10	0.07198	0.16296	0.23494	0.27765	-0.0047	0.27295
12.5	0.08946	0.15636	0.24582	0.32795	-0.00645	0.3215
15	0.11046	0.1491	0.25956	0.371534	-0.0082	0.363334
17.5	0.13342	0.14158	0.275	0.40596	-0.01	0.39596
20	0.15767	0.13413	0.2918	0.43163	-0.01188	0.41975

Table 3: Percentage difference in Lift and Drag forces due to PINN and CFD

AoA	2.5	5	7.5	10	12.5	15	17.5	20
% Diff. in Lift	0.3659	4.1776	1.1909	1.8867	3.7542	2.0570	2.1769	4.0262
% Diff. in Drag	5.8111	5.0959	0.8684	5.3077	0.5085	5.6945	3.8363	1.1309



(a) At $\alpha = 0^0$ (b) At $\alpha = 10^0$

Figure 11: Coefficients of Pressure, C_p

is calculated from $\alpha = 2.5^0$, after which the lift coefficient is sufficiently high for comparison. Consistently between 2.5^0 and 20^0, the difference in predicted c_l is less than 4.2%, while the difference in predicted c_d is less than 6%. This shows a good predictive ability of PINN for such flows.

To further probe the cause of differences between PINN and CFD, we plot pressure coefficients, C_p, along the chord. Figure 11(a) and (b) show prediction of $-C_p$ for $\alpha = 0^0$ and $\alpha = 10^0$ respectively. Predictions for CFD are also plotted for reference. PINN is able to successfully predict pressure distributions on both pressure and suction sides for most parts of the airfoil including the leading edge. However, significant differences were found for flow near the trailing edge. PINN shows an abrupt rise in the curve at around $x/c = 0.9$ on both sides. In fact, the flow seems unphysical in this region.

To correct this unphysical behaviour, several attempts were made. At first, predictions were obtained with different numbers of collocation points as well as for a much higher number of epochs but they did not provide any improvement near the trailing edge. The above-reported results were obtained when the governing equation and BC had equal weightage. To correct the behaviour near the trailing edge of the airfoil, the weightage of the airfoil BC was increased significantly. Figure 13 shows predictions when the weightage (λ) of the airfoil was increased to 10 and 100 times. For $\lambda = 10$, the behaviour in the central region of the airfoil became better, however, the trailing edge error remained. Increasing the weightage to $\lambda = 100$ worsened the results as a large mismatch was found in the central region. This may be due to the very weak enforcement of mass and momentum conservations in this region.

Next, we perform studies using two different activation functions including TANH and GELU to see their effects on TE flow predictions. Figure 13 shows the C_p predictions using all three activation functions. Although none of the activation functions are able to correct the TE behaviour, GELU seems

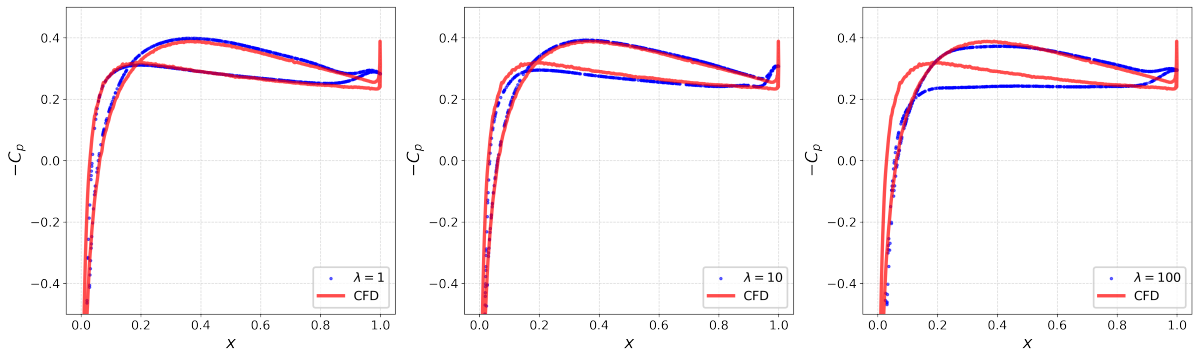


Figure 12: C_p prediction with different weights of airfoil BC

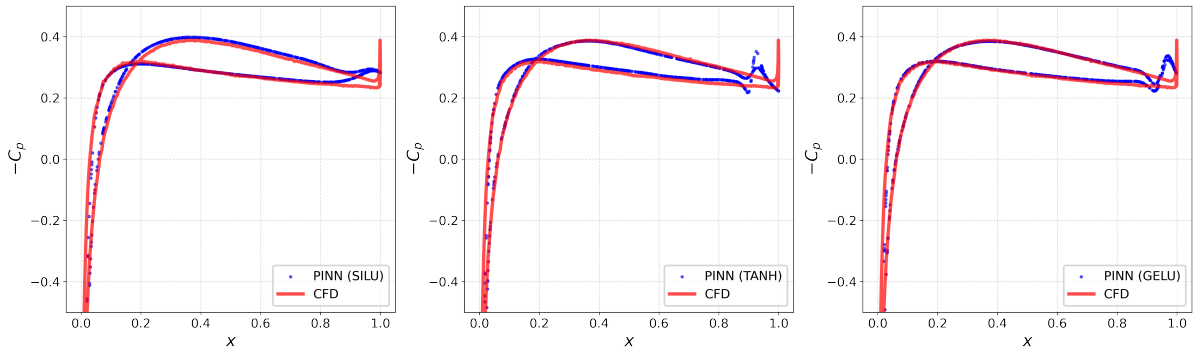


Figure 13: C_p prediction using three different Activation Functions

to provide a better profile match than SILU.

4 Conclusions

We report a consistent PINN framework that can be employed for airfoil flow predictions. However, we note that despite a reasonable match in integrated aerodynamic parameters with the CFD data, significant differences in the prediction of flow near the trailing edge are found. Specifically, the behaviour near the trailing edge is unphysical and needs to be corrected before the PINN model can be further employed for shape optimization. This issue requires further investigation and perhaps a close look at how boundary conditions are implemented in PINN. The study also highlights the significance of detailed and rigorous examination of PINN predictions for complex flows as integrated parameters may be forgiving to give misleading conclusions.

5 Acknowledgement

Prashant would like to express his sincere gratitude to the Prime Minister's Research Fellowship (PMRF) for their invaluable support in funding and facilitating research. The authors have made use of IIT Kanpur's computational facility and AIRAWAT for the simulations. The authors would also like to thank Dr. Deep Ray for his invaluable insights.

References

[1] Jichao Li, Xiaosong Du, and Joaquim RRA Martins. Machine learning in aerodynamic shape optimization. *Progress in Aerospace Sciences*, 134:100849, 2022.

[2] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[3] Y Sun, U Sengupta, and M Juniper. Physics-informed deep learning for simultaneous surrogate modeling and pde-constrained optimization of an airfoil geometry. 2023.

[4] G. Bebis and M. Georgiopoulos. Feed-forward neural networks. *IEEE Potentials*, 13(4):27–31, 1994.

[5] Oliver Hennigh, Susheela Narasimhan, Mohammad Amin Nabian, Akshay Subramaniam, Kaustubh Tangsali, Zhiwei Fang, Max Rietmann, Wonmin Byeon, and Sanjay Choudhry. Nvidia simnet™: An ai-accelerated multi-physics simulation framework. In *International conference on computational science*, pages 447–461. Springer, 2021.

[6] Ali Al-Safwan, Chao Song, and Umair bin Waheed. Is it time to swish? comparing activation functions in solving the helmholtz equation using physics-informed neural networks. *arXiv preprint arXiv:2110.07721*, 2021.

[7] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.