Advancing Airfoil Design: A Physics-Inspired Neural Network Model

Conference Paper · August 2024

DOI: 10.1115/GT2024-122682

7 authors, including:

Khalil Al Handawi
Université de Montréal
17 PUBLICATIONS **118** CITATIONS

SEE PROFILE

**Proceedings of Proceedings of ASME Turbo Expo 2024 &
Turbomachinery Technical Conference and Exposition
GT2024
June 24-28, 2024, London, United Kingdom**

GT2024-122682

ADVANCING AIRFOIL DESIGN: A PHYSICS-INSPIRED NEURAL NETWORK MODEL

Mathieu Salz
Mechanical Engineering
McGill University
Montréal, Canada
mathieu.salz@mail.mcgill.ca

Can Unlusoy
Mechanical Integrity Specialist
Siemens Energy Canada
Montréal, Canada
can.unlusoy@siemens-energy.com

Khalil Al Handawi*
Optimization Specialist
Siemens Energy Canada
Montréal, Canada
khalil.al-handawi@siemens-energy.com

Tittu Varghese Mathew
Probabilistic Modelling Specialist
Siemens Energy Canada
Oakville, Canada
tittu.varghese-mathew@siemens-energy.com

Bill Maier
Siemens Energy USA
Olean, United States
wmaier@siemens-energy.com

Ravichandra Srinivasan
Siemens Energy USA
Redmond, United States
ravichandra.srinivasan@siemens-energy.com

Michael Kokkolaras
Mechanical Engineering
McGill University
Montréal, Canada
michael.kokkolaras@mcgill.ca

ABSTRACT

Turbomachines are an integral part of the energy and industrial landscapes, and improvements to their efficiency benefit the environment, profitability of operation, and in turn, society at large. Therefore, the application of advanced methods for rapid design and development of high-performance turbomachinery components is of significant interest. In the past decade, the use of optimization methods has made inroads in improving turbomachinery aerodynamics. Recent advances in machine learning (ML) methods have the potential to augment design systems by providing the ability to explore larger design spaces and generate high-quality initial designs. Physics Informed Neural Networks (PINNs), based on the Navier-Stokes equations, are used to incorporate physical laws into the design process. This approach leverages the power of deep learning while ensuring that the designs conform to fundamental principles of fluid dynamics. The use of Physics Informed Neural Networks (PINNs) not only accelerates the design process by reducing the need for extensive simulations but also improves the accuracy of the designs by en-

*Address all correspondence to this author.

suring physical consistency as opposed to designs made using Generative Artificial Intelligence (AI) models. However, combining PINNs with Generative AI for airfoil optimization could provide a fruitful avenue in improving compressor blade designs.

Keywords: Physics informed neural networks (PINNs), Computational fluid dynamics (CFD), Blading design

1 Introduction

Blades are among the devices employed in turbomachines to manipulate the aerodynamic properties of working fluids, and are commonly defined by airfoil cross-sections. Airfoils in blades are typically designed to induce predetermined amounts of flow turning informed by target distributions of flow and loading parameters across the fluid passage. While the target flow angles and flow incidence models allow constraining the initial and final slopes of the airfoil camber lines, designers still have freedom in defining the blade-to-blade passage geometry, by manipulating the suction side (SS) and pressure side (PS) contours of the airfoil, as well as the spacing of airfoils. The free design variables may be optimized for a variety of objectives informed by design priorities, such as maximum efficiency, robustness to incidence, minimum material use; or to match prescribed profiles of aerodynamic quantities across the airfoils. Although design rules, criteria and conventions that exist within industrial organizations and in the literature may bound the design space, a true blank-sheet design of an airfoil necessitates the exploration of a large design space. In traditional iterative workflows, capturing fine levels of variation between designs across ranges of flow inlet and outlet angles requires large numbers of samples, which are then evaluated using flow solvers. Depending on the desired fidelity of analysis, evaluation of designs may be resource-intensive. In industrial settings with frequent design work on custom products, conducting a new search for each design prompt adds further to the resource use, and generates a large amount of data that is not utilized more than once with traditional iterative methods such as optimization.

Generative AI models can be employed to explore a vast design space efficiently. These models learn from existing designs and generate new design configurations based on desired performance characteristics. The generative models can explore complex design spaces, with the potential to improve performance over existing designs. However, as these models are fully data-driven, they can fail to fully capture the physics of the problem and thus produce sub-optimal designs. First, this paper seeks to reduce the computational expense associated with commercial Computational Fluid Dynamics (CFD) solvers by training a parametric PINN to be used as a surrogate model, in order to rapidly characterize the performance of airfoils. Secondly, we will propose a methodology for combining the use of Generative AI models for the generation of feasible candidate airfoil design with the use of PINNs as low-cost flow solvers to find optimal designs as means of efficiently exploring the large design space.

The main contribution of this paper is a flow characterization tool based on parametric PINNs for exploring compressor blading design spaces.

Section 2 discusses literature relevant to airfoil generation and machine learning-based physics modeling. This is followed by details on the method development in Section 3. The computational results are presented in Section 4. The paper ends with a discussion and conclusion in Sections 5 and 6, respectively.

2 Background

2.1 Machine Learning and CFD

The analysis and downstream engineering tasks of generated airfoil designs can be prohibitively expensive due to their reliance on simulation models. In the paper, we focus on the aerodynamic characterization of airfoils through the use of CFD.

Feedforward Neural Networks are theoretically proven to be universal function approximators [1], which has motivated their use in learning the function mapping of various Partial Differential Equations (PDEs) involved in solving the fluid flow problem. However, these networks need large quantities of data to effectively capture the PDE, essentially performing curve-fitting, and do not generalize to changes in the PDE parameters nor can extrapolate in time and space. In order to learn PDEs with little to no data, PINNs were introduced, which take in as input space and time coordinates and output quantities of interest such as velocity and pressure, by augmenting a purely data loss training paradigm with PDE-residual loss terms, as well as boundary and initial conditions as the selected problem demands [2]. These PDE residual terms are calculated using the automatic-differentiation capabilities of neural networks [3], producing nth order and mixed partial derivatives of the inputs with respect to the outputs of the network. Through the Navier-Stokes equations, PINNs have seen uses in fluid mechanics [4] for incompressible, compressible, and turbulent flows [5]. This work has been extended to the field of turbomachinery, with PINNs used to predict flow around low-pressure turbine blades and in periodic hills [6]. PINNs's capacity for automatic differentiation has also lead to their use in design optimization of airfoils [7], where airfoils parametrized using PARSEC had their parameters optimized to maximize their lift to drag ratio. We implement PINNs using the NVIDIA Modulus framework [8] which uses a PyTorch [9] backend.

2.2 Relevance of PINNs to Generative AI

Airfoil generative design relies on training a model using a database of feasible airfoil candidates. Such databases are accumulated through past engineering design iterations and experience and can be used to inform future designs. Generative models can capture input-output relations between random variables

and reuse data to produce samples with similar characteristics to those in their training datasets.

Adapting them to engineering design applications, generative models may be trained to generate design representations conditional on performance requirements, and retrained regularly as design datasets grow, effectively embodying and reusing existing knowledge to accelerate design processes. Large volumes of data generated with traditional design methods can thus be put to use in rapidly generating candidates for design prompts, including those with performance requirements not included in training datasets [10].

Generative processes may be complemented by analysis tools incorporating rule-based knowledge, in the form of equations governing the performance of designs [11]. Such knowledge could be informed by PINNs in the case of PDE-governed models such as those presented in this paper.

3 Methodology

3.1 PINN Problem Set-Up

We want to solve the 2D incompressible viscous laminar Navier-Stokes equations, with the PDE residuals for each equation defined as in Figure 2. We want to solve these equations over the space defined by $(-1,2)\times(-1,1)$, with a parametrized airfoil in the center of the domain as shown in Figure 3, walls above and below the airfoil to constrain the flow going from the inlet towards the outlet. An example airfoil profile with a turning angle of -6 degrees is shown in Figure 1. For each airfoil, camber lines were parameterized as cubic splines, with initial and final slopes given by metal inlet/outlet angles, first control point at the trailing edge being fixed, and second control point at the trailing edge having a free y-coordinate. For each inlet/outlet angle combination, y-coordinate of the TE control point was selected as the mean value in the range of values that produce an inflection point-free camber line. For the thickness distribution about the camber line, we used four control points: two placed at the leading and trailing edges with thickness equal to the leading and trailing edge radii of 0.005, and two at x coordinates of 0.1 and 0.9 with a thickness value of 0.01. This problem set-up is akin to an airfoil in the test section of a wind tunnel. For the purposes of this study, we consider airfoils that are parametrized only by metal turning angle θ, with fixed axial chords of 1 unit length and fixed thickness distributions, metal inlet angle and flow inlet angles at 0°, and camber line endpoint y-coordinates as a function of the turning angle, $y_{\text{camber,end}}(\theta)$.

Each turning angle yields a different airfoil, meaning different boundary conditions for the problem, and thus a different unique solution. We can thus consider the problem domain to be a 3D space with dimensions x, y, and θ (the turning angle) defined by $(-1,2)\times(-1,1)\times(-1,-12)$. Hence, the PINN is solving the Navier-Stokes equations parametrically over the same spatial domain but with different values of turning angles (i.e. different airfoils).

The boundary conditions employed are the same as those used in [7]:

1. a parabolic inlet condition given by $u = 1.2(y+1)(y-1)$ and $v = 0$,
2. an outlet condition of $p = 0$,
3. and a no-slip condition on the airfoil and top and bottom walls $u = v = 0$.

Each loss term is calculated over a batch of points (x, y, θ), with its final value taken to be the mean squared error (MSE) over this batch. The final total loss is a weighted aggregation of each loss term.

$$\mathcal{L}_{\text{total}} = \lambda_1\mathcal{L}_{\text{PDE}} + \lambda_2\mathcal{L}_{\text{inlet}} + \lambda_3\mathcal{L}_{\text{outlet}} + \lambda_4\mathcal{L}_{\text{airfoil}} + \lambda_5\mathcal{L}_{\text{wall}} \quad (1)$$

3.2 PINN Architecture

The PINN architecture is shown in Figure 2. The point coordinates x and y are used as inputs, along with the turning angle θ. The input is then fed into a Fourier Projection Layer, with frequencies fixed at initialization, followed by n fully connected hidden layers of width m. After each hidden neuron, an activation function (AF) of $\tanh(x)$ is applied in order to provide nonlinearity to the network. We enhance the AFs by using adaptive AFs which should increase the rate of convergence of the network [12]. Finally, the PINN outputs the flow variables u, v, and p. Using automatic differentiation, the derivatives of the flow variables with respect to x and y inputs can be calculated, allowing the construction of the PDE residuals for continuity, x-momentum, and y-momentum.

3.3 PINN Loss Optimization

In order to solve the Navier-Stokes equations, PINNs use a weighted loss function in Equation (1) to optimize for different objectives. The loss terms are those which characterize the boundary conditions, the PDE residuals, and if included, the data loss. The different loss sections defined over the (x,y) domain are, shown in Figure 3, the inlet, the outlet, the top and bottom walls, the interior, and the airfoil surface. The PDE residuals are optimized only in the interior domain. For both training and validation, CFD simulations were conducted using Ansys Fluent for three turning angles ($-1, -6.26, -12$ degrees). The CFD results are used as data loss terms used in training as the MSE between the PINN predictions and CFD results for u, v, p and their derivatives with respect to x and y in the interior domain and on the airfoil surface.

At each iteration, we choose a batch of points from each loss section randomly, calculate the squared error for each i^{th} point,

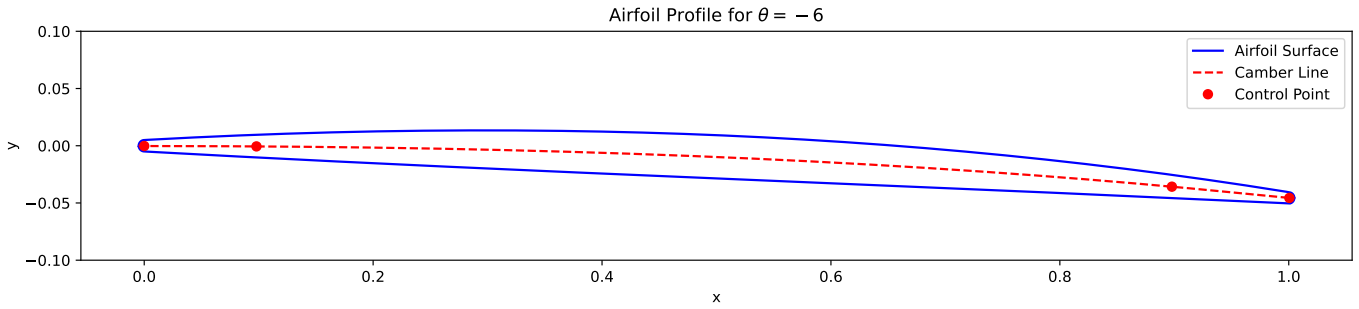


FIG. 1: Airfoil Profile for $\theta = -6$

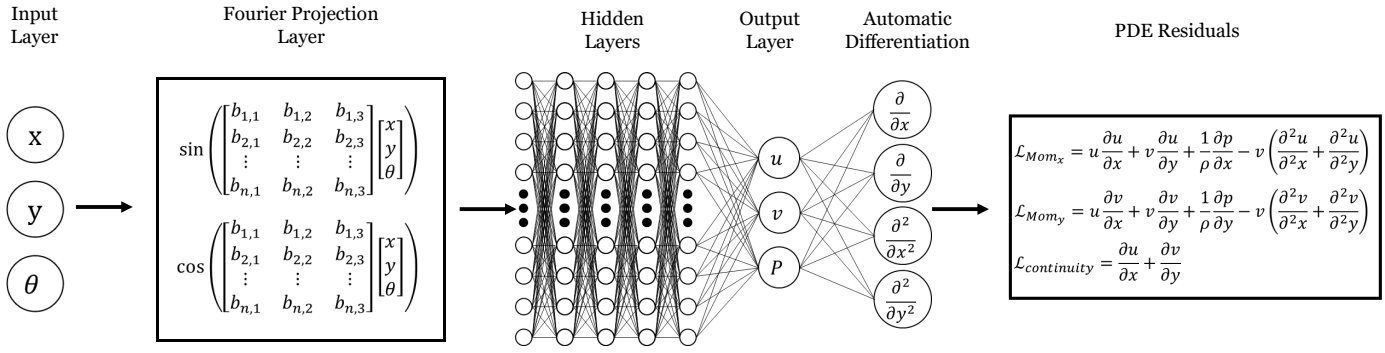


FIG. 2: PINN Architecture

and then take the mean over those points:

$$\mathcal{L}_{z,\text{domain}} = \frac{1}{N_{\text{batch}}} \sum_{i=0}^{N_{batch}} \lambda(x,y) \left(z_{i,\text{predicted}} - z_{i,\text{actual}}\right)^2, \qquad (2)$$

where $z = z(x,y,\theta)$ is an arbitrary variable (e.g., u, v, and p) over an arbitrary domain (e.g., inlet, outlet, interior, wall, and airfoil domains). The weight function $\lambda(x,y) : \mathbb{R}^2 \mapsto \mathbb{R}$ applies a spatial weighting on the predictions. For the PDE residuals, we use the positive Signed Distance Function (SDF) with respect to the airfoil to weigh the error as suggested by the NVIDIA Modulus documentation [8]. Due to the dimensions of the domain, $\lambda(x,y)$ is constrained between 0 and 1.4. This weighting should help training by reducing the magnitude of the error near the airfoil, where the gradients, and thus the PDE residuals, will be high. In addition, we use $\lambda = 10$ for $\mathcal{L}_{u,\text{inlet}}$, as the flow will propagate from the inlet towards the outlet through the optimization of the Navier-Stokes equations. We define the loss terms for each loss section as follows:

$$\mathcal{L}_{\text{inlet}} = \mathcal{L}_{u,\text{inlet}} + \mathcal{L}_{v,\text{inlet}} \qquad (3)$$

$$\mathcal{L}_{\text{outlet}} = \mathcal{L}_{p,\text{outlet}} \qquad (4)$$

$$\mathcal{L}_{\text{walls}} = \mathcal{L}_{u,\text{wall}} + \mathcal{L}_{v,\text{wall}} \qquad (5)$$

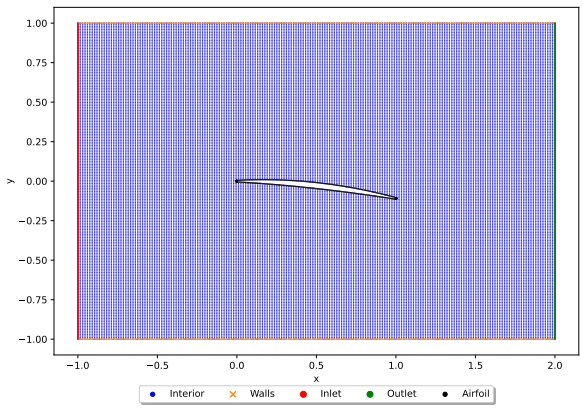
$$\mathcal{L}_{\text{airfoil}} = \mathcal{L}_{u,\text{airfoil}} + \mathcal{L}_{v,\text{airfoil}} \qquad (6)$$

$$\mathcal{L}_{\text{PDE}} = \mathcal{L}_{x-\text{momentum,in}} + \mathcal{L}_{y-\text{momentum,in}} + \mathcal{L}_{\text{continuity,in}} \qquad (7)$$

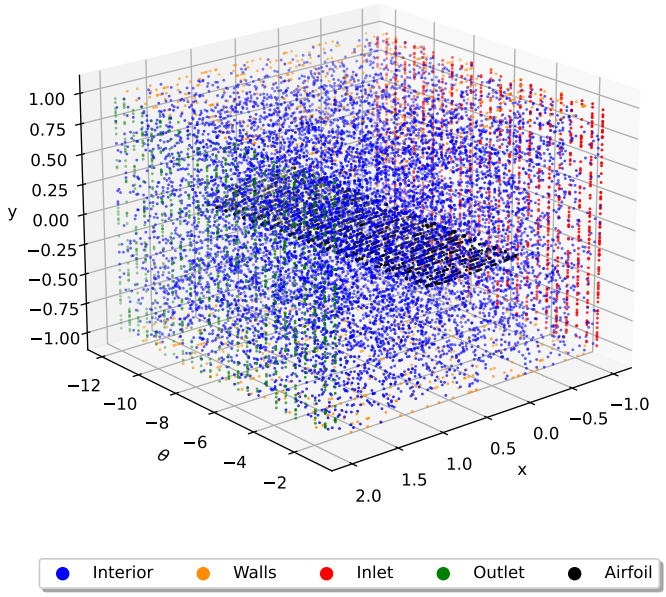
where the domain specific losses $\mathcal{L}_{u,\text{inlet}}$ through $\mathcal{L}_{\text{continuity,in}}$ are given by Equation (2). The PDE loss in Equation (7) is defined in the interior domain. The final total loss is the aggregated sum of all the loss terms.

$$\mathcal{L}_{\text{total}} = \sum_{i=1}^{5} \mathcal{L}_{\text{domain},i}, \qquad (8)$$

where $\mathcal{L}_{\text{domain},i}$ is given by Equations (3) through (7). However, the different loss terms may have different orders of magnitude, meaning that in an aggregated sum scheme some loss terms will be preferentially optimized over others. In order to remedy this, we use the loss scaling approach Relative Loss Balancing with



(a) Full Sample Space of Loss Sections for $\theta = -12$



(b) Batch Sample of Loss Sections over Angle Range

FIG. 3: Loss Sections

Random Lookback (ReLoBRaLo) to balance the loss term [13]. Finally, the total loss is backpropagated using the Adam optimizer in order to update the weights of the network [14]. Over the course of training, we use an exponential decay rate of 0.95 to slowly reduce the learning rate (LR), so that weight updates become smaller.

3.4 Evaluation Criterion

A trained PINN model can be evaluated using several criteria. First of all, the final aggregated loss can be used to compare models to each other. However, a lower aggregated loss might not necessarily mean a more physically accurate flow. In addition, using the CFD results, we calculate the MSE over all the CFD points for the flow variables predicted by the PINN to provide the physical accuracy over the full flow domain. Finally, for these same turning angles, we calculate the lift and drag over the airfoils characterized and compare them to the lift and drag calculated using the CFD results.

We calculate the shear stresses in the following manner:

$$\tau_{xx} = 2\mu\frac{\partial u}{\partial x} - \frac{2}{3}\mu\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) \tag{9}$$

$$\tau_{xy} = \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \tag{10}$$

$$\tau_{yy} = 2\mu\frac{\partial v}{\partial y} - \frac{2}{3}\mu\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) \tag{11}$$

$$\tau_{wall,x} = n_x\tau_{xx} + n_y\tau_{xy} \tag{12}$$

$$\tau_{wall,y} = n_x\tau_{xy} + n_y\tau_{yy}, \tag{13}$$

where n_x and n_y are the outward normals to the surface. We then calculate the lift and drag:

$$F_{\text{axial}} = \iint_A -pn_x + \tau_{\text{wall},x}\,dA \tag{14}$$

$$F_{\text{vertical}} = \iint_A -pn_y + \tau_{\text{wall},y}\,dA \tag{15}$$

$$\text{Lift} = F_{\text{vertical}}\cos(\alpha) - F_{\text{axial}}\sin(\alpha) \tag{16}$$

$$\text{Drag} = F_{\text{vertical}}\sin(\alpha) + F_{\text{axial}}\cos(\alpha), \tag{17}$$

where α is the angle of attack of the airfoil, as defined by the angle of its chord with respect to the x-axis. Using these four metrics, loss, MSE over flow, lift, and drag, we can compare models with different hyperparameters. These hyperparameters affect the architecture of the PINN (frequencies used in Fourier Projection Layer, number of hidden layers, width of hidden layers), as well as the training regime (intial LR, loss weights, loss balancing scheme).

4 Results

In order to find the PINN hyperparameters for best accuracy, we try different sets of hyperparameters to modify the model and its training regime. For the PINN architecture, we use a Multi-Layer-Perceptron (MLP), a Fourier Network, and a Modified Fourier Network [8]. In addition, we vary the amount of hidden layers between 4 and 5, as well as varying the width of these hidden layers between 50 and 100 neurons. Finally, we judge the effect of adding adaptive AFs to the convergence rate and accuracy of the model. The training regime is modified by varying the initial LR (5×10^{-2}, 1×10^{-2}), the point cloud den-

TABLE 1: Batch Sizes

Inlet	Outlet	Interior	Airfoil	Walls
1500	1000	10000	1500	1000

sity, the extent to which CFD data is used or not, and whether SDF weighting is used or not.

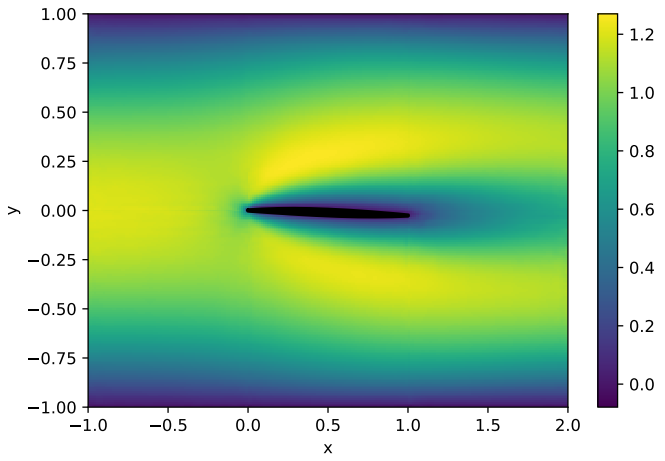
We distinguish between a low, a very low, and a high point cloud density, where the first is defined by the batch sizes as seen in Table 1, the second by batch sizes twice as small, and the last batch-sizes twice as large. We also distinguish between full, half, and none for CFD use, where full means that CFD data is used both in the interior domain and on the airfoil surface, half when used only in interior domain, and none when no data is used at all.

We will consider our base model, for the purposes of comparison, to be a Fourier Network with 5 hidden layers and 50 hidden neurons, which uses adaptive AFs, SDF weighting for the interior residual points, a low point cloud density, an initial LR of 1×10^{-2}, full validation, and the ReLoBRaLo loss balancing scheme.

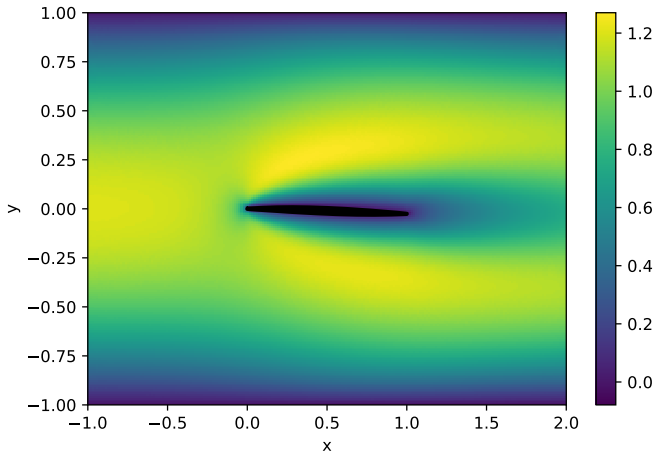
In Table 2, we show the performance of the baseline model in terms of the loss and the percentage relative error for lift and drag. We observe that the percentage relative error with respect to the CFD of the PINN is lower for drag than it is for lift. We will only compare models based on their final losses, and the mean relative error over lift and drag. These quantities will be relative to the baseline, with negative quantities indicating lower error, and positive quantities higher error.

4.1 Effect of Training Hyperparameters

In Table 3, we compare the baseline model to models with the same architecture, but with differences to the training hyperparameters in terms of loss balancing, loss weighting, LR, and use of CFD results during training. We find that the exclusion of CFD data decreases the performance of the PINN with respect to the relative error of lift and drag. The error in lift decreases the most from the inclusion of the CFD results in the interior region, while drag improves the most from the inclusion of the CFD results on the surface of the airfoil. It is interesting to note however that the no-validation model, despite having lower loss (-4.236), has much higher error for drag and lift (91.602 and 30.034, respectively) compared to the baseline (3.526 and 0.955, respectively). This decrease is due to the exclusion of the data loss terms in the total loss. The model with no SDF weighting has much higher loss, but has lower error than the baseline. This increase in total loss may be due to the fact that the SDF weighting decreases the loss of the interior points near the airfoil, where the loss will be highest due to the large gradients encountered there. However, without SDF weighting, the optimizer focuses



(a) PINN Results for u



(b) CFD Results for u

FIG. 4: Results Comparison for $\theta = -1$

on points near the airfoil where the loss is high, which explains the decrease in error in lift and drag. The exclusion of adaptive AFs increases performance for all metrics. Sum-loss aggregation decreases loss and error in drag but increases error in lift significantly. Finally, increasing the LR increases the loss significantly, slightly increases the error in drag, and significantly increases the error in lift.

4.2 Effect of Architecture

In Table 4, we compare the effect of changing architecture of the base model on the final performance of the PINN. We see that the Modified Fourier Network performs the best in terms of lowest loss and lowest relative error on lift. The MLP has the lowest error for drag, but has the highest loss. In addition, increasing the amount of frequencies in the Fourier network improves the performance of the PINN across all metrics, though to a lesser

TABLE 2: Baseline Model Performance (% relative error)

Architecture	Final Loss	Lift	Drag
Fourier	4.648	3.526	0.955

TABLE 3: Training Hyperparameters Relative to Baseline Model

Change	Final Loss	Lift	Drag
No SDF	+20.892	-1.905	-0.035
No Adaptive AFs	-0.546	-0.378	-0.257
Higher LR	+3.186	-0.774	+0.087
Sum Loss Agg.	-1.426	+0.819	-0.577
Half Val.	-4.107	-1.036	+26.948
No Val.	-4.236	+91.602	+30.034

TABLE 4: Effect of Architecture on Performance Relative to Baseline Model

Architecture	Final Loss	Lift	Drag
Fourier with extra frequencies	-0.342	-0.96	-0.242
Modified Fourier	-3.713	-1.905	-0.342
MLP	+0.849	+2.344	-0.399

TABLE 5: Effect of Depth & Width on Performance Relative to 5L50 MF

Architecture	Depth	Width (Neurons)	Loss	Lift	Drag
Fourier	4	50	+3.768	+0.846	+0.105
Fourier	4	100	+0.49	-0.55	-0.254
Mod. Fourier	4	100	-0.122	-0.731	-0.355
Mod. Fourier	5	100	-0.463	-0.696	-0.321

degree than the modified Fourier network. We will now employ the modified Fourier model with 5 layers and 50 neurons, which we will refer to as 5L50 MF, as a new baseline with which to compare other models since it is the best performing so far.

TABLE 6: Effect of Point Cloud Density on Performance Relative to 4L100 MF

Architecture	Density	Final Loss	Lift	Drag
Mod. Fourier	High	0.965	-0.121	+0.123
Mod. Fourier	Very Low	-0.218	+0.240	+0.180

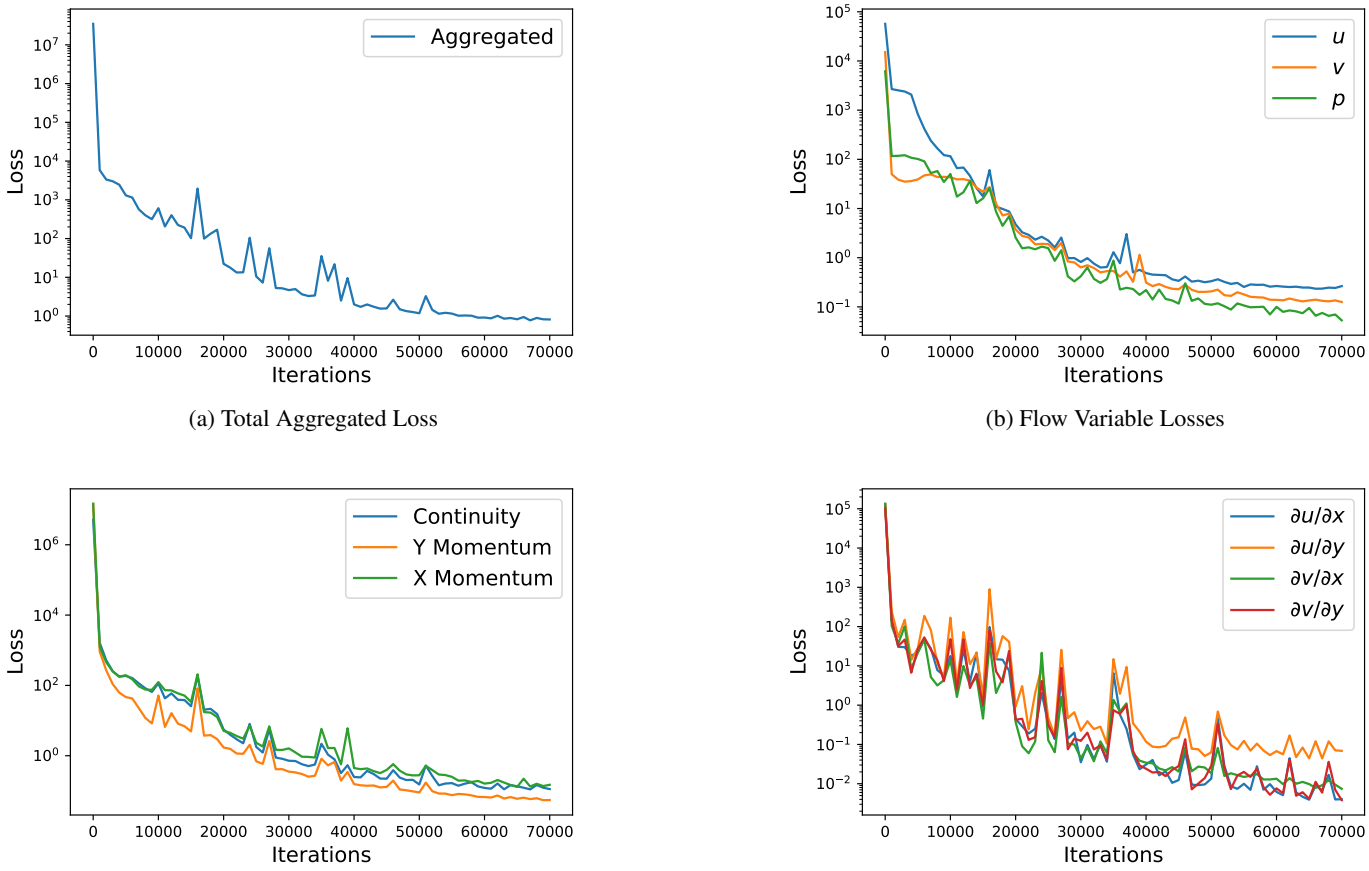
4.3 Effect of Size and Depth of Network

Additional layers and neurons increase the capacity of a model, which in turn may decrease error. In Table 5, we compare models with different widths and depths to the baseline. We find that the only model which has lower capacity compared to the baseline, the Fourier Network with 4 layers and 50 neurons, has no improvements with respect to the baseline. Out of the three models which improve (lower loss, relative error on drag and lift) upon the Modified Fourier Network with 5 layers and 50 neurons, the Modified Fourier network with 4 layers and 100 neurons shows the most improvement in terms of lift and drag, even though the Modified Fourier Network with 5 layers and 100 neurons has more capacity, and a lower loss. For the next set of hyperparameters, we will use the Modified Fourier Network with 4 layers and 100 neurons as a baseline, which we will refere to as 4L100 MF.

4.4 Effect of Point Cloud Density

Increasing the point cloud density can decrease the error by providing more accurate gradients when updating the weights and better capturing important flow regions, however, this can cause networks to fall into local minima. Decreasing the point cloud density can also decrease error, by providing gradients which are slightly noisy, which can avoid local minima to reach global objective function minima. In addition, increasing the point cloud density leads to additional computational expense and memory usage, while the reverse is true for decreasing the point cloud density. In Table 6, we see that decreasing the point cloud density, while decreasing the final loss, also decrease the performance compared to the baseline with respect to lift and drag. When increasing the point cloud density, the final loss is increased, with a boost in performance with regards to lift accompanied with a reduction in performance with regards to drag.

In Figure 5, we demonstrate different loss traces over the course of $70,000$ iterations. We see that the total aggregated loss, the flow variable losses, and the PDE residual losses all diminish smoothly, while the gradient losses are subject to dras-



(a) Total Aggregated Loss

(b) Flow Variable Losses

(c) PDE Residual Losses

(d) Gradient Losses

FIG. 5: Losses vs Iterations for Best Performing Model

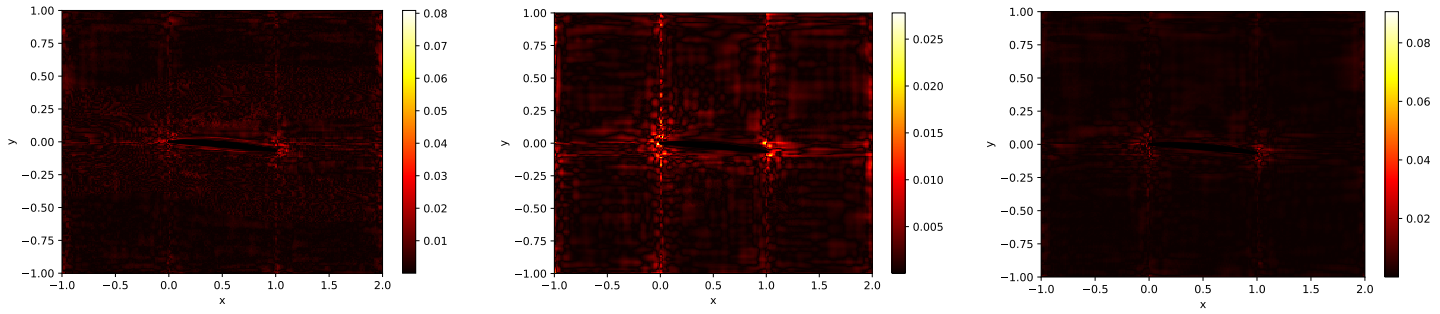
tic increases and decreases. As well, we observe that the loss for $\partial u/\partial y$ is significantly higher than the other gradients. These peaks could be due to the fact that the gradient losses, even though they are indirectly optimized through the PDE residual losses, are only directly optimized through the CFD data loss over the airfoil surface. The updates of the PINN parameters may be too large over this small and sensitive region, causing overcorrections in response to the loss.

5 Discussion

5.1 PINNs as Surrogate Models

We find the best performing model to be the modified Fourier network, with 4 layers and 100 neurons, ReLoBRaLo loss balancing, adaptive AFs, low point cloud density, an initial LR of 1×10^{-2}, and with SDF weighting. Figures 4 and 6 demonstrate visually the performance of the PINN in comparison to CFD for a turning angle of $\theta = -1^{o}$ for the flow variable u, as well as the absolute error for all the flow variables for a turning angle of $\theta = -6.26^{o}$. We see in Figure 6 that the highest absolute error between the CFD and PINN occurs at the leading and trailing edges of the airfoil for u, v, and p. For the prediction of u, there is also an area of relatively significant error at the boundary layer of the airfoil. During the training process, three CFD simulations were used for data, with each taking 15.945 seconds of wall-clock time to run on Ansys Fluent, whereas the PINN took 5 hours, 33 minutes, and 31 seconds to train on a Tesla K80 GPU.

From these results, we find that PINNs can be used to solve the Navier-Stokes equations for a parametrized airfoil. Relative error for lift and drag could be decreased through further hyperparameter tuning. In addition, instead of sampling uniformly over the x and y dimensions, points could be clustered near the airfoil as that is where the highest relative losses for the flow variables are found in the interior domain, which in turn affect the error for lift and drag. This point distribution could be done by using a standard mesh such as OH-block structured to improve resolution and better capture sensitive areas. In Section 4.1, we



(a) u (b) v (c) p

FIG. 6: PINN Absolute Error of flow variables for $\theta = -6.26$

do not investigate how the effect of permutations of different hyperparameters, which could shed light on the interplay between hyperparameters, nor do we run the trainings for each hyperparameter multiple times and average the results to account for the randomness imbued in the PINN training process. In addition, due to limited computational resources, we do not conduct a hyperparameter test for each new model architecture and size. Including more CFD results for other turning angles could also improve performance. As seen in the results section, the different PINN models need to be compared using not just the final loss, but as well the relative error and the lift, as a lower final loss may not necessarily indicate a better performing model overall. Including the error for lift and drag into the total loss function which is minimized could further improve results.

Several changes are needed in order to increase the generalizability of the PINN model outside of this limited set-up. First, additional airfoil parameters could be included in order to increase the amount of airfoils the PINN can be used on, which would in turn increase the dimensionality of the problem. Furthermore, the inlet speed should be increased to much higher values and the viscosity should be lowered to more closely match real conditions inside compressors. However, these modifications will require the use of the compressible Navier-Stokes equations, as well as the inclusion of an appropriate turbulence model. The former will require the PINN to not only predict the current flow variables used, but as well density or temperature, which in turn requires the addition of the Energy equation for the problem to be well posed.

5.2 Optimization using PINNs

PINNs are not only useful as surrogate models, but can also be used in different ways for the optimization of airfoil designs. Automatic differentation is already used for two very important gradient calculations in PINNs: the gradients of the outputs with respect to the inputs of the network in order to reconstitute the Navier-Stokes equations, and the gradients of the loss with respect to the parameters of the network in order to update the parameters during training. In addition, in Sun et al., 2023, automatic differentiation is used by somewhat mixing these two ideas: define a design objective function f (e.g lift-to-drag ratio), constraints $\mathbf{g}$ (e.g., loss bucket and surge characteristics) and then calculate the gradient of the objective and constraints with respect to the design variables ($\nabla f(\mathbf{x})$ and $\nabla \mathbf{g}(\mathbf{x})$) [7]. Now that these gradients are available, they can be used in any gradient-based optimization method in order to minimize or maximize the design objective function by modifying the design parameters. This optimization process starting from a single random (or perhaps standard) design would be reflective of the Point-Based Design (PBD) paradigm [15].

In order to produce an optimal design from within a vast design space and not just a local minima, the design space must be properly explored. Thus, several initial guesses methodically dispersed throughout the domain (e.g Latin Hypercube Sampling) need to be used as the starting points for several optimization runs. However, this approach could be inefficient due to a large amount of initial points needed to cover the design space, as well as the difficulty in estimating the objective and constraint functions and their gradients with respect to the design variables. These difficulties could be mitigated by using a Generative AI model to propose initial guesses.

As proposed by Unlusoy, 2023 [16], an airfoil design tool for axial blades that outputs satisfactory designs can be created by formulating a simplified and generalized version of the design problem, where, for the airfoil section of each radial span of the blade, the design parameters that minimize frictional losses and meet the flow inlet and exit angles are to be found. Considering design parameters $\mathbf{x}$ and performance metrics of total pressure loss, and flow inlet and exit angles $\mathbf{y}$ as realizations of random variables, we utilize conditional deep generative models, such as conditional Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs) and Real-Valued Non-Volume Preserving Transformations (RealNVPs), with high model complexity to capture the conditional distribution of $\mathbf{x}$ with respect to $\mathbf{y}$ using training data of $(\mathbf{x}, \mathbf{y})$ samples [17, 18, 19]. The modeled

distribution $p(\mathbf{x}|\mathbf{y})$ is then sampled from, i.e. "queried", with the target $\mathbf{y}$ of the design prompt to obtain many design options that satisfy the initial design requirements (such as optimization constraints).

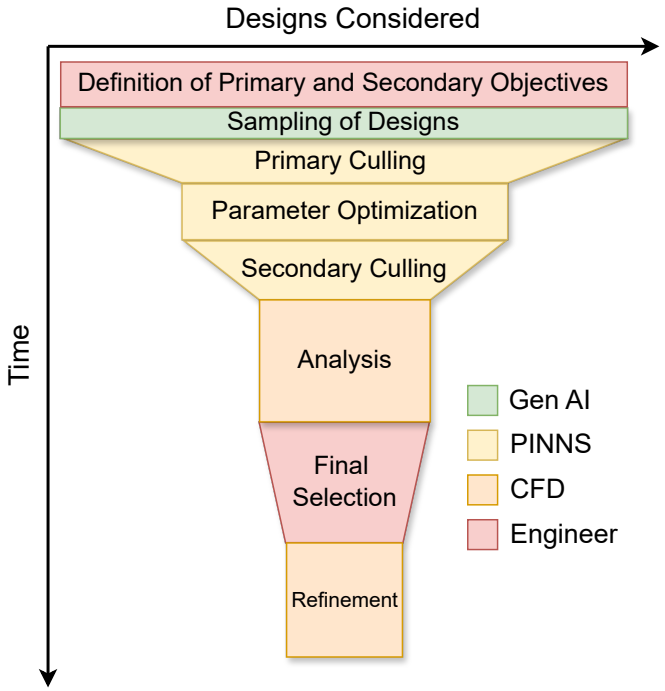


FIG. 7: Set-Based Concurrent Engineering (SBCE) Process Diagram

By querying the Generative AI model with target variables, we recover not just a single satisfactory sample, already likely near to an optimal design, but a variety of satisfactory designs which may each present their own advantages in terms of secondary design considerations. Thus, once a batch of airfoil designs has been generated, the PINN in its ability as a surrogate model can be used to either weed out unsatisfactory designs, shortlist a certain amount of the best designs with regards to the primary objective function, or select a subset wherein each design satisfies the primary objective as well as a unique secondary objective. Furthermore, using automatic differentiation and a gradient-based method, the remaining designs can be optimized as desired using the PINN. Finally, commercial CFD solvers could be used for a more accurate and in-depth analysis of the optimized designs in order to find the best one, which should not need much more modification. Hence, using a Generative AI model to produce a set of initial samples for a PINNs to cull and optimize could present benefits in terms of improved performance, and a facilitated method for pivoting between designs as secondary requirements are established or change. This workflow is best suited for a Set-Based Concurrent Engineering (SBCE) approach [20] which explores and iterates over several designs as opposed to iterating over a single best-guess design in PBD [15]. A Set-Based Concurrent Engineering (SBCE) process, as illustrated in Figure 7 would scale up the cost of its computational methods as it reduces the amount of candidate designs, unintuitively allowing more time to be spent on considering various designs while decreasing the total design time.

6 Conclusion

We have succesfuly demonstrated the application of PINNs to an airfoil parametrized by the turning angle. In addition, we lay out how, with further improvements to the PINN and complete parameterization of the airfoil, PINNs combined with Generative AI models can be used for the design and optimization of compressor blades using an SBCE approach.

Acknowledgements

The authors of this paper acknowledge financial support from Siemens Energy Canada Limited and the Mathematics of Information Technology and Complex Systems (MITACS) Ref. number IT36316.

References

[1] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(89)90020-8. URL https://www.sciencedirect.com/science/article/pii/0893608089900208.

[2] Maziar Raissi, Paris G. Perdikaris, and George 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. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2018.10.045. URL https://www.sciencedirect.com/science/article/pii/S0021999118307125.

[3] Atilim Gunes Baydin, Barak A. Pearlmutter, and Alexey Andreyevich Radul. Automatic differentiation in machine learning: a survey. *CoRR*, abs/1502.05767, 2015. URL http://arxiv.org/abs/1502.05767.

[4] Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, and George Em Karniadakis. Physics-informed neural networks (pinns) for fluid mechanics: a review. *Acta Mechanica Sinica*, 37(12):1727–1738, Dec 2021. ISSN 1614-3116. doi: 10.1007/s10409-021-01148-1. URL https://doi.org/10.1007/s10409-021-01148-1.

[5] Shinjan Ghosh, Amit Chakraborty, Georgia Olympia Brikis, and Biswadip Dey. Ransplnn based simulation surrogates for predicting turbulent flows. *arXiv preprint*, 2023.

[6] Sean K. Hanrahan, Melissa Kozul, and Richard D. Sandberg. Predicting transitional and turbulent flow around a turbine blade with a physics-informed neural network. In *Turbo Expo: Power for Land, Sea, and Air*, volume 13C: Turbomachinery — Deposition, Erosion, Fouling, and Icing; Design Methods and CFD Modeling for Turbomachinery; Ducts, Noise, and Component Interactions, page V13CT32A010, 06 2023. doi: 10.1115/GT2023-101238. URL https://doi.org/10.1115/GT2023-101238.

[7] Yubiao Sun, Ushnish Sengupta, and Matthew Juniper. Physics-informed deep learning for simultaneous surrogate modeling and pde-constrained optimization of an airfoil geometry. *Computer Methods in Applied Mechanics and Engineering*, 411:116042, 2023. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2023.116042. URL https://www.sciencedirect.com/science/article/pii/S0045782523001664.

[8] Modulus - a neural network framework. URL https://developer.nvidia.com/modulus.

[9] Pytorch documentation. URL https://pytorch.org/docs/stable/index.html.

[10] Lyle Regenwetter, Amin Heyrani Nobari, and Faez Ahmed. Deep Generative Models in Engineering Design: A Review. 144(7), jul 2022. ISSN 10500472. doi: 10.1115/1.4053859. URL https://dx.doi.org/10.1115/1.4053859.

[11] Marcus Sandberg, Ilya Tyapin, Michael Kokkolaras, Ola Isakasson, Jan-Olov Aidanpää, and Tobias Larsson. A Knowledge-based master-model approach with application to rotating machinery design. *Concurrent Engineering Research and Applications*, 19(4):295–305, dec 2011. ISSN 1063293X. doi: 10.1177/1063293X11424511/ASSET/IMAGES/LARGE/10.1177_1063293X11424511-FIG7.JPEG. URL https://journals.sagepub.com/doi/10.1177/1063293X11424511.

[12] Ameya D. Jagtap, Kenji Kawaguchi, and George Em Karniadakis. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *Journal of Computational Physics*, 404:109136, March 2020. ISSN 0021-9991. doi: 10.1016/j.jcp.2019.109136. URL http://dx.doi.org/10.1016/j.jcp.2019.109136.

[13] Rafael Bischof and Michael Kraus. Multi-objective loss balancing for physics-informed deep learning. *arXiv preprint*, 2021. doi: 10.13140/RG.2.2.20057.24169. URL http://rgdoi.net/10.13140/RG.2.2.20057.24169.

[14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint*, 2017.

[15] Khalil Alhandawi, Petter Andersson, Massimo Panarotto, Ola Isaksson, and Michael Kokkolaras. Scalable Set-based Design Optimization and Remanufacturing for Meeting Changing Requirements. *Journal of Mechanical Design*, pages 1–20, jul 2020. ISSN 1050-0472. doi: 10.1115/1.4047908. URL https://asmedigitalcollection.asme.org/mechanicaldesign/article/doi/10.1115/1.4047908/1085767/Scalable-Setbased-Design-Optimization-and.

[16] Can Unlusoy. Data-driven inverse aerodynamic design of compressor blading. Master's thesis, Jun 2023. URL https://escholarship.mcgill.ca/concern/theses/s4655n871.

[17] Mehdi Mirza and Simon Osindero. Conditional Generative Adversarial Nets. *arXiv preprint*, 2014. doi: 10.48550/ARXIV.1411.1784. URL https://arxiv.org/abs/1411.1784.

[18] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes, 2013. URL https://arxiv.org/abs/1312.6114.

[19] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density Estimation Using Real NVP. *arXiv preprint*, 2016. doi: 10.48550/ARXIV.1605.08803. URL https://arxiv.org/abs/1605.08803.

[20] Durward Sobek II, Allen Ward, and Jeffrey Liker. Toyota's principles of set-based concurrent engineering. *MIT sloan management review*, 40(2):67–83, 1999. ISSN 0019848X.