Hybrid Training of Physics-Informed Neural Networks with Approximate Supervision

Mark Sperry* and John T. Hwang†
University of California San Diego, 9500 Gilman Dr, La Jolla, CA 92093

A major bottleneck in multidisciplinary design optimization (MDO) is its high computational cost, with large-scale problems often requiring hours or days to complete. To address this issue, engineers turn to low-fidelity models or data-driven surrogate models such as neural networks. However, while these approaches significantly reduce runtime, they often sacrifice accuracy, which can impact the quality of optimization results. Physics-informed neural networks (PINNs) offer a promising middle ground by embedding physical governing equations directly into the training process, enabling both quick evaluation and high accuracy. Despite their potential, traditional PINNs often converge to non-physical solutions, limiting their use in real-world applications. Thus, many applications of PINNs in existing literature are limited to academic problems. In this paper, we demonstrate accurate flow field predictions on a three-dimensional domain around a complex aircraft geometry using PINNs, achieving a nondimensional PDE residual of less than 5×10^{-3} and an average surface tangency error of less than 2°. We avoid common failure modes associated with PINNs by leveraging inexpensive approximate data to help guide the network to a physical solution, qualitatively matching the reference solution. We also improve convergence speed compared to standard neural networks by encoding geometric information in the neural network architecture. We believe this demonstration on a practical application takes a step towards the use of PINN-based models in MDO workflows.

I. Introduction

Multidisciplinary design optimization (MDO) is a formal engineering design methodology that can consider complex interactions between disciplines through numerical optimization. Large-scale MDO is the restriction to high-dimensional problems with tens or more design variables. When using high-fidelity models, large-scale MDO algorithms can take many hours or even days [1], e.g., in the case of aircraft design using computational fluid dynamics (CFD) solvers. To mitigate this issue, engineers typically rely on one of two strategies: 1) low-fidelity models with simplified physics or 2) surrogate models.

Low-fidelity models are often efficient due to their reduced number of calculations, but they are subject to higher mathematical modeling error and/or discretization error. Surrogate models, especially neural networks, are similarly fast to evaluate but suffer from training error in addition to other difficulties such as high prediction error when extrapolating and high training time with surrogate models with a large number of parameters. While these strategies offer advantages in speed, the sacrifice in accuracy is often difficult to justify for MDO. Thus, there is a demand for approaches offering both the accuracy of high-fidelity models while being computationally efficient.

Recently, physics-informed neural networks (PINNs) have emerged as a promising alternative: given a sufficiently expressive network, they can (theoretically) approximate arbitrarily accurate continuous solutions to machine precision by embedding governing-physics-based equations directly into their loss function without the need for labeled data [2]. Thus, their appeal for MDO is clear as, once trained, the evaluation times of PINNs are as small as those of neural networks obtained using conventional supervised learning.

However, research on PINNs is not yet mature, having gained popularity only in the last few years. Despite their promise, standard PINNs have struggled to find their place as an option in real-world applications due to their poor convergence properties in practice [3]. Convergence speed is often low or the network converges to physically inaccurate solutions [4]. This is due to their difficulty in learning stiff PDEs with sharp gradients, often converging to a solution where large parts of the domain are constant. For example, applications of PINNs to predict flow around airfoils are generally limited to low Reynolds numbers in literature and are limited to 2D [5, 6].

*PhD Student, Department of Mechanical and Aerospace Engineering, AIAA student member
†Associate Professor, Department of Mechanical and Aerospace Engineering, AIAA senior member

The goal of this work is to help fill this research gap by exploring strategies to improve the convergence for PINNs to get to MDO-level accuracy without requiring substantial data generation costs. Further, we also aim to demonstrate PINNs on a complex real-world aerodynamic design problem, extending beyond simplified and notional problems commonly studied in the literature.

In this paper, we accurately predict flow around an aircraft geometry using a geometry-aware neural network, achieving an average nondimensional PDE residual of less than 5×10^{-3} and a surface tangency angle error below 2° in a single-case demonstration. The problem involves three-dimensional flow over a blended-wing-body aircraft assuming steady, inviscid, and incompressible flow conditions. We use data from an inexpensive, approximate model to guide the network to a physical solution, avoiding a common failure mode of traditional PINNs. Through direct comparison with purely data-driven models, self-supervised PINNs, and the approximate physics model itself, we show that the hybrid-trained network yields a solution that closely matches the reference panel method solution. This demonstration on a complex practical application takes a step towards the use of PINN-based models in MDO pipelines.

The paper is organized as follows. Section II provides an overview of the PINN methodology. Section III discusses the model architecture and hybrid training strategy used on the aerodynamic demonstration problem. Finally, section IV presents the demonstration problem on a blended-wing-body aircraft.

II. Background

Physics-informed neural networks (PINNs), initially introduced by Raissi et al. [2], are a class of neural network models designed to approximate the solution of partial differential equations (PDEs) by directly incorporating physics equations into the network training objective. Thus, PINNs are unique in that they can be trained in a self-supervised manner, not explicitly requiring labeled data which can be expensive to generate.

We consider a general PDE of the form

$$\mathcal{F}[u(\mathbf{x})] = 0, \quad \mathbf{x} \in \Omega, \tag{1}$$

subject to boundary conditions

$$\mathcal{B}[u(\mathbf{x})] = 0, \quad \mathbf{x} \in \partial\Omega, \tag{2}$$

where $\Omega \subset \mathbb{R}^d$ is a d-dimensional spatial domain with boundary $\partial\Omega$, $u(\mathbf{x})$ is the PDE solution as a function of space, $\mathcal{F}$ is a differential operator, and $\mathcal{B}$ enforces boundary conditions such as Dirichlet, Neumann, or Robin boundary conditions. For simplicity, we consider steady-state PDEs in this section although an additional time dimension can be added to Ω for time-dependent problems.

To approximate the solution $u(\mathbf{x})$ to equation 1 and 2, PINNs use a neural network defined as $\hat{u}(\mathbf{x}, \theta)$, where θ are trainable parameters. The network is trained to find optimal parameters θ^* such that $\hat{u}(\mathbf{x}, \theta^*) \approx u(\mathbf{x})$ by minimizing a composite loss function that enforces PDE and boundary constraints. The standard loss function of a traditional self-supervised PINN is defined as:

$$\mathcal{L}_{ssl}(\theta) = \lambda_F \, \mathcal{L}_F(\theta) + \lambda_B \, \mathcal{L}_B(\theta), \tag{3}$$

where the PDE loss term is defined as

$$\mathcal{L}_F(\theta) = \frac{1}{N_F} \sum_{i=1}^{N_F} \left\| \mathcal{F}[\hat{u}(\mathbf{x}_i, \theta)] \right\|^2, \tag{4}$$

with N_F PDE collocation points $\{\mathbf{x}_i\}_{i=1}^{N_F} \subset \Omega$, and the boundary condition loss term is defined as

$$\mathcal{L}_B(\theta) = \frac{1}{N_B} \sum_{j=1}^{N_B} \left\| \mathcal{B}[\hat{u}(\mathbf{x}_j, \theta)] \right\|^2, \tag{5}$$

with N_B boundary collocation points $\{\mathbf{x}_j\}_{j=1}^{N_B} \subset \partial\Omega$. The differential operator $\mathcal{F}[\hat{u}(\mathbf{x}, \theta)]$ can be evaluated exactly to machine-precision using automatic differentiation (AD), eliminating the need for manual derivation and implementation of derivatives. The traditional PINN architecture can be seen in figure 1.

If data points $\{(\mathbf{x}_k, u_k)\}_{k=1}^{N_D}$ are available, an additional data loss term may be incorporated:

$$\mathcal{L}_D(\theta) = \frac{1}{N_D} \sum_{k=1}^{N_D} \left\| \hat{u}(\mathbf{x}_k, \theta) - u_k \right\|^2. \tag{6}$$

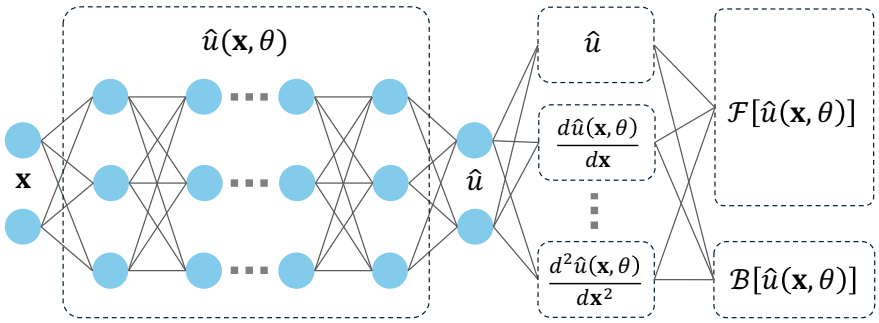


Fig. 1 The traditional self-supervised PINN architecture. State derivatives are computed by using automatic differentiation on the network $\hat{u}$.

The total loss then becomes

$$\mathcal{L}(\theta) = \lambda_F \mathcal{L}_F(\theta) + \lambda_B \mathcal{L}_B(\theta) + \lambda_D \mathcal{L}_D(\theta). \tag{7}$$

Neural network parameters θ^* are obtained by solving the optimization problem:

$$\theta^* = \arg\min_{\theta} \mathcal{L}(\theta). \tag{8}$$

In practice, the self-supervised PINN formulation often face significant challenges in practical applications, particularly with slow convergence or inaccurate predictions. Krishnapriya et al. demonstrated that PINNs struggle to solve complex problems or PDEs with large PDE coefficients, even when the network has enough expressive capacity [3]. To address this, they proposed the use of curriculum learning and sequence-to-sequence learning that gradually adds complexity to the training, allowing the network to slowly learn difficult solutions. Another common failure is the tendency for PINNs to converge to trivial solutions where parts of the domain satisfy the PDE in a non-physical way [7]. Further, by analyzing PINNs through neural tangent kernel theory, Wang et al. revealed their issues with spectral bias and loss imbalance during training [8]. Similarly, it has been shown that the composite loss function combining PDE residuals and boundary condition errors produces a non-convex loss landscape for complex problems, severely limiting training convergence [4]. In addition to solution accuracy concerns, PINNs have been found to converge slowly, losing out to traditional finite element analysis solvers [9]. Finally, McGreivy and Hakim argued that there was significant overoptimism on the idea of using machine learning for solving fluids problems by conducting a thorough evaluation of existing literature [10].

In response to these issues, a variety of strategies have been proposed to improve the robustness of PINNs. One major source of inaccuracy arises from low sampling resolution of PDE collocation points around regions of complex solution behavior [11]. To address this, adaptive collocation sampling methods that move collocation points during training have been proposed. These techniques typically move points based on their PDE residuals [12, 13] or solution gradients [14], allowing the network to see sharp areas of the solution.

Beyond sampling strategies, more fundamental changes to the PINN architecture to improve training robustness have been suggested. One well-known modification includes swapping out the soft-constrained PINN boundary conditions for hard-constrained ones, where boundary conditions are directly embedded into the structure of the model instead of enforced through soft penalty terms [15, 16]. While effective, this comes at the expense of generality and modularity, resulting in greater problem-specific implementation effort, since boundary conditions can vary significantly across application problems. Further, Jagtap and Karniadakis explored the use of adaptive activation functions, incorporating a scalable hyper-parameter that modifies the activation function, resulting in increased convergence speed and accuracy [17]. Another architectural change includes the conservative PINN framework where the domain is partitioned into subdomains with individual networks that are trained separately, and interface continuity is enforced between them to maintain consistency between boundaries [18]. Reformulations of the problem itself have also been studied. For example, using the variational form of the PDE as opposed to the strong form to train the PINN have been shown to increase accuracy and reduce training cost [19].

Another well-studied technique is that of loss-balancing, which addresses the issue with multi-task learning associated with PINNs [20]. Xiang et al. proposed a method to adaptively update weights at every epoch based on the maximum likelihood estimation [21]. Bischof and Kraus introduced ReLoBRaLo, a method to update weights based on

loss terms and gradients [22]. Finally, a learning rate annealing algorithm based on gradient statistics was also proposed [23].

Finally, another concept that has been explored is the idea of PINNs in the context of multifidelity modeling. Aliakbari proposed a technique where neural networks trained on low-fidelity data can be combined with physics-informed losses using transfer training [24]. Another approach includes combining deep operator networks representing different fidelities together based on low-fidelity data and physics-informed neural networks [25].

However, despite these advancements, practical use of PINNs still remains limited due to their reliability issues discussed earlier. Most existing studies in literature are mainly limited to simplified academic PDEs or geometrically simple domains. For example, fluid flow around an airfoil is commonly needed in MDO, but the majority of PINNs applied to airfoil analysis are limited to the two-dimensional case with relatively low Reynolds number [5, 6]. Although a few high-speed problems in literature have been explored, they require significantly intrusive modifications to the model, such as mesh transformation, or they are solved on a simplified geometry [26–28].

III. Model Architecture and Hybrid Training with Approximate Supervision

In this section, we describe the model architecture and approximate supervision used for the aerodynamic demonstration problem.

As discussed in section I, there is demand for solvers that are both simultaneously fast and accurate in MDO. However, well-used techniques to mitigate this like low-fidelity analytical models or data-driven surrogate models such as neural networks often fail to deliver the accuracy required for optimization. Purely self-supervised PINNs, as detailed in the previous section, offer a promising middle ground, but often do not converge to good solutions in practice.

The standard remedy of adding a data error term as per equation 6 can help the PINN converge to more physically accurate solutions. However, this approach comes with its own significant challenges. Firstly, generating high quality data that PINNs can leverage during training can take a significant amount of time due to the high computational cost of high-fidelity models. Secondly, they are often labor-intensive to set up, requiring domain knowledge. For example, using a high-fidelity CFD code for data generation can take hours to generate and set up.

To mitigate these issues, we adopt the use of a hybrid learning strategy for PINNs using cheap approximate data as opposed to expensive accurate data, aiming to retain the accuracy benefits of PINNs. We substitute the data loss term $\mathcal{L}_D$ in equation 7 with an approximate data loss term to get a hybrid loss:

$$\mathcal{L}(\theta) = \lambda_F \, \mathcal{L}_F(\theta) + \lambda_B \, \mathcal{L}_B(\theta) + \lambda_{approx} \, \mathcal{L}_{approx}(\theta), \tag{9}$$

where $\mathcal{L}_{approx}$ is identical to the data loss term shown in equation 6 with approximate data and λ_{approx} is its respective weighting term.

Determining the loss weights is dependent on several factors including the accuracy of the data and the scaling of the specific problem. For the demonstration provided in this paper, we choose weights of $\lambda_F = 0.8$, $\lambda_{approx} = 0.2$ and $\lambda_B = 1.0$ based on training sweeps ran with different PDE and loss weights. Additionally, the sampling strategy of approximate data points is also important. A lower concentration of samples can be used in areas where the approximate data is known to be especially inaccurate or where the PINN has trouble resolving. These considerations are left for future investigation.

Another common issue with PINNs is their difficulty learning sharp solution features as discussed in the previous section. Motivated by this, our PINN architecture for the predictor $\hat{u}(\mathbf{x}, \theta)$ includes three key components. First, we have a standard multilayer perceptron (MLP) we train θ for, using the tanh activation function using 4 hidden layers of 250 neurons each. Second, we add an implicit geometry representation using a signed distance function (SDF) and its derivatives as a function of the spatial point $\mathbf{x}$. Third, we add Fourier feature embeddings of the spatial inputs and the SDF. This architecture is seen in figure 2.

Since sharp solution features often occur near solid boundaries, we provide geometric information about the surface to the network via an SDF $d(\mathbf{x})$ which represents the shortest distance from a point $\mathbf{x}$ in space to the surface. The value is positive outside the surface and negative inside.

In addition to surface distance, we also provide information about the surface direction at each point in space to the network. Specifically, we give the network information about the normal direction of the surface, which can help the network satisfy flow tangency conditions. To do this, we can leverage the following property of the SDF:

$$\mathbf{n}(\mathbf{x}) = \nabla d(\mathbf{x}), \tag{10}$$

where $\mathbf{n}(\mathbf{x})$ is a unit normal vector pointing to $\mathbf{x}$ from the closest point on the surface.

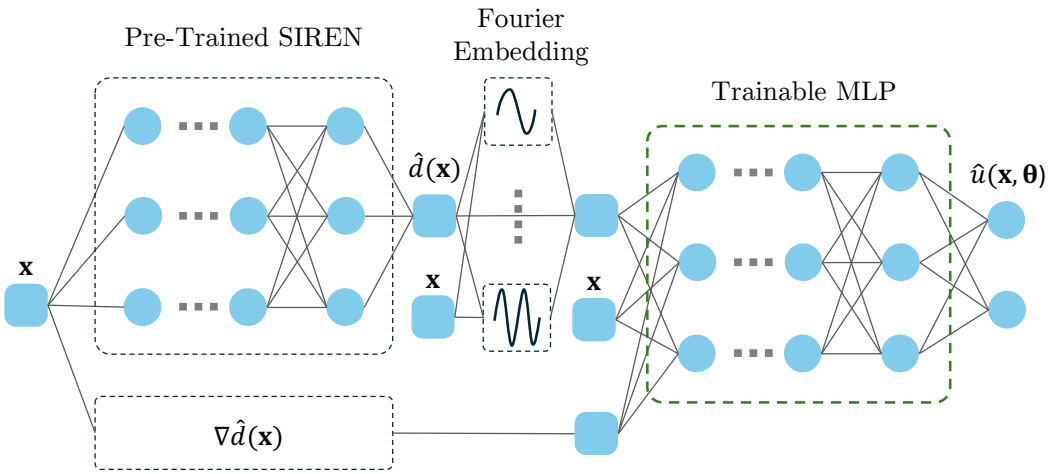


Fig. 2 Architecture of the geometry-aware neural network $\hat{u}$. The signed-distance function approximated by a pretrained Sinusoidal Representation Network is frozen during PINN training.

While SDFs are often analytically defined for simple geometries, we approximate the SDF using a neural network $\hat{d}(\mathbf{x})$ trained offline. Specifically, we train a Sinusoidal Representation Network (SIREN) [29], which uses sine activation functions of the form $sin(\omega x)$ at each layer. SIRENs have been found to be well-suited for representing geometric information and high-frequency functions, and have desirable properties for PINNs. Notably, their derivatives preserve the sinusoidal form, which is suitable for PINNs where second- or third-order derivatives of the field are commonly required. We pretrain this network using 3 hidden layers with 150 neurons each and a constant frequency $\omega = 30$ for all layers. The training dataset consists of randomly sampled points inside and outside the geometry, labeled with precomputed signed distances. A higher density of points was sampled around the surface to increase accuracy around the geometry. The training process is straightforward, only relying on labeled data and predicting a single scalar value. Further, high accuracy is not critical as the SDF serves primarily to guide the PINN training and the loss terms are what governs the correctness of the trained PINN.

Finally, we concatenate the spatial coordinates $\mathbf{x}$ with $\hat{d}(\mathbf{x})$ to form an augmented input vector $\mathbf{x}_f = [\mathbf{x}, \hat{d}(\mathbf{x})]$, which is then embedded via Fourier feature mappings and fed as inputs to the trainable MLP to enable the network to better represent high-frequency variations in the target function. Specifically, we compute

$$[\sin(2\pi f_k x_f), \cos(2\pi f_k x_f)] \text{ for } k = 1, ..., K, \tag{11}$$

where $f_k \in 2, 4, 8, 16, 32$ are selected frequencies. This technique helps reduce spectral bias in neural networks by explicitly providing high-frequency components of the input. Finally, we also include the predicted unit surface normal vectors $\nabla\hat{d}(\mathbf{x})$ as separate inputs to the MLP.

IV. Application to 3D Flow Prediction around an Aircraft

A. Problem Definition

In this section, we present a demonstration of PINNs on the problem of modeling the air flow over a blended-wing-body aircraft (figure 3).

The flow conditions are $U_\infty = 212$ m/s, $\alpha = 0°$, $p_\infty = 30,100$ Pa, and $\rho_\infty = 0.459$ kg/m^3 in the far-field. We solve the governing equations under the assumptions of steady-state, inviscid, and incompressible flow. In nondimensionalized form, the steady three-dimensional incompressible Euler equations are given by:

$$\mathbf{u} \cdot \nabla\mathbf{u} + E_u \nabla p = 0, \qquad \nabla \cdot \mathbf{u} = 0 \tag{12}$$

where $\mathbf{u} = (u, v, w)$ is the dimensionless velocity field, p is the dimensionless pressure, and $E_u = \dfrac{p_0 - p_\infty}{\rho_\infty U_\infty^2}$ is the Euler number. We use a characteristic length of 10 meters, a reference velocity equal to the freestream velocity, and

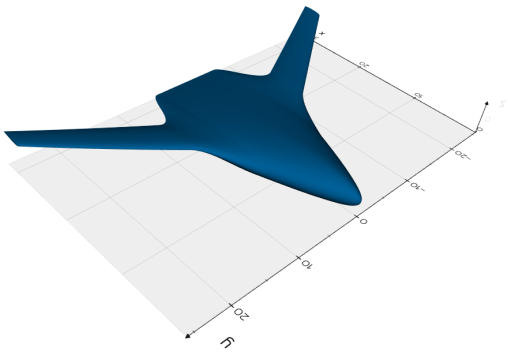


Fig. 3 The blended-wing-body configuration geometry.

a stagnation pressure p_0 of 41,500 Pa for the nondimensionalization of the PDE. We make the incompressible flow assumption to simplify the problem for this initial investigation, recognizing that future work will need to incorporate compressibility given the high cruise Mach numbers that are typical for an aircraft of this type.

B. PINN Training

The computational domain is defined as a three-dimensional rectangular volume enclosing the aircraft geometry. We exploit symmetry, only predicting flow in the $y > 0$ direction, enforcing symmetry constraints along the $y = 0$ plane.

We train the PINN using the hybrid loss term according to equation 9, composed of the PDE residual, boundary condition error, and approximate data error term. The prediction model $\hat{u}$ is as described in the previous section, predicting flow states $\mathbf{u}$ and p as a function of spatial coordinates $\mathbf{x}$. The PDE loss is computed as the mean-squared error of the residuals of the incompressible Euler equations according to equation 12 evaluated on the predicted solution $\hat{u}$. We sample approximately 600,000 PDE collocation points in the rectangular domain, with a higher density of points in the region close to the aircraft surface to better capture flow features near the wall. Finally, to partially address the concerns with fixed sampling addressed in section II, we apply random perturbations to the collocation points at every training epoch.

The boundary condition loss consists of three terms: far-field Dirichlet constraints, symmetry plane constraints, and flow tangency constraints on the aircraft surface. The far-field constraints are imposed using the standard data loss calculation according to equation 6, where the known data is the far-field freestream values. The symmetry constraint enforces no flow and smooth derivatives across the $y = 0$ plane. In total, there are approximately 27k points uniformly sampled across the six boundary domain walls.

The flow tangency condition enforces zero flow through the aircraft surface by minimizing the normal component of the predicted velocity at specific points on the body surface. Specifically, we minimize the squared dot product of the predicted velocity vector with the local outward-facing unit surface normal:

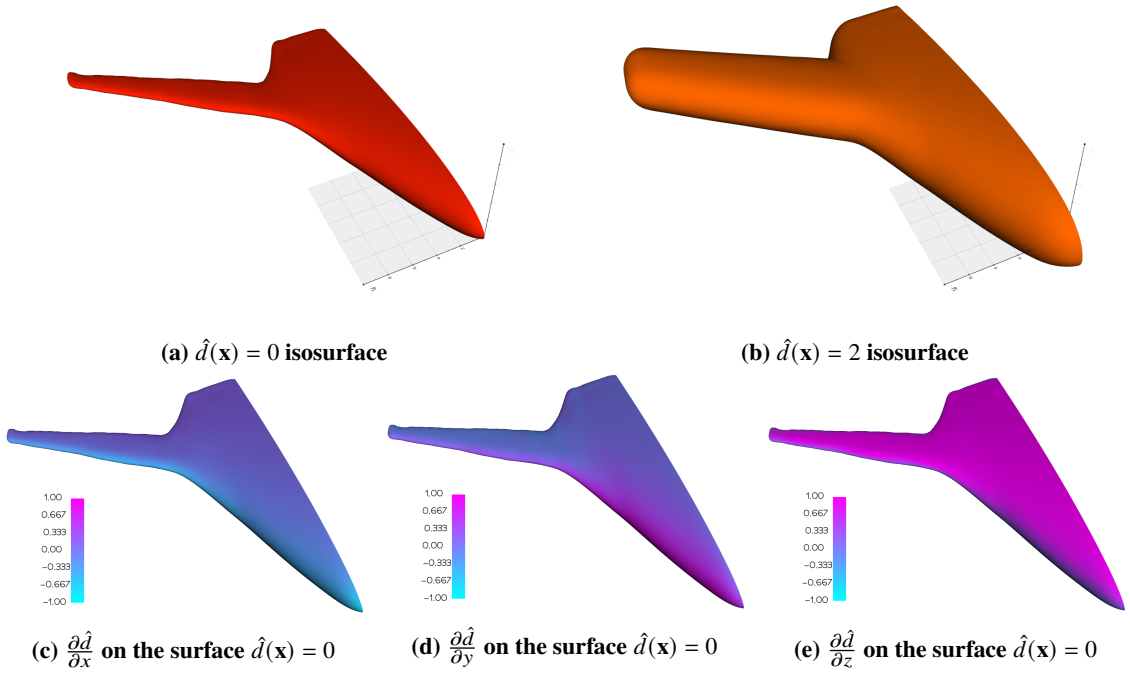
$$\mathcal{L}_{\text{tan}} = \frac{1}{N_{\text{tan}}} \sum_{i=1}^{N_{\text{tan}}} (\mathbf{u}_i \cdot \mathbf{n}_i)^2 , \tag{13}$$

where N_{tan} represents the number of points sampled on the surface. For this problem, we sample approximately 27k points on the surface.

The approximate data used for training is generated using VortexAD, an open-source panel method solver based on potential flow theory [30]. VortexAD solves for inviscid, incompressible flows around solid bodies by discretizing their surfaces with constant-strength source and doublet panels. A single forward evaluation at a flow condition takes several seconds to execute. However, the inaccuracy comes from discretization error and the low-order formulation, which leads to discontinuities across panels and results in erroneous velocities when evaluating points near or on the surface [31]. Specifically, with a low-order panel method, there can be significant violations of the flow-tangency conditions in the regions between the points at which those conditions are enforced. The approximate pressures and velocities on the aircraft surface are shown in figures 6a and 9 respectively. At specified evaluation points in space, we compute induced velocities from the source and doublet distributions to generate velocity data. Pressures at equivalent points are computed using Bernoulli's principle and freestream conditions. The source and doublet distributions along the surface

are evaluated using a mesh resolution of 8,000 panels. Velocity and pressure data is generated at roughly 200,000 spatial points in the domain. Although a higher density of points are used around the aircraft, we exclude points close to the surface due to the aforementioned erroneous velocities near the surface. As discussed in the previous section, we use loss weights of $\lambda_F = 0.8$, $\lambda_{approx} = 0.2$ and $\lambda_B = 1.0$

The SDF described in section III was pretrained on 400,000 points in space with precomputed SDF labels. The SDF training data also includes points on the aircraft surface with an SDF value of zero. The trained SDF $\hat{d}(\mathbf{x})$ and surface normal vectors for the blended-wing-body geometry computed from equation 10 are shown in figure 4.



(a) $\hat{d}(\mathbf{x}) = 0$ **isosurface** **(b)** $\hat{d}(\mathbf{x}) = 2$ **isosurface**

(c) $\frac{\partial \hat{d}}{\partial x}$ **on the surface** $\hat{d}(\mathbf{x}) = 0$ **(d)** $\frac{\partial \hat{d}}{\partial y}$ **on the surface** $\hat{d}(\mathbf{x}) = 0$ **(e)** $\frac{\partial \hat{d}}{\partial z}$ **on the surface** $\hat{d}(\mathbf{x}) = 0$

Fig. 4 The geometry surface prediction $\hat{d}(\mathbf{x})$. The network is pretrained prior to the PINN training.

The network is trained using the Adam optimizer with weight decay regularization. We use 20 randomized mini-batches per epoch and leave 10% of the data as a test set. The implementation is written entirely in JAX and trained using fp32 precision with GPU acceleration. For this problem, we used a static learning rate of 1e-4 as we observed slower convergence when a standard decaying learning rate schedule was used.

The PINN training loss history, evaluated on the test set, is shown in figure 5. We compare the convergence speed of four different $\hat{u}$ model configurations to assess the benefit of each model component discussed in section III: a baseline standard MLP, an MLP with only Fourier feature embeddings, an MLP with only geometric features from $\hat{d}(\mathbf{x})$, and finally the full architecture with an MLP in addition to the Fourier features and $\hat{d}(\mathbf{x})$ as described previously. Among these, the full architecture incorporating the geometry and Fourier features achieves the fastest convergence to the same test loss compared to the other configurations. However, each epoch evaluation still takes approximately 1.5 seconds on a NVIDIA GeForce RTX 3060 GPU, indicating the need for further convergence speed improvements.

C. Results

The hybrid PINN achieves an average nondimensional PDE error of less than 5×10^{-3} evaluated on 640,000 out-of-sample points distributed across two uniform grids: one across the entire domain, and a second with finer resolution around the aircraft body. Further, it achieved an average surface tangency error of less than 2° degrees on 12,000 surface points also outside the training set, indicating accurate enforcement of boundary conditions on the aircraft.

Figure 6 presents a qualitative comparison of the predicted pressure coefficient C_p across four different models to demonstrate the advantages of the hybrid training approach. These include: C_p computed from the induced velocities

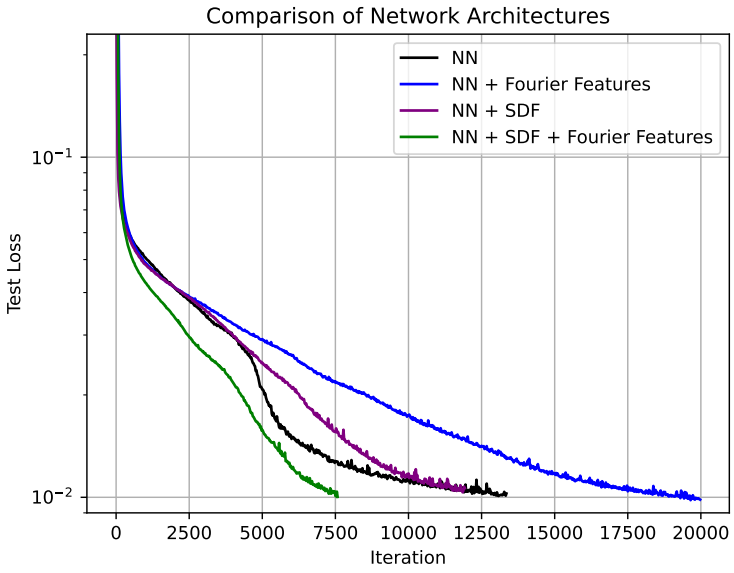


Fig. 5 Comparison of different network architectures on training convergence.

also used as the approximate training data (figure 6a), a neural network trained solely on the approximate data (figure 6c), a self-supervised PINN trained exclusively on the PDE and boundary condition constraints (figure 6e), and the hybrid PINN, which integrates the approximate data, the PDE, and boundary condition constraints (figure 6g). The C_p for each model is computed using

$$C_p = \frac{p - p_\infty}{\frac{1}{2}\rho U_\infty^2}, \tag{14}$$

where p is the physical pressure.

As the reference solution, we compute a baseline C_p distribution using the panel method solution evaluated at 8,000 panel centers. The surface tangency condition is exactly enforced at the panel centers as it uses the fundamental definition of the doublet strengths in relation to the flow field. These values can only be evaluated at the panel centers and thus was not seen in the training data. On the other hand, the induced velocities, used as approximate data, has singularities due to the constant strength panels, causing noisy predictions near the surface. We note that the reference solution still contains discretization error and thus is not considered the ground truth solution.

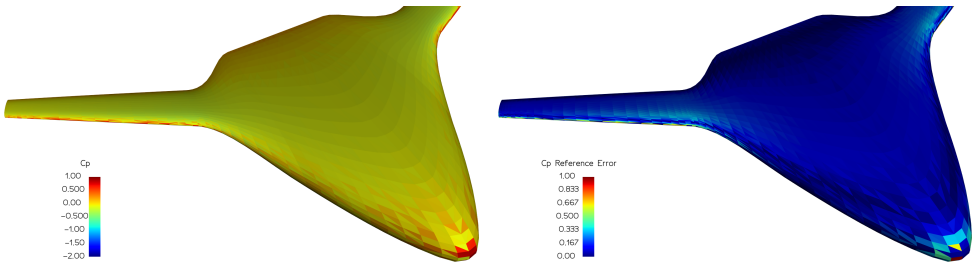
The induced-velocity-based solution exhibits clear artifacts and high frequency noise at the leading edge due to the aforementioned singularities. Training a neural network on this dataset somewhat regularizes the output both due to the lower data density near the surface and the natural inherent smoothing of neural networks. However, it fails to accurately capture the sharp changes in pressure gradients along the stagnation points.

In contrast, the purely self-supervised PINN predicts an almost uniform C_p across the aircraft surface, a well-documented and expected failure mode of PINNs in the absence of data as discussed extensively in section II. The corresponding flow field around the aircraft predicted by the self-supervised PINN can be seen in figure 8, where it can be observed that it predicts near-zero flow around the surface.

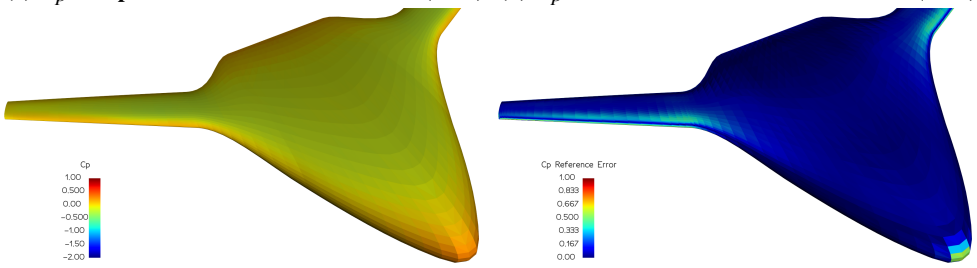
The hybrid PINN, however, produces a C_p distribution that qualitatively aligns well with the reference panel method solution. By using the approximate induced velocity data during training, the hybrid approach allows the network to satisfy the governing equations and boundary conditions without falling back to a degenerate solution, even for a complex case like this demonstration. Nevertheless, inconsistencies with respect to the reference solution still remain, particularly around the leading edge regions with high surface curvature.

Further, all three neural networks are capable of evaluating the C_p surface distribution in around 3×10^{-4} seconds, a significant speed up compared to the 15 seconds required for the panel code as expected. We note this is a preliminary comparison as the networks are only able to evaluate a single flow condition with a fixed geometry unlike the panel code. However, we predict that a larger network with parameterization will not add significant evaluation time.

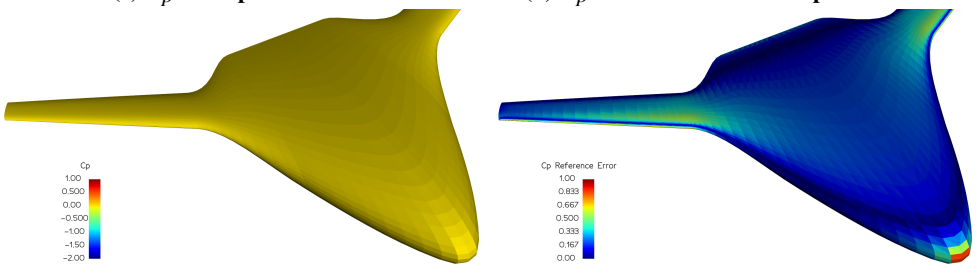
Figure 7 presents velocity and pressure fields for the hybrid PINN in the y-plane at various points along the spanwise direction of the aircraft from the centerline towards the wingtip. The hybrid PINN predicts the expected acceleration



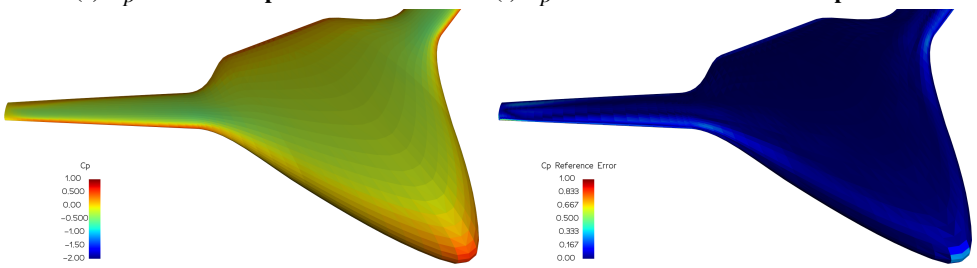
(a) C_p computed from induced velocities (data) (b) C_p ref. error from induced velocities (data)



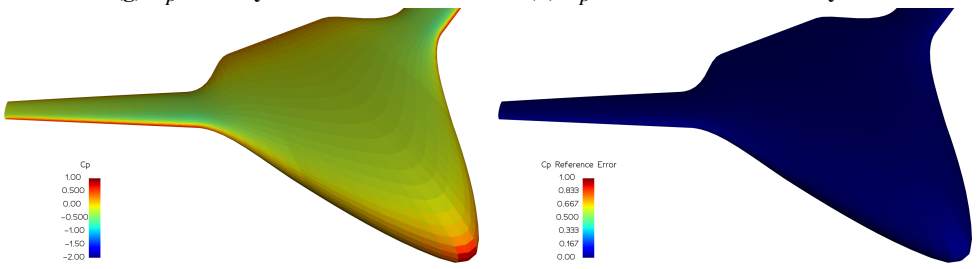
(c) C_p from pure data NN (d) C_p reference error from pure data NN



(e) C_p from self-supervised PINN (f) C_p reference error from self-supervised PINN



(g) C_p from hybrid PINN (h) C_p reference error from hybrid PINN



(i) Reference C_p from panel method (j) Reference C_p error from panel method

Fig. 6 A comparison of C_p to a reference solution. The reference solution is computed using the panel method, where C_p is evaluated at the centers of all 8,000 panels.

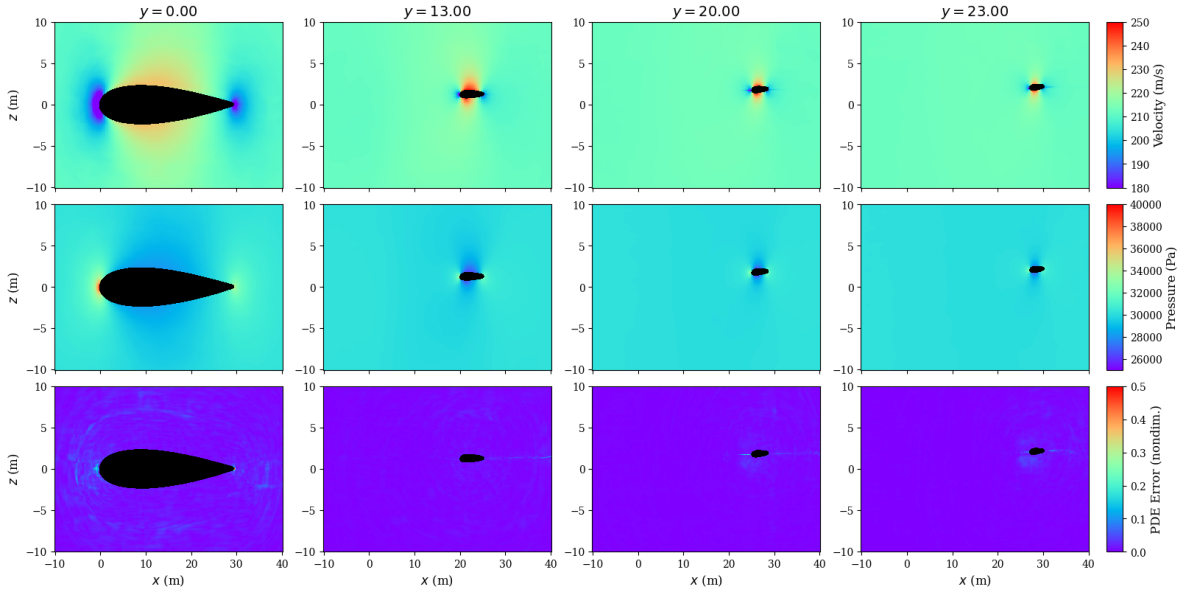


Fig. 7 Predicted flow field around the aircraft at different spanwise locations using the hybrid PINN.

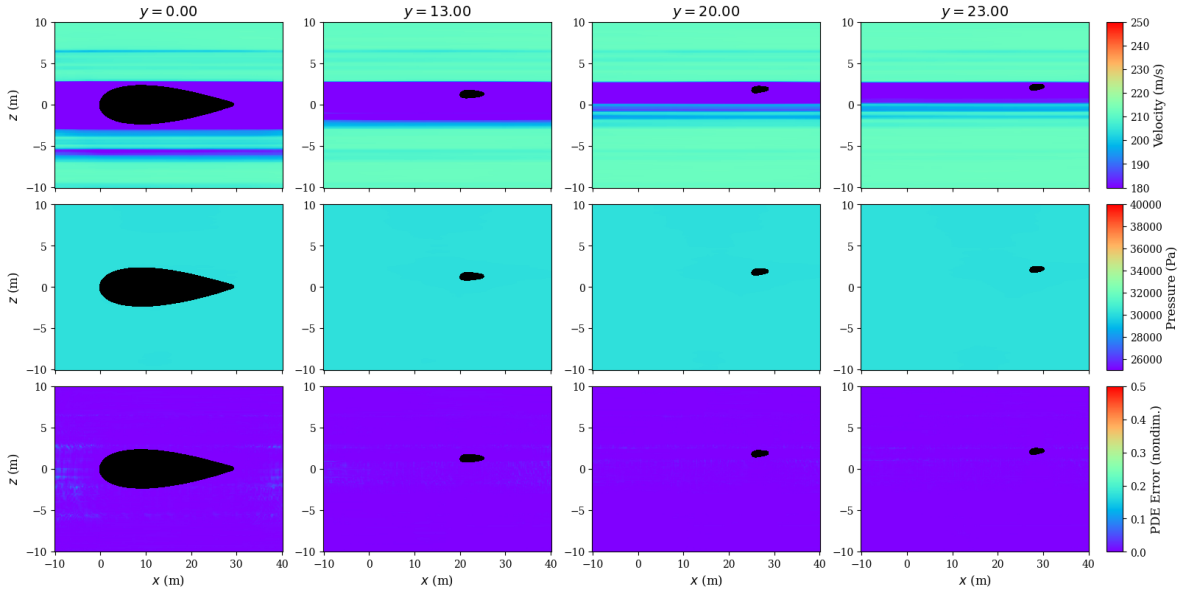


Fig. 8 Predicted flow field around the aircraft at different spanwise locations using a self-supervised PINN.

over the body and its associated pressure changes in the flow even at the wingtip with sharp changes in the solution. However, there are artifacts in the PDE solution especially near the stagnation points of the wing, indicating the PINN still cannot fully resolve flow in that area. This is consistent with the discrepancies observed in the C_p distributions near the leading edge, where deviations from the reference solution are most pronounced. Further training and reduction of the loss function did not lead to significant decreases to the PDE error or changes to the flow predictions.

Figure 8 showcases the predicted velocity and pressure fields from the self-supervised PINN at the same evaluation points. We observe a direct visualization of one of the well-known failure modes of pure PINNs discussed in section II. The velocity and pressure fields converge to nearly static values in large parts of the domain, satisfying the PDE at most collocation points, confirming the tendency for PINNs to collapse without data. This also confirms that even using cheap approximate data can play a critical role in guiding the network towards a meaningful solution.

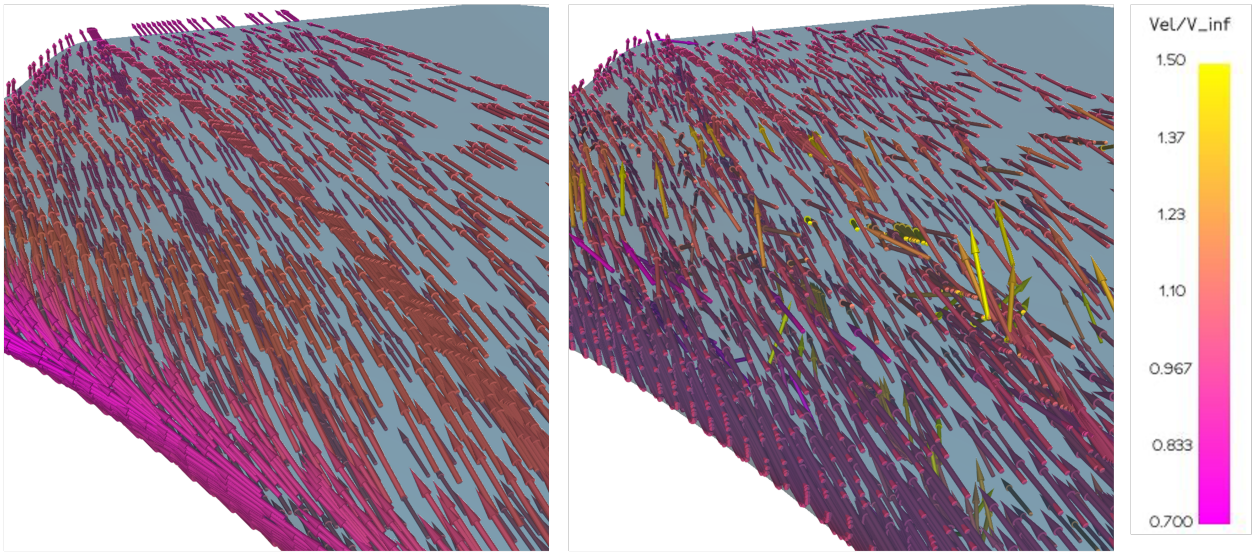


Fig. 9 Surface velocity vector comparison of the hybrid PINN prediction (left) and the induced velocity (right) at identical surface points near the transition region.

Finally, figure 9 compares the hybrid PINN's surface velocities with those from the panel method induced velocities that served as the approximate supervision. Firstly, we can observe that the discontinuity-induced velocity spikes from the panel method induced velocity have been eliminated by the hybrid PINN, smoothing out the velocities to be tangent along the surface. Further, we notice the induced velocity violating the surface tangency constraint along the leading edge, with the flow going through the surface in the streamwise direction. The hybrid PINN correctly diverts the flow away from the fuselage. These qualitative trends demonstrate the regularizing effects of the PDE and boundary condition constraints not possible with pure data.

In summary, we demonstrated a qualitatively successful three-dimensional aerodynamic application of PINNs on an industrially relevant problem. The use of hybrid training using approximate data demonstrated significant qualitative improvements over the pure data and pure physics training methods. By adding approximate data, the PINN was able to avoid the degenerate solutions that the traditional self-supervised PINN converged to.

V. Conclusion

In this paper, we demonstrated the application of PINNs on a practical application not commonly seen in PINN literature: predicting flow around a complex three-dimensional geometry at high speed using the incompressible Euler equations. We achieved a mean nondimensional PDE error of less than 5×10^{-3} and an average surface tangency error less than 2°. We used a pretrained SDF network and a Fourier feature embedding layer to improve optimization convergence. Further, we adopted a hybrid training strategy where induced velocities from an open-source panel code was used as approximate data to help avoid degenerate solutions. We showed improved qualitative performance of this approach compared to training on pure data and training on pure PINN, matching surface C_p distribution to a reference solution better and improving prediction compared to the approximate model.

Despite this, several limitations remain. Inaccuracies between the hybrid PINN C_p prediction compared to the reference solution were evident around the leading edge of the aircraft, indicating the need for higher accuracy in these regions. Furthermore, a more rigorous quantitative validation of results is necessary. In addition, training times were still significant even for a fixed geometry case with one operating condition, limiting the current applicability of PINNs for use in MDO.

Acknowledgments

The authors would like to acknowledge the Multidisciplinary Science and Technology Center of the Aerospace Systems Directorate, Air Force Research Laboratory for funding and supporting this effort through the Collaborative Center for Design and Research of Interdisciplinary Systems program. The authors would also like to express their

gratitude to Luca Scotzniovsky for his help in data generation. Finally, the authors acknowledge the use of OpenAI's ChatGPT in rewording parts of this paper to improve readability.

References

[1] Ruh, M. L., Fletcher, A., Sarojini, D., Sperry, M., Yan, J., Scotzniovsky, L., van Schie, S. P., Warner, M., Orndorff, N. C., Xiang, R., Joshy, A. J., Zhao, H., Krokowski, J., Gill, H., Lee, S., Cheng, Z., Cao, Z., Mi, C., Silva, C., Wolfe, L., Chen, J.-S., and Hwang, J. T., *Large-scale multidisciplinary design optimization of a NASA air taxi concept using a comprehensive physics-based system model*, 2024. https://doi.org/10.2514/6.2024-0771, URL https://arc.aiaa.org/doi/abs/10.2514/6.2024-0771.

[2] Raissi, M., Perdikaris, P., and Karniadakis, G., "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, Vol. 378, 2019, pp. 686–707. https://doi.org/https://doi.org/10.1016/j.jcp.2018.10.045, URL https://www.sciencedirect.com/science/article/pii/S0021999118307125.

[3] Krishnapriyan, A., Gholami, A., Zhe, S., Kirby, R., and Mahoney, M. W., "Characterizing possible failure modes in physics-informed neural networks," *Advances in Neural Information Processing Systems*, Vol. 34, edited by M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Curran Associates, Inc., 2021, pp. 26548–26560. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/df438e5206f31600e6ae4af72f2725f1-Paper.pdf.

[4] Basir, S., and Senocak, I., "Critical Investigation of Failure Modes in Physics-informed Neural Networks," , 2022. URL https://arxiv.org/abs/2206.09961.

[5] Sun, Y., Sengupta, U., and Juniper, M., "Physics-informed deep learning for simultaneous surrogate modeling and PDE-constrained optimization of an airfoil geometry," *Computer Methods in Applied Mechanics and Engineering*, Vol. 411, 2023, p. 116042. https://doi.org/https://doi.org/10.1016/j.cma.2023.116042, URL https://www.sciencedirect.com/science/article/pii/S0045782523001664.

[6] Ang, E., and Ng, B. F., *Physics-Informed Neural Networks for Flow Around Airfoil*, ???? https://doi.org/10.2514/6.2022-0187, URL https://arc.aiaa.org/doi/abs/10.2514/6.2022-0187.

[7] Leiteritz, R., and Pflüger, D., "How to Avoid Trivial Solutions in Physics-Informed Neural Networks," , 2021. URL https://arxiv.org/abs/2112.05620.

[8] Wang, S., Yu, X., and Perdikaris, P., "When and why PINNs fail to train: A neural tangent kernel perspective," , 2020. URL https://arxiv.org/abs/2007.14527.

[9] Grossmann, T. G., Komorowska, U. J., Latz, J., and Schönlieb, C.-B., "Can Physics-Informed Neural Networks beat the Finite Element Method?" , 2023. URL https://arxiv.org/abs/2302.04107.

[10] McGreivy, N., and Hakim, A., "Weak baselines and reporting biases lead to overoptimism in machine learning for fluid-related partial differential equations," *Nature Machine Intelligence*, Vol. 6, No. 10, 2024, pp. 1256–1269.

[11] Wu, C., Zhu, M., Tan, Q., Kartha, Y., and Lu, L., "A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks," *Computer Methods in Applied Mechanics and Engineering*, Vol. 403, 2023, p. 115671.

[12] Gao, Z., Yan, L., and Zhou, T., "Failure-Informed Adaptive Sampling for PINNs," *SIAM Journal on Scientific Computing*, Vol. 45, No. 4, 2023, pp. A1971–A1994. https://doi.org/10.1137/22M1527763, URL https://doi.org/10.1137/22M1527763.

[13] Mao, Z., and Meng, X., "Physics-informed neural networks with residual/gradient-based adaptive sampling methods for solving partial differential equations with sharp solutions," *Applied Mathematics and Mechanics*, Vol. 44, No. 7, 2023, pp. 1069–1084.

[14] Hou, J., Li, Y., and Ying, S., "Enhancing PINNs for solving PDEs via adaptive collocation point movement and adaptive loss weighting," *Nonlinear Dynamics*, Vol. 111, No. 16, 2023, pp. 15233–15261.

[15] Lagari, P. L., Tsoukalas, L. H., Safarkhani, S., and Lagaris, I. E., "Systematic Construction of Neural Forms for Solving Partial Differential Equations Inside Rectangular Domains, Subject to Initial, Boundary and Interface Conditions," *International Journal on Artificial Intelligence Tools*, Vol. 29, No. 05, 2020, p. 2050009. https://doi.org/10.1142/S0218213020500098, URL https://doi.org/10.1142/S0218213020500098.

[16] Lu, L., Pestourie, R., Yao, W., Wang, Z., Verdugo, F., and Johnson, S. G., "Physics-informed neural networks with hard constraints for inverse design," , 2021. URL https://arxiv.org/abs/2102.04626.

[17] Jagtap, A. D., Kawaguchi, K., and Karniadakis, G. E., "Adaptive activation functions accelerate convergence in deep and physics-informed neural networks," *Journal of Computational Physics*, Vol. 404, 2020, p. 109136. https://doi.org/10.1016/j.jcp.2019.109136, URL http://dx.doi.org/10.1016/j.jcp.2019.109136.

[18] Jagtap, A. D., Kharazmi, E., and Karniadakis, G. E., "Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems," *Computer Methods in Applied Mechanics and Engineering*, Vol. 365, 2020, p. 113028. https://doi.org/https://doi.org/10.1016/j.cma.2020.113028, URL https://www.sciencedirect.com/science/article/pii/S0045782520302127.

[19] Kharazmi, E., Zhang, Z., and Karniadakis, G. E., "hp-VPINNs: Variational physics-informed neural networks with domain decomposition," *Computer Methods in Applied Mechanics and Engineering*, Vol. 374, 2021, p. 113547. https://doi.org/https://doi.org/10.1016/j.cma.2020.113547, URL https://www.sciencedirect.com/science/article/pii/S0045782520307325.

[20] Rohrhofer, F. M., Posch, S., Gößnitzer, C., and Geiger, B. C., "Data vs. Physics: The Apparent Pareto Front of Physics-Informed Neural Networks," *IEEE Access*, Vol. 11, 2023, p. 86252–86261. https://doi.org/10.1109/access.2023.3302892, URL http://dx.doi.org/10.1109/ACCESS.2023.3302892.

[21] Xiang, Z., Peng, W., Liu, X., and Yao, W., "Self-adaptive loss balanced Physics-informed neural networks," *Neurocomputing*, Vol. 496, 2022, pp. 11–34. https://doi.org/https://doi.org/10.1016/j.neucom.2022.05.015, URL https://www.sciencedirect.com/science/article/pii/S092523122200546X.

[22] Bischof, R., and Kraus, M. A., "Multi-Objective Loss Balancing for Physics-Informed Deep Learning," *Computer Methods in Applied Mechanics and Engineering*, Vol. 439, 2025, p. 117914. https://doi.org/https://doi.org/10.1016/j.cma.2025.117914, URL https://www.sciencedirect.com/science/article/pii/S0045782525001860.

[23] Wang, S., Teng, Y., and Perdikaris, P., "Understanding and Mitigating Gradient Flow Pathologies in Physics-Informed Neural Networks," *SIAM Journal on Scientific Computing*, Vol. 43, No. 5, 2021, pp. A3055–A3081. https://doi.org/10.1137/20M1318043, URL https://doi.org/10.1137/20M1318043.

[24] Aliakbari, M., Mahmoudi, M., Vadasz, P., and Arzani, A., "Predicting high-fidelity multiphysics data from low-fidelity fluid flow and transport solvers using physics-informed neural networks," *International Journal of Heat and Fluid Flow*, Vol. 96, 2022, p. 109002. https://doi.org/https://doi.org/10.1016/j.ijheatfluidflow.2022.109002, URL https://www.sciencedirect.com/science/article/pii/S0142727X22000777.

[25] Howard, A. A., Perego, M., Karniadakis, G. E., and Stinis, P., "Multifidelity deep operator networks for data-driven and physics-informed problems," *Journal of Computational Physics*, Vol. 493, 2023, p. 112462. https://doi.org/https://doi.org/10.1016/j.jcp.2023.112462, URL https://www.sciencedirect.com/science/article/pii/S0021999123005570.

[26] Cao, W., Song, J., and Zhang, W., "A solver for subsonic flow around airfoils based on physics-informed neural networks and mesh transformation," *Physics of Fluids*, Vol. 36, No. 2, 2024. https://doi.org/10.1063/5.0188665, URL http://dx.doi.org/10.1063/5.0188665.

[27] Wassing, S., Langer, S., and Bekemeyer, P., "Physics-Informed Neural Networks for Transonic Flows around an Airfoil," , 2025. URL https://arxiv.org/abs/2408.17364.

[28] Ren, X., Hu, P., Su, H., Zhang, F., and Yu, H., "Physics-informed neural networks for transonic flow around a cylinder with high Reynolds number," *Physics of Fluids*, Vol. 36, No. 3, 2024, p. 036129. https://doi.org/10.1063/5.0200384, URL https://doi.org/10.1063/5.0200384.

[29] Sitzmann, V., Martel, J. N. P., Bergman, A. W., Lindell, D. B., and Wetzstein, G., "Implicit Neural Representations with Periodic Activation Functions," , 2020. URL https://arxiv.org/abs/2006.09661.

[30] Scotzniovsky, L., and Hwang, J. T., "A fast, memory-efficient panel method for large-scale multidisciplinary design optimization under uncertainty using graph-based modeling," *AIAA Aviation 2025 forum*, 2025.

[31] Bock, K., "Towards a 3D Galerkin-Type High-Order Panel Method: A 2D Prototype," *New Results in Numerical and Experimental Fluid Mechanics XIII*, edited by A. Dillmann, G. Heller, E. Krämer, and C. Wagner, Springer International Publishing, Cham, 2021, pp. 581–591.