Engineering Applications of Artificial Intelligence 129 (2024) 107615

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Research paper

Deep neural operators as accurate surrogates for shape optimization

Khemraj Shukla [a,1], Vivek Oommen [a,1], Ahmad Peyvan [a,1], Michael Penwarden [b,1], Nicholas Plewacki [c], Luis Bravo [c], Anindya Ghoshal [c], Robert M. Kirby [b], George Em Karniadakis [a,*]

[a] Brown University, Providence, RI, 02912, United States of America
[b] University of Utah, Salt Lake City, UT, 84112, United States of America
[c] Weapons and Materials Directorate, U.S. Army Research Laboratory, Aberdeen Proving Ground, MD, 21005, United States of America

ARTICLE INFO

Keywords:
Neural operators
DeepONet
Airfoil shape optimization
Navier–Stokes equations
Surrogate models

ABSTRACT

Deep neural operators, such as DeepONet, have changed the paradigm in high-dimensional nonlinear regression, paving the way for significant generalization and speed-up in computational engineering applications. Here, we investigate the use of DeepONet to infer flow fields around unseen airfoils with the aim of shape constrained optimization, an important design problem in aerodynamics that typically taxes computational resources heavily. We present results that display little to no degradation in prediction accuracy while reducing the online optimization cost by orders of magnitude. We consider NACA airfoils as a test case for our proposed approach, as the four-digit parameterization can easily define their shape. We successfully optimize the constrained NACA four-digit problem with respect to maximizing the lift-to-drag ratio and validate all results by comparing them to a high-order CFD solver. We find that DeepONets have a low generalization error, making them ideal for generating solutions of unseen shapes. Specifically, pressure, density, and velocity fields are accurately inferred at a fraction of a second, hence enabling the use of general objective functions beyond the maximization of the lift-to-drag ratio considered in the current work. Finally, we validate the ability of DeepONet to handle a complex 3D waverider geometry at hypersonic flight by inferring shear stress and heat flux distributions on its surface at unseen angles of attack. The main contribution of this paper is a modular integrated design framework that uses an over-parametrized neural operator as a surrogate model with good generalizability coupled seamlessly with multiple optimization solvers in a plug-and-play mode.

1. Introduction

Two types of neural network solvers for regression problems exist, one for which the network learns the map between input data and output data and the other where neural operators learn function-to-function maps. In this paper, we consider the latter. This recent paradigm shift in perspective, starting with the original paper on the deep operator network or DeepONet (Lu et al., 2021, 2019), provides a new modeling capability useful in engineering design — that is, the ability to replace very complex and computational resource-taxing multiphysics systems with neural operators that can provide functional outputs in real-time. Specifically, unlike physics-informed neural networks (PINNs) (Raissi et al., 2019; Meng et al., 2023a) that require optimization during training and testing, a DeepONet does not require any optimization during inference; hence, it can be used in real-time forecasting, including design, autonomy, control, etc. An architectural diagram of a DeepONet with the commonly used nomenclature for its components is shown in Fig. 1. DeepONets can take a multi-fidelity or multi-modal input (De et al., 2022; Howard et al., 2022; Lu et al., 2022b; Jin et al., 2022; Zhu et al., 2022) in the branch network and can use an independent network as the trunk, a network that represents the output space, e.g., in space–time coordinates or parametric space in a continuous fashion. In some sense, DeepONets can be used as surrogates similarly to reduced order models (ROMs) (Hesthaven and Ubbiali, 2018; Hesthaven et al., 2016; Benner et al., 2017; Williams et al., 2015; Chiavazzo et al., 2014; Lieberman et al., 2010; Bui-Thanh et al., 2008; Benner et al., 2015; Amsallem et al., 2015; Carlberg and Farhat, 2008; Choi et al., 2020). However, unlike ROMs, they are over-parametrized, which leads to both generalizability and robustness to noise that is not possible with ROMs; see the recent work of Kontolati et al. (2022).

In the present work, we investigate the possibility of using DeepONets to represent functions over different solution domains, which

* Corresponding author.
E-mail address: George_Karniadakis@Brown.edu (G.E. Karniadakis).
[1] Equal contribution.

https://doi.org/10.1016/j.engappai.2023.107615
Received 29 January 2023; Received in revised form 26 October 2023; Accepted 23 November 2023
Available online 30 November 2023
0952-1976/

Nomenclature

m	Maximum camber in percentage of the chord
p	Position of maximum camber in percentage of the chord
t	Maximum thickness of the airfoil in percentage of the chord
ρ	Non-dimensional density
u	Non-dimensional velocity of the fluid in x direction
v	Non-dimensional velocity of the fluid in y direction
p	Non-dimensional pressure
T	Non-dimensional temperature
$\mathcal{G}$	Output of the DeepONet model
θ_b	Set of all trainable weights and biases of the branch network
θ_t	Set of all trainable weights and biases of the trunk network
N_ϕ	Number of basis functions learned by the DeepONet
τ_{ij}	Components of viscous stress tensors
f	Generalized input function to the DeepONet
ξ_g	Geometric parameter
c	Chord length
Ma	Mach number
ξ_f	Flow parameter
Re	Reynolds number
Pr	Prandtl number
$\boldsymbol{\alpha}$	Output of the branch network
$\boldsymbol{\phi}$	Output of the trunk network
τ_w	Wall Shear Stress
L	Lift
D	Drag
$\vec{n}$	Unit Normal
$\vec{t}$	Unit Tangent

has not been explored before. In particular, we focus on aerodynamic design by considering the classical problem of optimizing the shape of an airfoil at subsonic flow conditions. Aerodynamic shape optimization (ASO) of airfoils plays a vital role in the design of efficient modern commercial aircraft. The aerodynamic geometric optimization process is usually applied to the airfoil shape that forms the cross sections of the three-dimensional (3D) airfoil. Constrained geometric optimization is performed to reduce the drag force while increasing the lift to enhance the aircraft's fuel efficiency, reducing the transportation cost. The constrained optimization process often requires a numerical model that predicts the flow field around a given airfoil subject to geometric constraints and computes the lift and drag for desired flow conditions. Traditionally, the numerical models are compressible flow numerical solvers, which are computationally intensive to realize the flow field around a complex airfoil accurately. Surrogate models can be introduced to circumvent the time-consuming part of the optimization loop where the numerical solver calculates aerodynamic forces.

ASO traditionally uses two main paradigms: gradient-based and gradient-free optimization approaches. The gradient-based approaches require calculating the cost function derivative with respect to the design variables. When the number of design variables exceeds a certain threshold, the gradient-based optimization becomes infeasible due to its expensive computational cost (Yu et al., 2018). The adjoint formulation is derived from either the Euler or Navier–Stokes equations to make the gradient optimization independent of the design variables (Reuther et al., 1996; Carpentieri et al., 2007; Nadarajah and Jameson, 2001; Srinath and Mittal, 2010). Solving the adjoint equations can be as time-consuming as solving the governing equations (i.e., the rule of thumb is that forward and adjoint solutions taken together are at least twice the cost of the forward solver). Also, the optimization process can fall into a local minima leading to a non-optimized geometry (Chernukhin and Zingg, 2013). Gradient-free approaches (Li et al., 2019; Wu et al., 2022; Yıldız et al., 2022; Aye et al., 2019; Kumar et al., 2023; Anosri et al., 2023; Meng et al., 2023b; Yıldız et al., 2022) can avoid local minima by employing the direct optimization approach that uses many costly numerical simulations. The numerous simulations of the flow can help achieve the global minimum of the cost function at the expense of high computational costs. Surrogate models can be deployed to realize the flow field with acceptable accuracy and an immense speedup compared to the full CFD simulations. The surrogate model can then be used in a gradient-based or a gradient-free global optimization process. The surrogate-based models are usually coupled with gradient-free optimizations such as genetic algorithm and particle swarms optimization (PSO) (Eberhart and Kennedy, 1995) methods. Krige (1951) proposed the Kriging surrogate model that is employed in aerospace design experiments (Liu et al., 2017; Li et al., 2019). The Kriging surrogate model must be trained both before and during the optimization process since the surrogate model is usually inaccurate when based solely on the initial training. High generalization error of the Kriging surrogate model results in inaccurate model prediction at the initial training stage. As proposed in this work, a deep operator network (DeepONet) can alleviate this issue since it maps a function to another function, significantly improving the generalization error (Lu et al., 2021).

The parameterization of the geometry drastically affects ASO's computational cost and accuracy. Parameterizing the geometry reduces the number of design variables the optimization algorithm must search. Reducing the number of design variables simplifies the optimization process, as well as the constraints imposed by the user, and decreases the sensitivity to noise. The parametric model must reproduce a wide range of airfoil shapes and keep the number of design variables minimal. Various geometric parametric models have been employed for ASO. Carpentieri et al. (2007) employed orthogonal Chebyshev polynomials to construct the airfoil curves. An orthogonal polynomial is used to cover the entire design space. Lepine et al. (2001) used Non-Uniform Rational Basis Spline (NURBS) to parameterize a large class of airfoil shapes by only using 13 control points. Following the Lepine et al. (2001) idea, Srinath and Mittal (2010) also employed NURBS for the parameterization. Other researchers have used B-Spline (Wang et al., 2019), and Bezier (Papadimitriou and Papadimitriou, 2016) curves, Hicks and Henn's functions (Hicks and Henne, 1978) for airfoil shape construction. Painchaud-Ouellet et al. (2006) used NURBS for the shape optimization of an airfoil within transonic regimes. They showed that using NURBS ensures the regularity of the airfoil shape. The airfoil shape can also be constructed using a deformation method (Hicks and Henne, 1978). This method adds a linear combination of bumps to a baseline airfoil shape for parameterization (Chen and Fidkowski, 2017; He et al., 2019). The Class function/shape function Transformation (CST) approach (Wu et al., 2019) employs Bernstein polynomials (Akram and Kim, 2021) to parameterize airfoils and other aerodynamic geometries. Other researchers employed proper orthogonal decomposition (POD) (Wu et al., 2019) to reduce the number of design variables. In the current study, we employ the NACA 4-digit airfoil and NURBS parameterizations for the airfoil shape construction.

With the significant advancement in computational power, Deep Neural Network (DNN) tools have gained much attention for serving as accurate surrogate models in a broad spectrum of scientific disciplines (Zhang et al., 2021; Zhiwei et al., 2020; Renganathan et al., 2021). In prior work, robust neural network-based algorithms for time-series classifications were developed (Xing et al., 2022; Xiao et al., 2021). Neural network models are also employed for diagnosing bearing faults (Mishra et al., 2022c,b,a). The DNN approach can be readily trained for numerous input design variables to predict the cost function

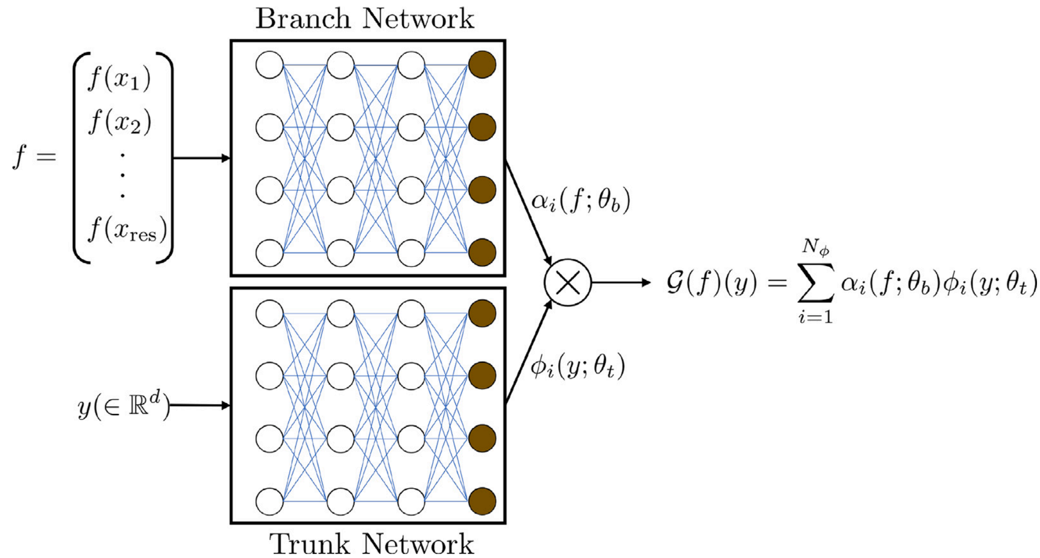


Fig. 1. A schematic representation of a DeepONet that is trained to learn the mapping from the input function f to the output function $\mathcal{G}(f)(y)$, evaluated at y. DeepOnet consists of a branch and a trunk network.

of the optimization loop. Du et al. (2021) trained a feed-forward DNN to receive airfoil shapes and predict drag and lift coefficients. They also used RNN models for estimating the pressure coefficient. The optimal airfoil design determined using the surrogate model was compared with an airfoil design obtained with a CFD-based optimization process (Du et al., 2021). Hao et al. (2023) provides a comparative study of neural operator learning methods for flow field prediction around airfoils. Liao et al. (2021) designed a surrogate model using a multi-fidelity Convolutional Neural Network (CNN) with transfer learning. This learning method transfers the information learned in a specific domain to a similar field. The low-fidelity samples are taken as the source, and the high-fidelity ones are assigned as targets. Tao and Sun (2019) introduced a Deep Belief Network (DBN) to be trained with low-fidelity data. The trained DBN was later combined with high-fidelity data using regression to create a surrogate model for shape optimization. Existing surrogate models for shape optimization are all trained to predict lift, drag, or pressure coefficients. For example, Zhao et al. (2023) uses a DeepONet to learn the mapping from iced airfoil geometries to their aerodynamic coefficients. In contrast, the flow field around the aerodynamic shape is not inferred. Prior works have also investigated the capabilities and limitations of the different neural operators in various benchmark cases in Lu et al. (2022a). The recent Geo-FNO (Li et al., 2022) and CORAL (Serrano et al., 2023) propose neural operator-based models capable of learning solutions of PDEs on general geometries. However, both these works ignore the contribution of the viscous forces while computing the lift and drag forces, making it less realistic. Here, we construct a surrogate model that predicts the viscous flow field around the airfoil shape using a DeepONet. Predicting the flow field provides additional information that can be used in the cost function of the optimization loop. We aim to develop an aerodynamic shape optimization framework using a surrogate model that can infer the flow field around the geometry. The surrogate model is constructed using a DeepONet and is trained using high-fidelity CFD simulations of airfoils in a subsonic flow regime. The surrogate model is then implemented in two different optimization frameworks for shape optimization. The novelties of this study include the following:

- Generating a DeepONet-based surrogate model is an efficient and inexpensive instantiation of the exorbitant CFD solver.
- The surrogate model is invariant to the input space, which can be defined as low or high-dimensional parameterizations.
- Prediction of high-dimensional flow field can be used for various cost functions in the constrained optimization loop.
- Drag and lift coefficients are computed using the inferred high-dimensional flow field, resulting in more accurate predictions.
- Integration of the Dakota optimization framework with the DeepONet surrogate.

The remainder of the article is organized as follows. We begin by defining the optimization process. We then present the data generation for training the surrogate model using the open-source spectral/hp element Nektar++ CFD solver. The training procedure of the DeepONet-based surrogate model is explained. Later, the optimization results using two different methods are represented. Finally, we summarize our findings in the Conclusions section. In the Appendix, we verify the accuracy of the data generated by repeating selected simulations using different codes. Additionally, we provide validation of the Dakota optimizer by comparing it against multiple approaches. We find that all the approaches studied herein converge to the same solution.

2. Problem setup

To highlight the capabilities of DeepONets as function-to-function maps that can be used within the airfoil shape optimization process, we start by reviewing the traditional end-to-end shape optimization pipeline augmented with DeepONet training. A schematic of the pipeline is shown in Fig. 2. Reviewing the figure from upper left to lower right, we start with an experimental setup. This represents the determination of the feasible set from which the parametric airfoils in training will be drawn, the aerodynamic conditions, and any other engineering constraints related to the problem. We choose NACA four-digit airfoils as our geometric representation, which provides the upper and lower surface equations, given a random draw of parameters. This representation is then used in three places: (1) to directly mesh the flowfield around the airfoil, for which we use Gmsh; (2) to use in querying the surrogate model on the surface of the airfoil when predicting the objective lift-to-drag ratio; and (3) as the branch input to the DeepONet function-to-function map, which is pre-processed using NURBS to lower the dimensionality.

In terms of creating the surrogate model, this can be viewed as the following forward problem. One deciphers the geometric and flow parameters, which are then used to create the inputs to a CFD solver: a geometric representation of the airfoil and its corresponding mesh to be used for approximating the flowfield and a flow parameter file. These are then input to a flow solver — in our case, the CFD solver Nektar++. Results from this solver are used to generate training data

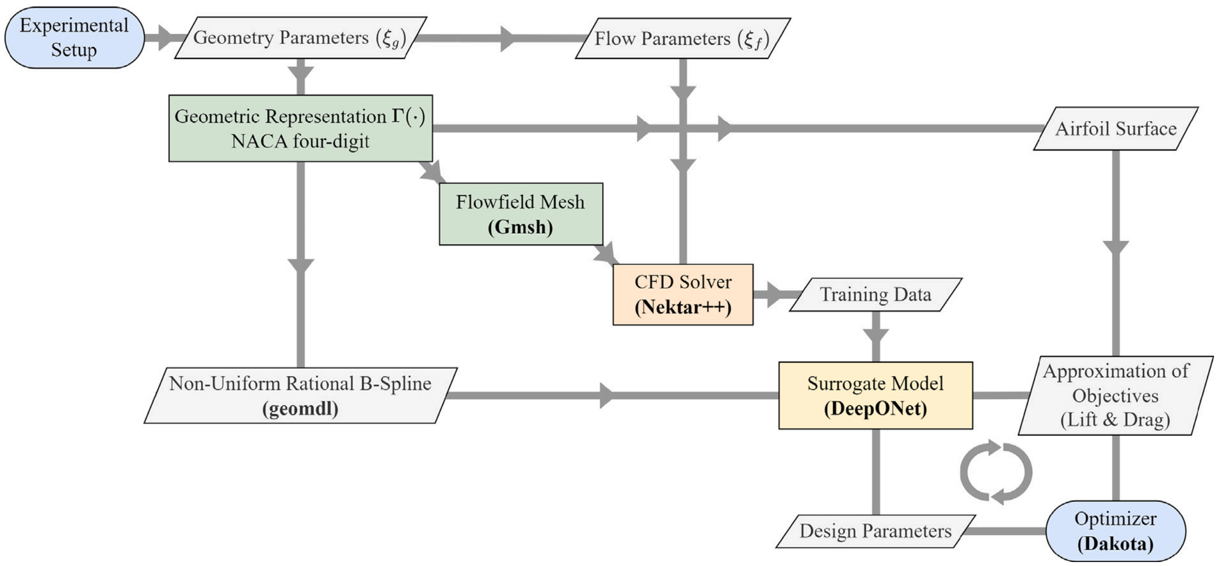


Fig. 2. Diagram of constrained airfoil geometry optimization with a DeepONet surrogate model. The blue ovals indicate the beginning and end states of the method. The main intermediate steps are highlighted in colored boxes, whereas the auxiliary steps are gray parallelograms.

used to train our DeepONet surrogate. Finally, the lower-right quadrant of the diagram denotes the shape optimization process using the trained DeepONet surrogate. This process is iterative as the optimizer, for which we use Dakota, queries the DeepOnet surrogate model with new design parameters until the objective function is sufficiently minimized. This process differs from existing approaches because the bulk of the optimization process is done offline. Generating data using the CFD solver can be expensive, but with the final trained DeepOnet, the online cost of geometric optimization is orders of magnitude faster than other methods. Furthermore, as long as the objective can be created by the DeepONet flowfields trained over, a different objective function can be defined, not only lift-to-drag and an airfoil can be quickly optimized with respect to the new objective without any additional cost.

For our experiments, we have focused on using 2D compressible Navier–Stokes fields at Reynolds number $Re = 500$ and Mach number $Ma = 0.5$ for our training. These values have been chosen to allow us to focus on DeepONet's ability to capture variations in domains (instead of the compounding effects of unsteadiness, etc.). Given the success of DeepONets under this experimental setup, future work will extend this pipeline to more complex flows and experimental conditions, such as varying the angle of attack, morphing geometry, handling unsteady flow, or going into the high-speed flow regimes.

3. Methodology

3.1. Data generation

For each example in the dataset, we define a set of geometric parameters (ξ_g) and flow parameters (ξ_f). The geometric parameters are then converted into another representation, such as surface coordinates derived from the NACA airfoil equations; this transformation is given by $\Gamma(\xi_g)$. These points are then used to mesh the flowfield domain with Gmsh (Geuzaine and Remacle, 2020-06-22), which is then input into the flowfield simulation software Nektar++ (Cantwell et al., 2015; Moxey et al., 2020). We obtain the solutions fields from Nektar++ through post-processing for density, x-velocity, y-velocity, and pressure. This is saved in two sets, one in a subdomain around the airfoil for training the DeepONet and one at airfoil sensors on the surface for validation. We also save the Nektar++ lift and drag forces to validate the discrete integration of the airfoil forces.

3.1.1. Geometry generation

NACA 4-digit airfoils provide an excellent testbed application for geometry optimization using DeepONets since they can represent a wide range of shapes from well-known and studied parametrized geometric equations. Our geometry optimization framework could be easily extended to any parametrized geometry, such as NACA 5-digits or beyond. Following Jacobs et al. (1933), we define the parametric equations for the surface of an airfoil with a chord length of one as follows:

$$y_t = \frac{t}{0.2}\left(a_0\sqrt{x} + a_1 x + a_2 x^2 + a_3 x^3 + a_x x^4\right) \tag{1}$$

$$y_c = \begin{cases} \frac{m}{p^2}\left(2px - x^2\right) & \text{if } x < p \\ \frac{m}{(1-p)^2}\left(1 - 2p + 2px - x^2\right) & \text{if } x > p \end{cases} \qquad \theta = \tan^{-1}\left(\frac{dy_c}{dx}\right) \tag{2}$$

$$x_u = x - y_t \sin(\theta), \quad y_u = y_c + y_t \cos(\theta)$$
$$x_l = x + y_t \sin(\theta), \quad y_l = y_c - y_t \cos(\theta) \tag{3}$$

where $a_0 = 0.2969, a_1 = -0.1260, a_2 = -0.3516, a_3 = 0.2843, a_4 = -0.1015$. We can, therefore, define our geometry as a point cloud with coordinate sets $\left(x_u, y_u, x_l, y_l\right)$ parametrized by $\xi_g = (t, p, m)$. A series of x locations are found using cosine spacing with 100 points; this increases the geometric fidelity around the leading and trailing edge, increasing the accuracy of the mesh and flowfield simulation at these important locations. To simplify the problem and reduce the likelihood of flow separation or turbulence, we constrain ξ_g. The maximum thickness (t) is set to a constant 0.15, and the domain of the parametric space left by the position of maximum camber (p) and maximum camber (m) is $p \times m \in [0.2, 0.5] \times [0.0, 0.09]$. Therefore, for one geometric example in either the train or test set, we draw a ξ_g tuple where the variable parameters are drawn from a uniform distribution within their domains. We perform this draw 50 times and obtain the surface coordinates from Eqs. (1)–(3), splitting it into 40 training and 10 testing examples as seen in Fig. 3. The test/train split is an essential aspect of deep learning; here, we choose a relatively sparse sampling highlighting the ability of DeepONets to generalize well to unseen parameters.

Next, to lower the input dimensionality into the DeepONet branch, we fit the airfoil surface with Non-Uniform Rational B-Splines (NURBS) with 30 control points using geomdl (Bingol and Krishnamurthy, 2019). This reduces the input dimensionality from 200 (x, y) pairs to only 30.

Fig. 3. Train and test set geometries sampled over the ξ_g domain. Note that the numbers correspond to NACA airfoils, and duplicate numbers are due to rounding to the nearest integer for readability. Therefore, the true NACA parameters are drawn from a uniform distribution and are real-valued.

3.1.2. Mesh generation

The meshes are generated using Gmsh (Geuzaine and Remacle, 2020-06-22) for parametrized airfoil geometry with a minimum characteristic length of 0.01 at parametrized locations. The airfoil flowfields are meshed using the 200 surface points exactly from the NACA equations, not the NURBS fit, which contains some inaccuracy. A spline function is used to represent the 1D geometry of the boundaries of airfoils. To resolve the flow at leading and trailing edges, meshes are refined by using the splitting approach (Mark et al., 2008). The NURBS low-dimensional representation is a step to reduce overparameterization in the DeepONet, which is unnecessary for generating high-fidelity training data. The mesh generation for all 50 airfoils is automated using Gmsh's Python API integrated with the geometry generation in Python, so no manual operations are needed. Fig. 4 shows the mesh of the entire simulated domain Ω_S, which is then input into Nektar++ along with the flow parameters to generate the DeepONet training data. As seen in the figure, only a subset of the solved steady-state domain Ω_T is used in training the DeepONet. This is because the DeepOnet is a function-to-function map and does not strictly obey boundary conditions or is affected by phenomena such as reflections due to the boundaries. It performs regression on the dataset and not the solving of the system of equations and, therefore, can be taken as a smaller domain, even without freestream conditions. Since the objective is geometric optimization, this subdomain simplifies the DeepONet training problem and cost of training.

3.1.3. Flowfield simulation

We used a compressible flow solver implemented in Nektar++ to generate the flow field data. Nektar++ (www.nektar.info) is an open-source software framework (Cantwell et al., 2015; Moxey et al., 2020) designed to support the development of high-performance, scalable solvers for partial differential equations using the spectral/hp element method. The 2D compressible flow solver uses the two-dimensional compressible Navier–Stokes equations expressed as,

$$
\begin{aligned}
&\frac{\partial \rho}{\partial t}+\frac{\partial \rho u}{\partial x}+\frac{\partial \rho v}{\partial y}=0\\
&\frac{\partial \rho u}{\partial t}+\frac{\partial \rho u^2+p}{\partial x}+\frac{\partial \rho u v}{\partial y}=\frac{1}{Re}\left(\frac{\partial \tau_{xx}}{\partial x}+\frac{\partial \tau_{yx}}{\partial y}\right)\\
&\frac{\partial \rho v}{\partial t}+\frac{\partial \rho u v}{\partial x}+\frac{\partial \rho v^2+p}{\partial y}=\frac{1}{Re}\left(\frac{\partial \tau_{xy}}{\partial x}+\frac{\partial \tau_{yy}}{\partial y}\right)\\
&\frac{\partial E}{\partial t}+\frac{\partial (E+p)u}{\partial x}+\frac{\partial (E+p)v}{\partial y}\\
&\quad=\frac{1}{Re}\left[\frac{\partial\left(u\tau_{xx}+v\tau_{xy}+\kappa\frac{\partial T}{\partial x}\right)}{\partial x}+\frac{\partial\left(u\tau_{xy}+v\tau_{yy}+\kappa\frac{\partial T}{\partial y}\right)}{\partial y}\right],
\end{aligned}
\tag{4}
$$

where $p=R\rho T$, $R=\frac{1}{\gamma Ma^2}$, $k=\frac{\gamma}{\gamma-1}\frac{\mu R}{Pr}$, with μ being the non-dimensional viscosity and computed by using the Sutherland law as

$$
\mu=\frac{T^{3/2}}{Re}\frac{1+C/T_\infty}{T+C/T_\infty}. \tag{5}
$$

Expressions for τ_{xx}, τ_{yy}, and τ_{xy} are as follows

$$
\begin{aligned}
\tau_{xx}&=2\mu\left(u_x-\frac{u_x+v_y}{3}\right),\\
\tau_{yy}&=2\mu\left(v_y-\frac{u_x+v_y}{3}\right),\\
\tau_{xy}&=\mu\left(u_y+v_x\right).
\end{aligned}
$$

We aim to simulate the flow past airfoils by solving the compressible Navier–Stokes equations given by Eq. (4) with free-stream parameters $M_\infty=0.5$, $Re_{L=1}=500$, $u_\infty=1$, $v_\infty=0$, $T_\infty=1$, AoA $=0$ and $Pr=0.72$. The flow domain in non-dimensional units is $[-3,11]\times[-3,3]$ and discretized by conforming triangular elements. To solve (4), we use the discontinuous Galerkin spectral element method (DGSEM) with basis functions spanned in 2D by Legendre polynomials of the second degree. For advection and diffusion terms, weak and interior penalty-based dG approach with Roe upwinding is used in space. A diagonally implicit Runge–Kutta (DIRK) method is used as a time integrator for the advection and diffusion terms. The boundary conditions of inflow, outflow, adiabatic wall at the airfoil surface, and high-order boundary conditions at the top and bottom are imposed weakly. For detailed descriptions of the solvers and methods, readers are encouraged to read (Mengaldo et al., 2014).

3.2. Surrogate model: DeepONet

3.2.1. Brief review of DeepONets

Neural operators are neural network models developed based on the universal operator approximation theorem (Chen and Chen, 1995). The neural operators learn the mapping between spaces of function and directly learn the underlying operator from the available training data. DeepONets (Lu et al., 2021) and Fourier Neural Operators (FNO) (Li et al., 2020) are the two popular neural operators extensively used for solving a wide spectrum of problems in diverse scientific areas. A DeepONet consists of a branch network that encodes the input function and a trunk network that learns a collection of basis functions. The DeepONet output is computed by taking the inner product between the branch and trunk network outputs.

3.2.2. Training and testing of DeepONets

We train four different DeepONet models to learn the pressure (p), density (ρ), and velocity (u, v) fields for a given airfoil geometry (ξ_g) from the training data. The trunk network learns a collection of basis ($\boldsymbol{\phi}$) as functions of spatial coordinates, and the branch network learns

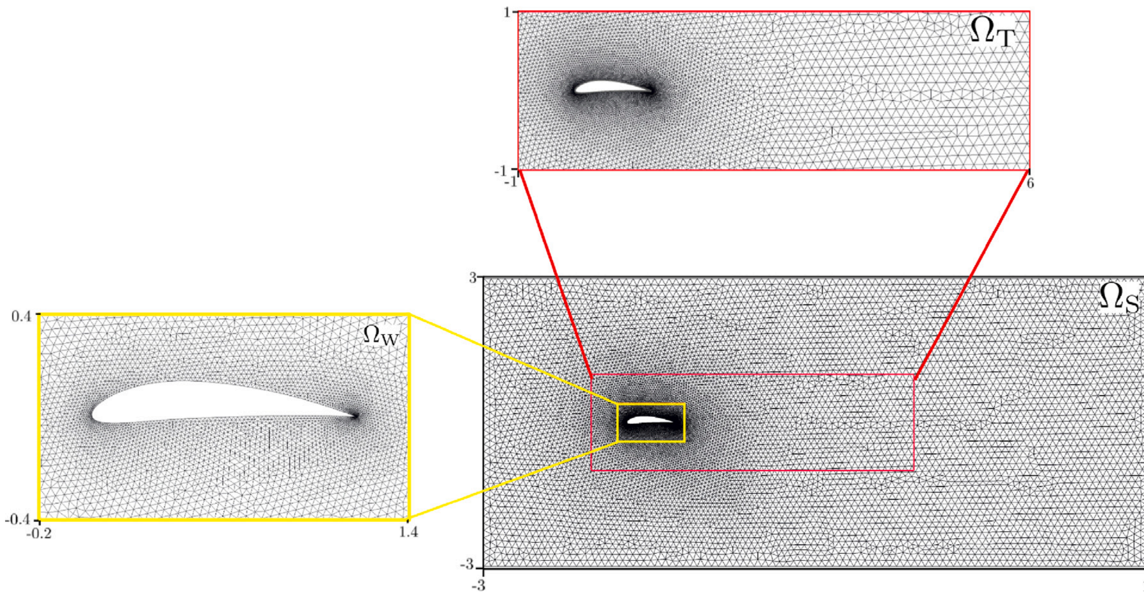


Fig. 4. Discretization of an airfoil with bounding domain. Ω_S represents the entire domain over which the compressible Navier–Stokes equations are solved. Ω_T (inset image with red border) represents the domain for which a DeepONet is trained. The inset image with yellow boundary represents the subdomain (Ω_W), showing the mesh refinement around the airfoil.

Table 1
Hyperparameters of NURBS-DeepONet and the Parameter DeepONet.

NURBS-DeepONet	
Branch Network Architecture:	[30,100,100,50]
Branch Network Activation:	tanh
Trunk Network Architecture:	[2,100,100,50]
Trunk Network Activation:	tanh
N_ϕ:	50
Optimizer:	Adam
Learning Rate:	1.00E−04

Parameter-DeepONet	
Branch Network Architecture:	[2,100,100,50]
Branch Network Activation:	tanh
Trunk Network Architecture:	[2,100,100,50]
Trunk Network Activation:	tanh
N_ϕ:	50
Optimizer:	Adam
Learning Rate:	1.00E−04

the corresponding coefficients ($\boldsymbol{\alpha}$) as a function of the airfoil geometry. The DeepONet output is defined as

$$\mathcal{G}^q(\xi_g)(x,y) = \sum_{i=1}^{N_\phi} \alpha_i(\xi_g;\theta_b^q)\phi_i(x,y;\theta_t^q) \quad q \in \{p,\rho,u,v\}. \tag{6}$$

In the case of airfoils, the geometry can be fed into the branch network by directly providing the geometric parameter, ξ_g. However, ξ_g need not exist explicitly for a general arbitrary geometry. Under such a scenario, the geometry is often represented using the NURBS control points. To demonstrate the effectiveness of using a DeepONet in either of the situations, we investigate parameter-DeepONet that directly takes ξ_g as the branch network input and NURBS-DeepONet that takes NURBS control points of the airfoil geometry as the input to the branch network. The hyperparameters of the DeepONet used in this study are provided in Table 1.

3.2.3. Lift and drag calculation

We present two ways to evaluate the objective function — in our case, lift-to-drag. The map created by the trained DeepONets takes an input geometry and spatial (x,y) location in the output space and returns the field's value at that location. So it follows that to estimate the lift and drag, we evaluate the discrete integral over the surface of the airfoil, for which the (x,y) points are easily generated for objective function queries by Eqs. (1)–(3). The discrete integrals for lift and drag are given by Eqs. (8) and (9) subject to the approximation of wall shear stress in Eq. (7).

$$\tau_w = \mu \frac{dU}{d\vec{n}} \tag{7}$$

$$L = \int dF_y = \sum p\vec{n_y}ds + \sum \tau_w \vec{t_y} ds \tag{8}$$

$$D = \int dF_x = \sum p\vec{n_x}ds + \sum \tau_w \vec{t_x} ds \tag{9}$$

The pressure is directly obtained from one of the DeepONet predictions for the first term in the aerodynamics forces. In the second term, the viscosity $\mu(p,\rho)$ is obtained by Eq. (5) as a function of the DeepONets for pressure and density. Finally, the change in speed of the flow over the airfoil surface $\frac{dU}{d\vec{n}}$, is obtained in two ways:

1. Finite-difference **(A)**:

$$\frac{dU}{d\vec{n}} = \frac{-U_2 + 4U_1 - 3U_0}{2h} \tag{10}$$

2. Automatic-differentiation **(B)**:

$$\frac{dU}{d\vec{n}} = (v_y - u_x)sin\theta cos\theta - v_x sin^2\theta + u_y cos^2\theta \tag{11}$$

where $h = 0.001$ and θ is the angle between the x-y axis and each segment's normal-tangential axis. Approach (A) is a second-order forward finite difference approximation obtained by sampling the x-velocity and y-velocity DeepONets at the appropriate locations defined by the surface normal and spacing h. Approach (B), derived in Appendix A.4, utilizes the now well-known development in automatic differentiation (Baydin et al., 2018), primarily utilized in physics-informed machine learning for approximating partial derivatives to obtain the PDE residual. Here, since the DeepONet directly takes in the spatial (x,y) coordinates and outputs the velocity components (u,v), the computational graph is complete, and the partials (u_x, u_y, v_x, v_y) can be estimated with this method. The required sampling for each method is shown in Fig. 5. While (A) takes three times the number of point evaluations, (B) requires the gradients to be computed so the cost of each can be viewed

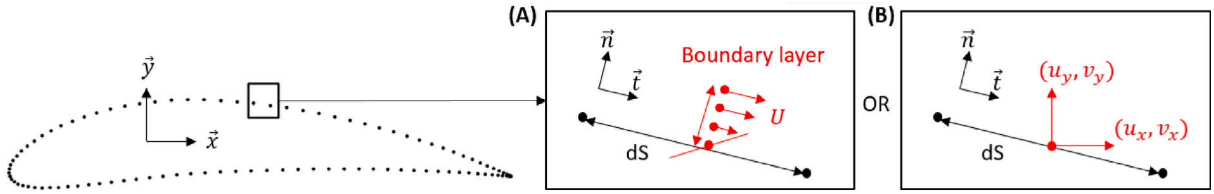


Fig. 5. Illustration of the discrete integral for lift and drag. The points in red indicate the surrogate model samples used in the construction of the approximation to $\frac{dU}{d\vec{n}}$ with a finite difference approximation of the gradient or the automatic differentiation approximation using the direct network gradients.

Table 2
Relative L^2 errors of the state variables trained DeepONet models.

	NURBS-DeepONet		Parameter-DeepONet	
	Train rel. L^2 error	Test rel. L^2 error	Train rel. L^2 error	Test rel. L^2 error
$\mathcal{G}^p$	4.68e−03	6.05e−03	5.23e−03	6.85e−03
$\mathcal{G}^u$	4.97e−03	6.21e−03	4.12e−03	5.38e−03
$\mathcal{G}^v$	3.73e−03	4.60e−03	3.31e−03	4.25e−03
$\mathcal{G}^\rho$	4.57e−03	5.89e−03	4.00e−03	5.18e−03

as comparable. However, the flexibility of using automatic differentiation in this way may allow for more complex objective functions in the future, given the right mapping and subsequent computational graph.

4. Results

4.1. Results from DeepONet model

The training and testing relative L^2 error of the NURBS and parameter-based DeepONets for all four different fields are reported in Table 2. We observe that the NURBS and parameter-based DeepONets predict fields with similar accuracy. NURBS-DeepONet has marginally better predictions of the pressure field, while parameter-DeepONet generates marginally better predictions for velocity and density fields. The main takeaway is that either representation is sufficient for the geometry optimization of this experiment. However, we must consider that, in the future, more complex geometries may be used, particularly in the sense of local morphing. Therefore, the NURBS representation will likely be necessary as a direct parameter mapping may miss local nuances. The predicted fields and the absolute pointwise error by the best DeepONet models for the flowfield parameters are shown in Fig. 7. It can be seen that the global prediction is, in general, accurate; the error is primarily localized to the airfoil's leading edge. In the future, adaptive weighting schemes will be used to improve the DeepONet training, particularly at the points of difficulty, such as the surface and leading edge. The corresponding relative L^2 error of the fields over the entire dataset is shown in Fig. 13. We observe minimal generalization error and that DeepONets are globally accurate (see Fig. 6).

Regarding geometry optimization, we must concern ourselves not only with the global flowfield accuracy but also with the accuracy on the surface of the airfoil in particular. Fig. 8 shows the corresponding surface prediction plots for the same airfoil presented in Fig. 7 as a function of the x-direction over the airfoil. We observe good accuracy in the pressure and density fields, which will provide very accurate predictions of the lift and drag force components due to pressure as well as the viscosity $\mu(p,\rho)$, which is a function of these fields per Eq. (5). The surface's x and y velocity fields do not agree because the DeepONet does not strictly obey a no-slip condition. However, aside from the leading edge, the prediction errors are close to zero. Furthermore, the fields are not directly related to the objective lift-to-drag but indirectly related through the estimate of the change in flow speed over the surface $\frac{dU}{d\vec{n}}$ obtained by approaches (A) and (B) in Eq. (10) and (11). Therefore, the inaccuracy does not significantly affect the overall objective prediction. This is corroborated by Fig. 9, which shows the sorted lift-to-drag ratio for the entire dataset. The results of both numerical integration approaches (A) and (B) are very accurate when compared to the stored lift-to-drag results from the Nektar++ data generation step. We can also see in the error plot that it is quite uniform, and there are no discernible biases in the geometric parameter space, indicating that we have learned the entire space well enough for the final optimized result to be accurate.

4.2. Constrained shape optimization results

The objective of constrained shape optimization is to maximize the lift-to-drag ratio over a feasible region of parameters, which are m and p for this case. Eq. (12) gives this objective in the form of a minimization problem, as is standard for most optimizers that perform gradient-based or gradient-free optimization. Therefore, the definition of a constrained optimization problem for airfoil is expressed as

$$\begin{aligned} \underset{m,p}{\text{minimize}} \quad & -f(m,p) \\ \text{subject to} \quad & m_{\min} \le m \le m_{\max} \\ & p_{\min} \le p \le p_{\max}, \end{aligned} \tag{12}$$

where $f(m,p)$ represents ratio of lift to drag and $[m_{min}, m_{max}]$ and $[p_{min}, p_{max}]$ are bounds for feasible search region.

One of the present study's goals is to optimize the shape for any arbitrary geometry. Therefore, we integrated the DeepONet-based surrogate model with Dakota, which is a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis (Adams et al., 2022). Dakota is freely available and offers a very efficient and scalable implementation. We integrated the DeepONet with Dakota in a modular approach as shown in Algorithm 1, where $\mathcal{D}$ is an algorithm chosen from a set of optimizers provided by Dakota and DeepONet-based model Φ is passed as an argument to $\mathcal{D}$. For example, to achieve the solution of Eq. (12), we use an efficient global algorithm (EGO), which is a derivative-free approach that uses a Gaussian process model for the optimization of the expected improvement function and is based on the NCSU Direct algorithm (Finkel and Kelley, 2004). The reason behind choosing this method is to avoid tuning various hyperparameters. To use the algorithm to solve the problem in Eq. (12), we set a seed, which is to be used for Latin Hypercube Sampling (LHS) to generate the initial set of points for constructing the initial Gaussian process. To gain efficiency, we used batch-sequential parallelization offered by Dakota on an eight-core CPU (2.3 GHz Intel core i9).

The constrained optimization landscapes for approaches (A) and (B) are shown in Fig. 10 obtained by brute force evaluation of the respective objectives. Also plotted are the locations of the dataset in the parameter space $(p,m) \in [0.2, 0.5] \times [0.0, 0.09]$, displaying the sparse sampling used to obtain accurate optimization results. We also observe that the landscape for this experimental setup is simple and

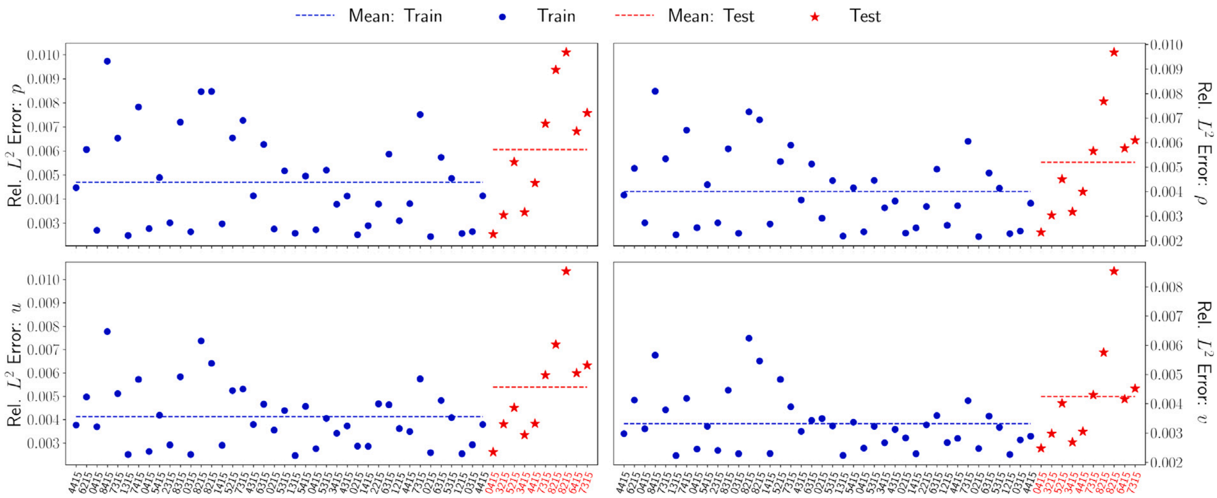


Fig. 6. Error Scatter Plots. The relative L^2 error corresponding to each of the train and test samples for pressure, velocity, and density fields, with respect to the DeepONet predictions, are shown in this figure.

Algorithm 1: Integration of DeepONet-based surrogate model with Dakota.

Require: m, p, x, y: maximum camber, the position of maximum camber, spatial coordinates for flow-field prediction
Require: Trained DeepONet: $\mathcal{G}(m, p, x, y)$
Require: $\mathcal{D} : \mathcal{D} \in$ Algorithms in Dakota
Require: $\Psi(u, v, \rho, p, \xi_g)$: Function producing the lift L and drag D
Require: N: Number of objective function evaluations
Require: $m_{\min}, m_{\max}, p_{min}, p_{\max}$: Bounds of feasible region
$n \leftarrow 0$, $m_{\text{opt}} \leftarrow m_0$, $p_{\text{opt}} \leftarrow p_0$ ▷ Initialize
while n != N **do**
$m_{\text{opt}}, p_{\text{opt}} \leftarrow \mathcal{D}(-f(m_{\text{opt}}, p_{\text{opt}}), m_{\min}, m_{\max}, p_{min}, p_{\max}, \mathcal{G}, \Psi)$ ▷ Optimization process for parameters
$n \leftarrow n + 1$
end while

convex. In future work, more complex conditions like local morphing will make the landscape more complex, likely nonconvex, requiring more sophisticated optimization methods. While not needed here, we still utilize state-of-the-art optimization in our framework with Dakota. Additionally, we can see that the local minimum given the constrained parameter bounds, set to the dataset sampling bounds, is at the boundary. This implies that the global minimum lies outside of our trained bounds, which is not known a priori when generating data. In future work, we hope to evaluate DeepONets potential for extrapolating outside the trained bounds, potentially using transfer learning.

The optimization process finds the minimizer of function in 15 evaluations, and the optimum value which maximizes L/D is $(m_*, p_*) = (0.2, 0.067)$. To achieve consistency in the optimization process, we ran the EGO algorithm 15 times with different seeds and observed the same optimal point. Furthermore, we validated the optimization results with a comparison to other approaches in Appendix A.3. The results obtained from all the approaches are in excellent agreement and are reported in detail in Table 7 along with their wall clock times. Finally, the most significant contribution of the paper is shown in Table 3. As we can see, the integration of DeepONet into an airfoil geometry optimization framework has lowered the online cost of new objective evaluations by $32,000+$ times. This makes it entirely possible to have almost real-time optimization results, costing a few minutes instead of days. Furthermore, the trained models can be put on any hardware, such as a standard laptop, and real-time accurate flowfields can be predicted in seconds, meaning the geometry optimization is not hardware dependent at test time. We have demonstrated that

Table 3
Relative cost of single objective function evaluation during geometry optimization A Flowfield mapping with finite-difference approximation. B Flowfield mapping with automatic-differentiation approximation.

Model type	Relative cost of single objective function evaluation
Baseline CFD (Nektar++)	32,253
DeepONet (**A**)	1.34
DeepONet (**B**)	1

integrating DeepONets into a geometry optimization pipeline suffers little in accuracy and provides the tradeoff of obtaining and training on an offline dataset with almost instantaneous optimization results when used online compared to a traditional CFD method.

To validate the parameters of optimized airfoil ($p = 0.2, 0.067$), we compare the u velocity field in Fig. 11 obtained from trained DeepONet and Nektar++. In general, the velocity contour plots show an excellent agreement and therefore validate the workflow of shape optimization presented in this work. A closer look at error plot suggests higher errors along the surface of the airfoil, with a maximum of 0.046 for normalized x-velocity. The results can be further improved by giving more importance to the region near the airfoil surface (via proper weighting) during training of the DeepONet.

5. Hypersonic waverider study

To demonstrate DeepONet's capability to generate an accurate surrogate model on three-dimensional complex fields, hypersonic aerothermodynamic data was generated using the US3D commercial CFD package. The analysis was based on the 3D waverider model and experimental data measured at the Arnold Engineering Development Center (AEDC) Hypervelocity Wind Tunnel Number 9 (Kammeyer and Gillum, 1994). This geometry will be hereafter referred to as the AEDC waverider (see Fig. 12). US3D is a state-of-the-art analysis tool developed as a collaborative effort between NASA Ames, the University of Minnesota, and VirtusAero, Inc. This code is massively parallel using the Message Passing Interface (MPI) libraries and has been deployed on the Department of Defense (DoD) High-Performance Computing (HPC) system Warhawk, which was used in this work. US3D solves the compressible Navier–Stokes equations on an unstructured finite-volume mesh with high-order, low-dissipation fluxes. The solver has been tailored to excel at the complex evaluation of hypersonic flows including strong shocks, shock boundary layer interactions, and plasma dynamics, and has well-demonstrated accuracy for applied hypersonic configurations (Candler et al., 2015).

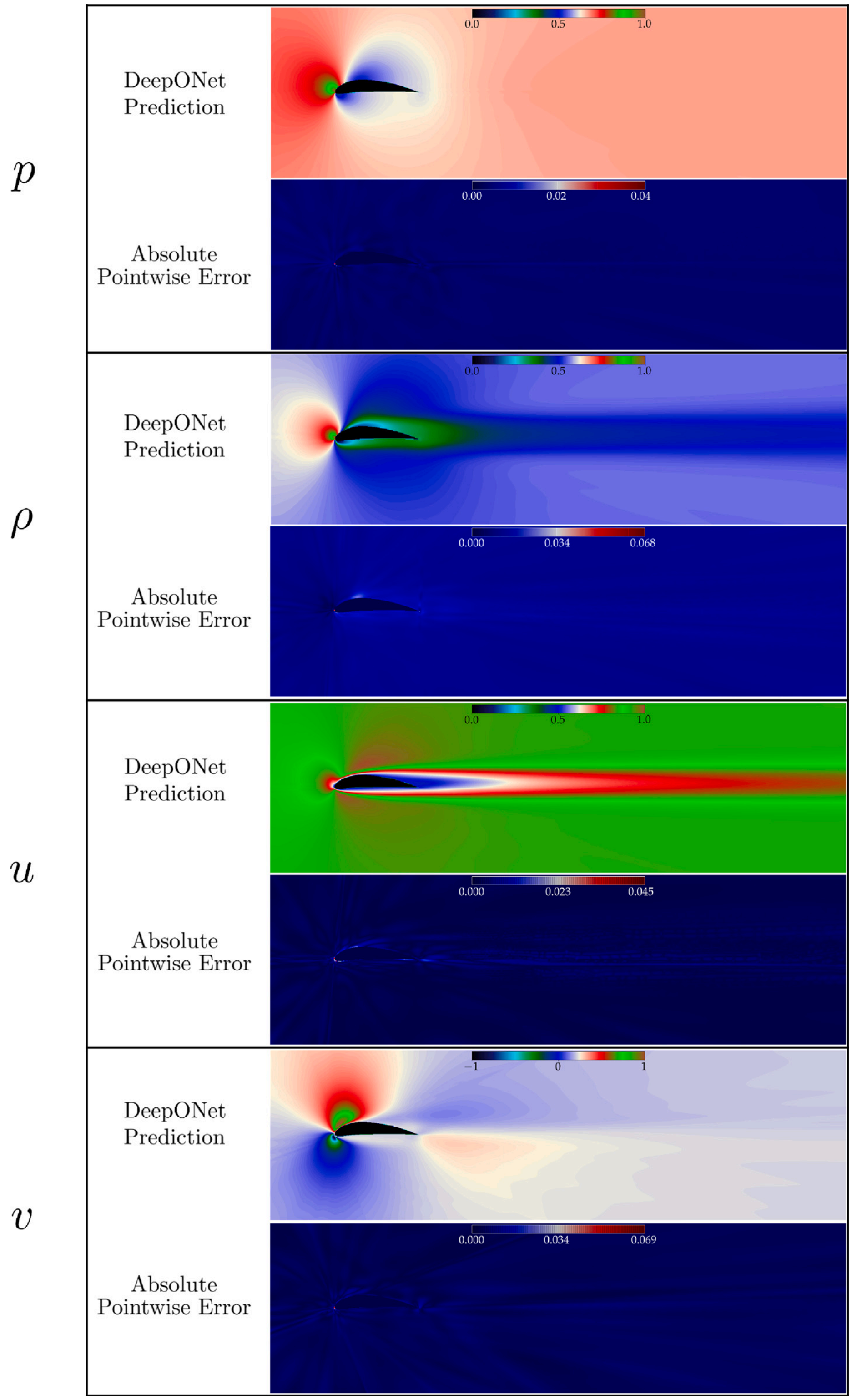


Fig. 7. DeepONet Predictions. The pressure, density, and velocity fields around the test set airfoil NACA 7315 predicted by the DeepONet, and the corresponding pointwise absolute errors are also provided.

In this effort, the free stream and surface boundary conditions were selected, consistent with the reported experimental conditions as follows: $\rho_{\text{inf}} = 0.5644$ kg/m^3, $T_{\text{inf}} = 72.77$ K, and $v_{\text{inf}} = 1279.25$ m/s with Mach number of 7.36. The surface temperature of the AEDC waverider is isothermal and held at 300 K based on experimental conditions. Turbulence was modeled using the classical Menter-SST Reynolds Averaged Navier Stokes (RANS) formulation (with a vorticity source term) along with 5 species of chemical kinetics to handle the non-equilibrium chemistry. The angle of attack was modified in 1-degree increments between -10 and $+10$ to provide a wide range of aerothermodynamic loading reported in the AEDC wind tunnel. The grid was created using the meshing software LINK3D and consisted of 50.4 million cells, with wall spacing producing y+ values well below one. In addition, the wake region behind the waverider was excluded,

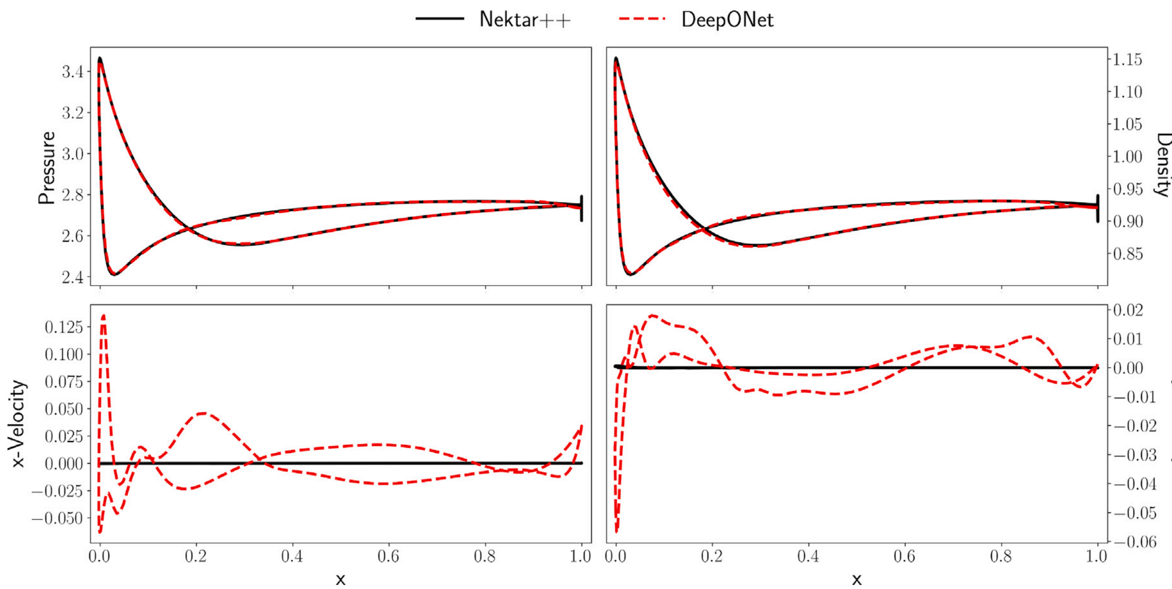


Fig. 8. Plot of the flowfield variables on the surface of the test set airfoil NACA **7315**. All plots display accurate predictions on the surface, which are then used to compute the lift and drag forces. The no-slip condition is not directly enforced by the DeepONet, which results in the velocity plot difference. However, it can be seen by the y-scale that the prediction is close to zero, aside from the leading edge, and does not greatly affect the overall lift and drag computation.

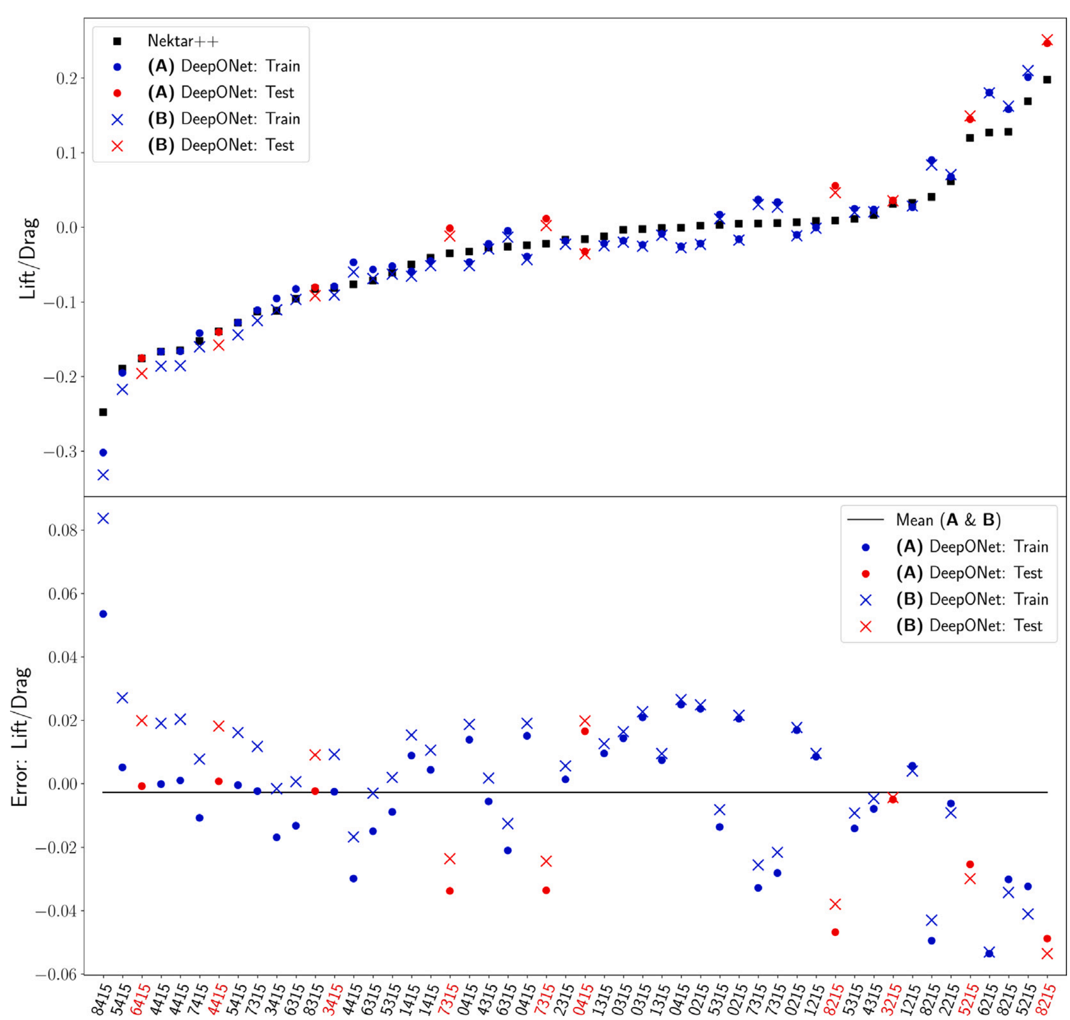


Fig. 9. Plot of the computed lift-to-drag objective for the entire dataset sorted by the Nektar++ reference values. As seen in both plots, the approximation to the high-fidelity CFD solution is very accurate and consistent throughout the entire parametric domain. In particular, we note that the testing set performs comparably to the training set, meaning there is little to no generalization error, which is necessary when inferring unseen queried geometries during optimization.

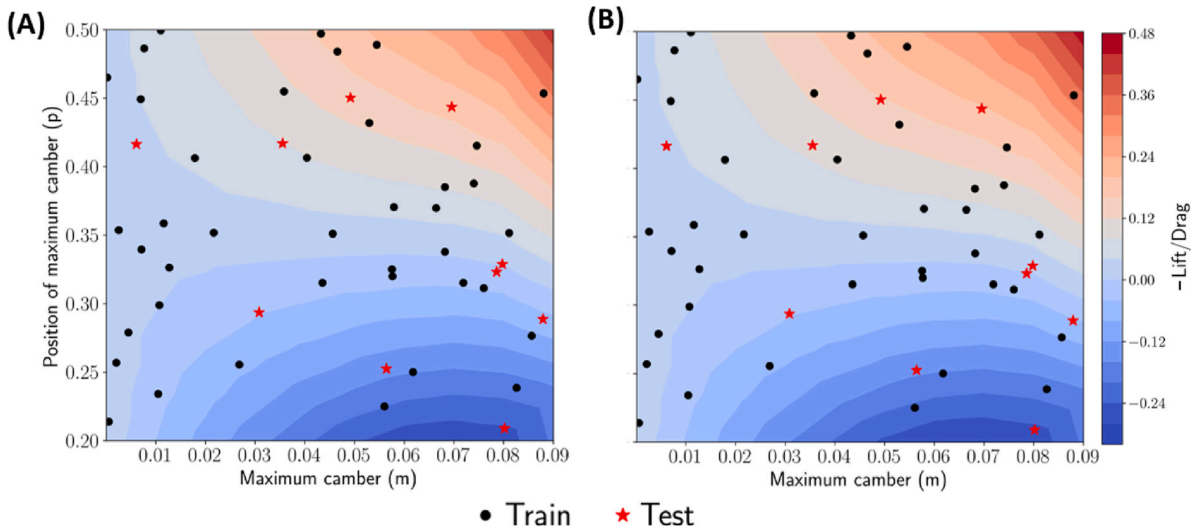


Fig. 10. Visualization of the lift/drag landscape obtained from brute force sampling of p and m using a 10 × 10 grid. The train and test sets are also plotted to show the sparse dataset used by the DeepONet (A) Landscape obtained using finite difference approximation. (B) Landscape obtained using automatic differentiation approximation.

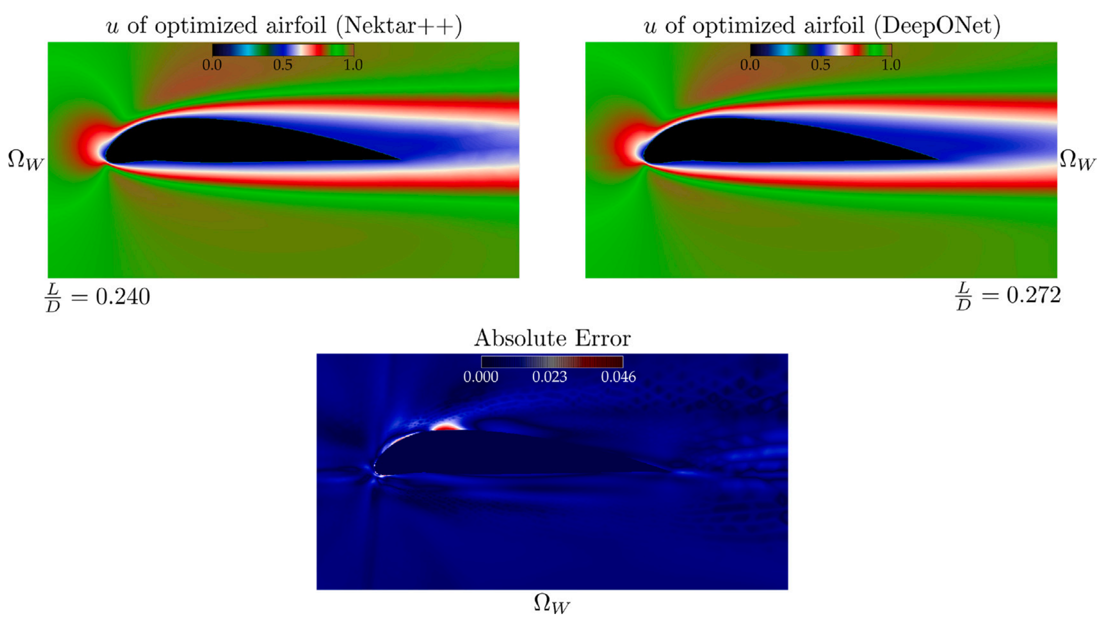


Fig. 11. Optimized airfoil. The u velocity field of the flow past the optimized airfoil with $(p, m) = (0.2, 0.067)$ in Ω_W. The velocity field simulated in Nektar++ and predicted by the DeepONet are shown here.

and the fluid domain ends at the rear of the vehicle. The simulations were run to 20+ flowthrough times to ensure that shock structures and boundary layers are well established and that the flow solution is stable.

Regarding surrogate neural operators, our literature survey suggests that DeepONet can serve as accurate surrogates for 3D or higher dimensional parametric PDEs. For example, in the paper Kontolati et al. (2023), a DeepONet is used to infer 3D atmospheric flows over the globe. In the paper by Meng et al. (2022), a DeepONet is used to solve the stochastic Darcy equation in 100 dimensions. Herein, we demonstrate the application of DeepONet for approximating shear stress (τ_y) and heat flux (Q_w) fields around an AEDC waverider whose geometry is provided in Fig. 13.

The dataset consists of τ_y and Q_w fields at the surface of the waverider geometry corresponding to 21 angles of attack varying from −10° to 10° with an increment of 1°. Details regarding the partitioning of the dataset into training and testing categories and the corresponding relative L^2 errors are reported in Table 4 and visualized in Fig. 14.

In Figs. 15 and 16, subfigure (a) represents the predicted heat fluxes on the top and bottom surfaces of the waverider, and subfigure (b) represents the true and the DeepONet predicted heat flux profiles along the top and bottom centerlines respectively. For the top and the bottom centerline profiles, we observe relative L^2 errors of 5.23% and 3.00%, respectively.

Next, we compare Q_w and τ_y fields simulated by the US3D solver with the surrogate 3D DeepONet's predictions, across the entire surface of the waverider. In Fig. 17, we present the flux and shear stress fields at the surface of the AEDC waverider at 2° angle of attack. The right column represents a zoomed-in view of the leading edge of the waverider to better visualize the quality of the surrogate 3D DeepONet prediction at the region where the variance of the fields is the largest. We observe that the maximum absolute errors for heat flux and shear stress are 9.7% and 4.1% respectively, for this test sample.

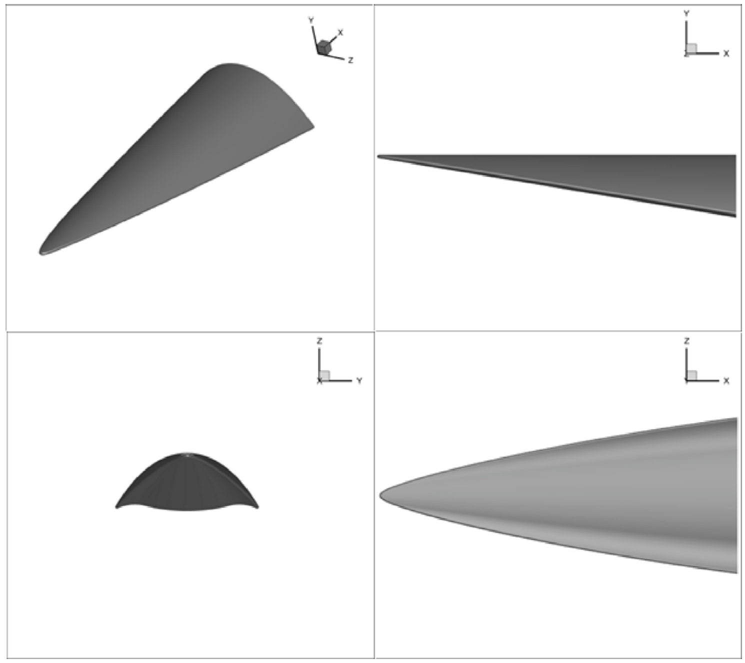


Fig. 12. Different views of the AEDC Waverider geometry.

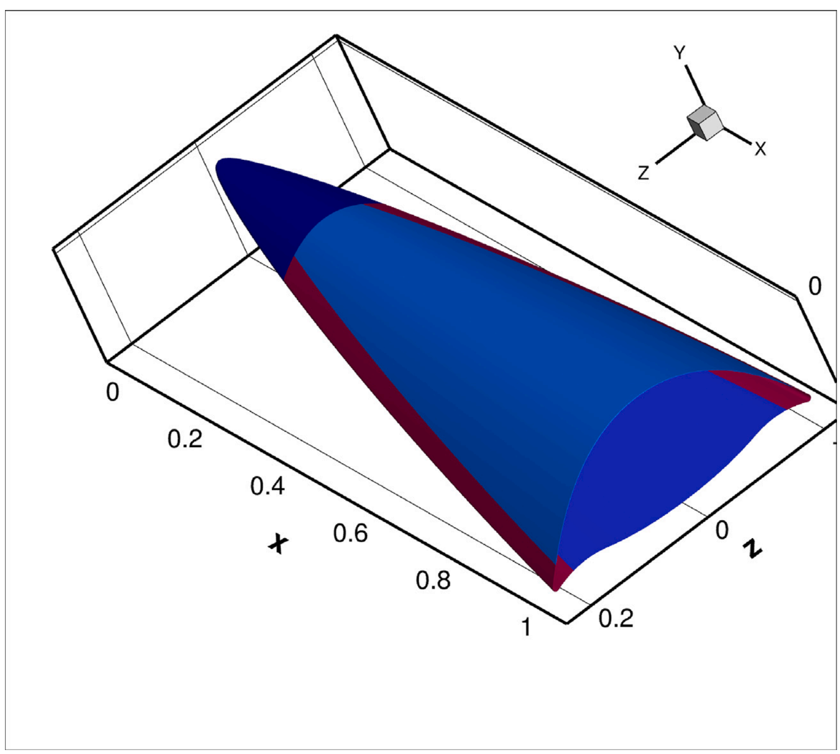


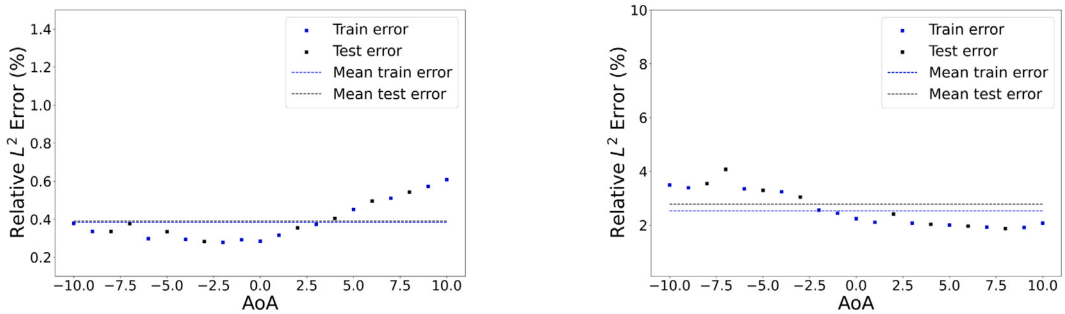
Fig. 13. The AEDC waverider geometry is used to compute the flowfields at 21 angles of attack. Diverse colors were assigned to distinct quadrants, aiding in mesh optimization and refinement, specifically targeting precise representation of regions with significant curvature.

6. Computational complexity of the DeepONet

DeepONets exhibit quick and cost-effective inference but require pre-training. The cost of the training process is comprised of two components: the first is related to dataset creation, involving the computation of numerous numerical solutions of the compressible Navier–Stokes equation for different geometrical parameters. The second cost element pertains to the gradient descent-based training procedure itself. To understand these distinct costs, Di Leoni et al. (2023) have introduced a set of three metrics that focus on tackling the computational complexity. These metrics are outlined as follows:

$$R_t = \frac{C_t}{N_s C_s}, \quad R_e = \frac{C_e}{C_s}, \quad N_e^* = N_s + \frac{C_t}{C_s},$$

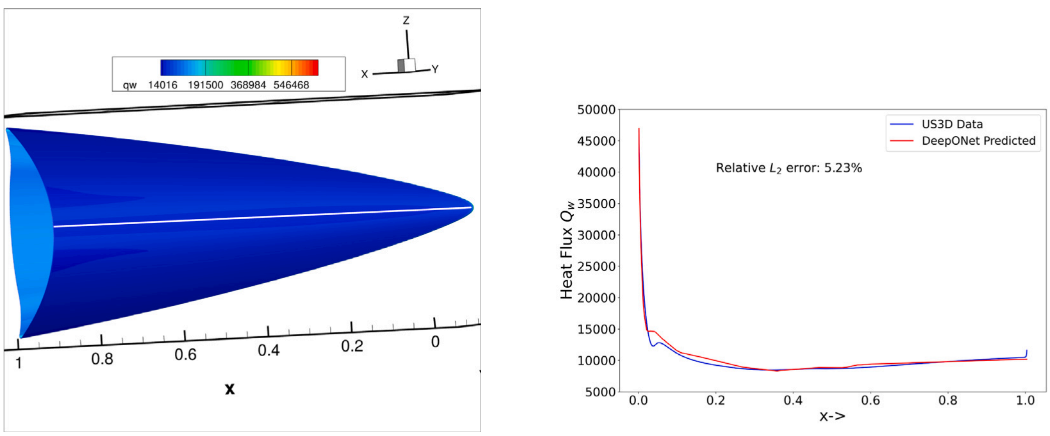
where R_t, R_e, and N_e^* are the training ratio, evaluation ratio, and break-even number, respectively. C_t is the cost in time of training the DeepONet, N_s is the number of simulations needed to generate the



(a) Training and testing error in τ_y for the AEDC waverider

(b) Training and testing error in Q_w for the AEDC waverider

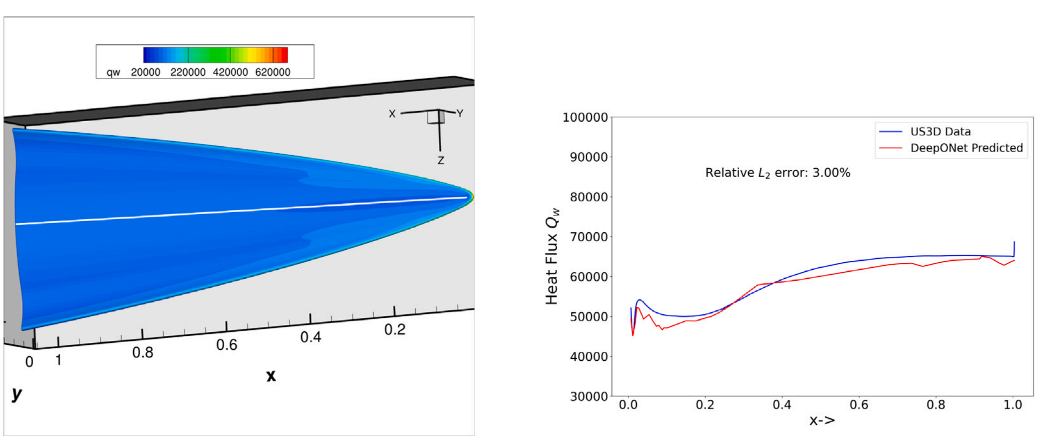
Fig. 14. The L^2 norm of relative error for training and testing samples in (a) Total shear stress τ_y and heat flux Q_w. Train and test mean errors for τ_y and Q_w are $(0.38\%, 0.39\%)$ and (2.53%, 2.78%), respectively.



(a) Heat flux (Q_w) profile on top surface of the AEDC waverider

(b) A comparison heat flux (Q_w) value at top surface of waverider predicted from US3D and DeepONet

Fig. 15. In subfigure (a), a comparison is shown in the calculated heat flux (Q_w) at an angle of attack (AoA) of 2° (test case) along the bottom surface profile of the waverider, indicated by a white line. Subfigure (b) presents a comparison between the heat flux values (Q_w) obtained using the US3D solver and those predicted by the DeepONet model. The L^2 relative error between actual and predicted value is 5.23%.



(a) Heat flux (Q_w) profile on bottom surface of the AEDC waverider

(b) A comparison heat flux (Q_w) value predicted from US3D and DeepONet

Fig. 16. In subfigure (a), a comparison is shown in the calculated heat flux (Q_w) at an angle of attack (AoA) of 2° (test case) along the lower surface profile of the waverider, indicated by a white line. Subfigure (b) presents a comparison between the heat flux values (Q_w) obtained using the US3D solver and those predicted by the DeepONet model. The L^2 relative error between actual and predicted value is 3.00%.

dataset, C_s is the cost in time of running each simulation, and C_e is the cost in time of evaluating a DeepONet.

The most important number here is the break-even number, which signifies the count of evaluations at which the DeepONet starts to offer

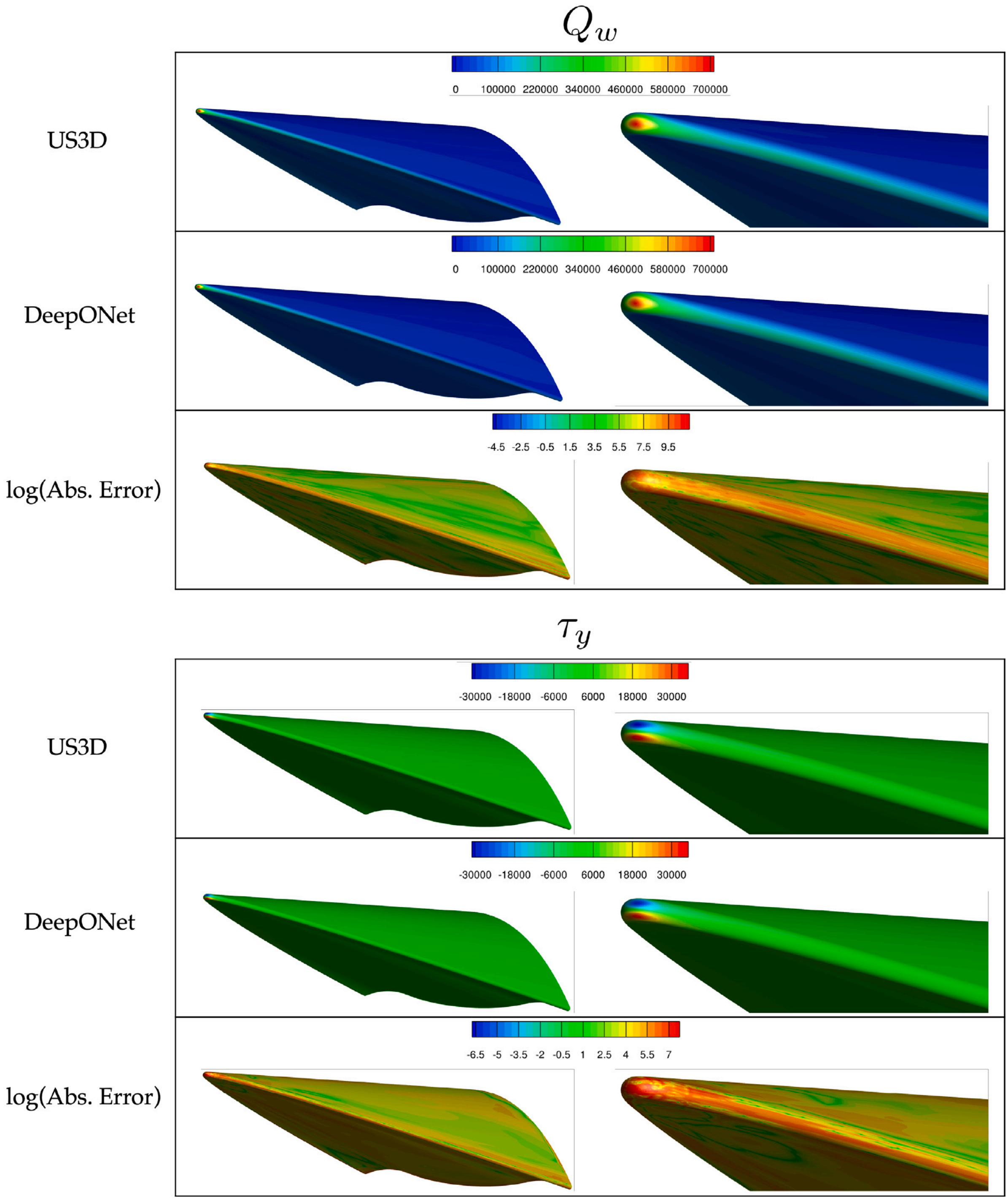


Fig. 17. Heat flux (Q_w) (upper panel) and total shear stress (τ_y) (lower panel) distributions on the surface of the AEDC waverider. We compare the fields simulated by the US3D solver against those predicted by the surrogate DeepONet for an unseen 2° angle of attack.

advantages over the numerical solver. This analysis is based on the total cost ratio and helps determine this threshold. Therefore, the total computational complexity for DeepONet is expressed as Di Leoni et al. (2023) $R_c = \frac{N_s C_s + C_t + N_e C_e}{N_e C_s}$.

Specifically, we will use the AEDC waverider as an example to illustrate the aforementioned values. Here $C_t = 8$ Hrs on single A100, 80 GB GPU, $C_s = 20.5 \times 10^3$ Core-hours, $N_s = 21$, $C_e = 0.001$ s. Therefore $R_t = 1.86 \times 10^{-5}$, $R_e = 4.88 \times 10^{-8}$ and $N_e^* = 21.00039$. These numbers indicate that the training duration of a DeepONet is both controllable and significantly reduced compared to the data generation stage in the current scenario. To summarize: if we require more than $N_e^* \geq 21$ simulations then it is computationally more efficient to train and deploy surrogate DeepONet but at the cost of lower accuracy in comparison to the numerical solver. However, it is crucial to note that these estimates could significantly differ based on the specific application, smoothness, and regularity of the solutions.

7. Conclusions

We have successfully integrated DeepONets as a surrogate model into the shape optimization framework for airfoils. Having summarized prior work in this field, we empirically demonstrate the efficacy of

Table 4
The L^2 norm of relative error between actual and predicted total shear stress τ_y and heat flux Q_w from DeepoNet for the AEDC waverider.

AoA	Sample type	L^2 error in τ_y	L^2 error in Q_w
−10	Train	0.38%	3.50%
−9	Train	0.34%	3.39%
−8	Test	0.34%	3.55%
−7	Test	0.38%	4.07%
−6	Train	0.30%	3.35%
−5	Test	0.33%	3.30%
−4	Train	0.29%	3.24%
−3	Test	0.28%	3.05%
−2	Train	0.28%	2.57%
−1	Train	0.29%	2.44%
0	Train	0.28%	2.24%
1	Train	0.32%	2.11%
2	Test	0.36%	2.42%
3	Train	0.37%	2.08%
4	Test	0.40%	2.04%
5	Train	0.45%	2.01%
6	Test	0.50%	1.97%
7	Train	0.51%	1.93%
8	Test	0.54%	1.87%
9	Train	0.57%	1.91%
10	Train	0.61%	2.08%

DeepONets in terms of retaining sufficient flowfield accuracy used in evaluating the objective function of lift-to-drag, as well as the significant computational speed up as a replacement for a traditional CFD solver during online constrained geometry optimization. We have provided thorough validation of the results presented as well as extensive experimentation such as two approaches (A) and (B) when approximating the wall shear stress and two forms of DeepONet inputs (NURBS and ξ_g) to ensure a robust pipeline. The NURBS-DeepONet enables the surrogate model to accurately predict the flow around arbitrary geometries and not only the NACA 4-series geometries considered in this study. Importantly, DeepONets exhibit almost no generalization error over the dataset, so it follows that the resulting optimized geometry ($p = 0.2, m = 0.067$) is accurate and achieved 32,253 speed-up compared to the CFD baseline. This behavior is expected because, as a data-driven model, the DeepONet is capable of accurately predicting the flow fields around an unseen geometry sampled from the same $p \times m$ space used during training. The error in prediction may increase when airfoil geometries are sampled from a different distribution. Ideally, if we were to have a larger training dataset comprising samples that span a broader distribution, the proposed framework would perform well in practical applications. However, generating such a training dataset can be computationally expensive. Therefore, incorporating the physics (Raissi et al., 2019) while training the surrogate DeepONet can be a way to make the framework robust to out-of-distribution cases. Nevertheless, the computational complexity pertaining to training a DeepONet/Physics-Informed DeepONet can be alleviated by easily extending the training routines across multiple GPUs across multiple nodes in a data-parallel (Goyal et al., 2017) sense. The framework is general and can address more complex problems with multiple inputs, e.g. different Mach numbers and different angles of attack that can be input to either the branch or the trunk networks. Hence, with relatively small modifications, such a framework can handle optimization in the high-speed flow regimes that exhibit flow unsteadiness, shocks, non-equilibrium chemistry, and even morphing geometry. Furthermore, the approaches presented are flexible due to the integration of machine learning in the form of function-to-function maps using DeepONet. Therefore, improvements such as the introduction of multi-fidelity training and physics-informed machine learning can be leveraged to reduce the cost of data generation. We also successfully show the application of automatic differentiation, which performs comparably to the traditional approach of finite differences in the wall shear stress calculation. We also validated the ability of DeepONet to handle a complex 3D geometry under challenging hypersonic conditions. This experiment showcases the ability of our framework to be able to translate to more challenging shape optimization problems in the future. Finally, we hope to utilize transfer learning and uncertainty quantification using the recently developed library *NeuralUQ* (Zou et al., 2022) to extrapolate outside of the trained geometric parameter domain to find global optima with confidence.

CRediT authorship contribution statement

Khemraj Shukla: Methodology, Software, Validation, Data curation, Visualization, Writing – original draft. **Vivek Oommen:** Methodology, Software, Validation, Data curation, Visualization, Writing – original draft. **Ahmad Peyvan:** Methodology, Software, Validation, Data curation, Visualization, Writing – original draft. **Michael Penwarden:** Methodology, Software, Validation, Data curation, Visualization, Writing – original draft. **Nicholas Plewacki:** Data curation, Formal analysis. **Luis Bravo:** Supervision, Conceptualization, Resources, Funding acquisition, Writing – review & editing. **Anindya Ghoshal:** Supervision, Conceptualization, Resources, Funding acquisition, Writing – review & editing. **Robert M. Kirby:** Supervision, Conceptualization, Resources, Funding acquisition, Writing – review & editing. **George Em Karniadakis:** Supervision, Conceptualization, Resources, Funding acquisition, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

The research reported in this document is performed in connection with cooperative agreement contract/instrument W911NF-22-2-0047 with the U.S. Army Research Laboratory. L.B., A.G., and N.P. were supported by the US Army Research Laboratory 6.1 basic research program in vehicle power and propulsion sciences. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies or positions, either expressed or implied, of the U.S. Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

Appendix

A.1. Nektar++ cross-verification

We selected the flow parameters as $M = 0.5$ and $Re = 500$ to ensure a steady-state solution. Calculating the steady-state solution requires careful consideration. To ensure the steady state solution, for each NACA profile, we recorded the value of conservative variables at six different locations in the wake. We then examined the conservative variables' time history to ensure the steady state is reached and the solution update has stopped. Fig. 18 shows the location of the history points, where the transient flow could last longer than other spatial locations. As a sample, we plot the time history of variables at point 5 (Fig. 19), which experiences the highest flow fluctuations in time. According to Fig. 19, the solution reached a

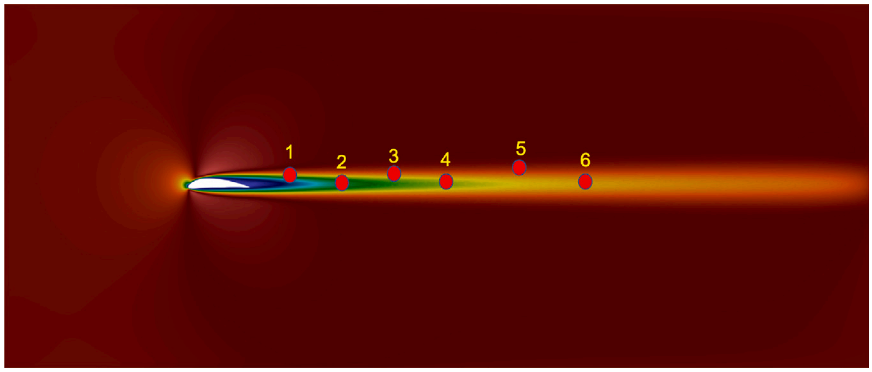


Fig. 18. History points locations in the wake for NACA airfoils. The conservative variable values are stored in time at these locations to monitor the steady-state solution.

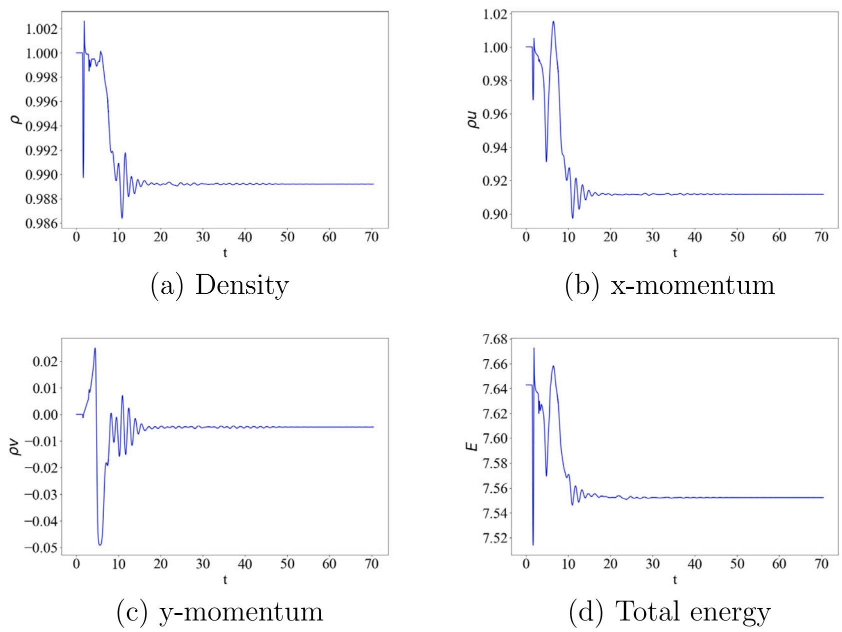


Fig. 19. Time history of (a) ρ, (b) ρu, (c) ρv, and (d) E at point 5 (see Fig. 18) for the NACA7315 airfoil. The profiles show that the simulation has reached a steady-state solution where the flow variables reach constant states.

stationary state when the conservative variables approached constant values.

We also validated the simulation setup of NACA airfoils in Nektar++ with the results obtained by an in-house code based on the discontinuous spectral element method of Kopriva and Kolias (1996), Peyvan et al. (2021). We selected the NACA0020 airfoil for cross-verification. The steady-state solutions of the flow with $M = 0.5$ and $Re = 500$ are computed, and the results are shown in Fig. 20. The flow field primitive variables computed by both solvers agree and show the validity of the Nektar++ simulation setup, including mesh and simulation parameters. According to Fig. 20, the number of elements employed for the NekTar++ simulations is sufficient for an accurate prediction. After validating the flow field, we performed an extra flow simulation around the NACA4402 airfoil. The drag and lift coefficients of this airfoil are reported by Kunz (2003) but for an incompressible flow at $Re = 1000$.

We employed the automatic mesh generation setup used for the training set to create the mesh and used the same simulation setup as the training set. We computed the drag and lift coefficients and compared them with the literature. Table 5 compares the drag and lift coefficients computed by Nektar++ with values reported by Kunz (2003) for a similar but not exactly the same setup.

Table 5
Comparison between the drag and lift coefficients computed by Nektar++ with Kunz (2003). For the reference solution, the setup is different, i.e., the Reynolds number is 1000, and the flow is assumed to be incompressible.

	C_d	C_l	Error
Nektar++	0.1050	0.1700	1.7%
Kunz (2003)	0.1032	0.1852	8.2%

A.2. DeepONet hyperparameter optimization experiments

We perform an experiment on the neural network architecture of the DeepONet model. For this study, we use the Parameter DeepONet, which approximates the density field around the airfoil. We train different DeepONet models with the number of hidden layers in branch and trunk networks as d = 2, 4, 6. We also vary the width of the network, that is, the number of neurons in each hidden layer as w = 20, 50, 100.

From Table 6, we observe that the prediction error shows a decreasing trend with respect to the width of the networks. The prediction error saturates near a depth of 4. Nevertheless, hyperparameter optimization with respect to the width and depth of the networks does yield an optimal architecture that achieves an order of magnitude improvement in terms of the relative L^2 error.

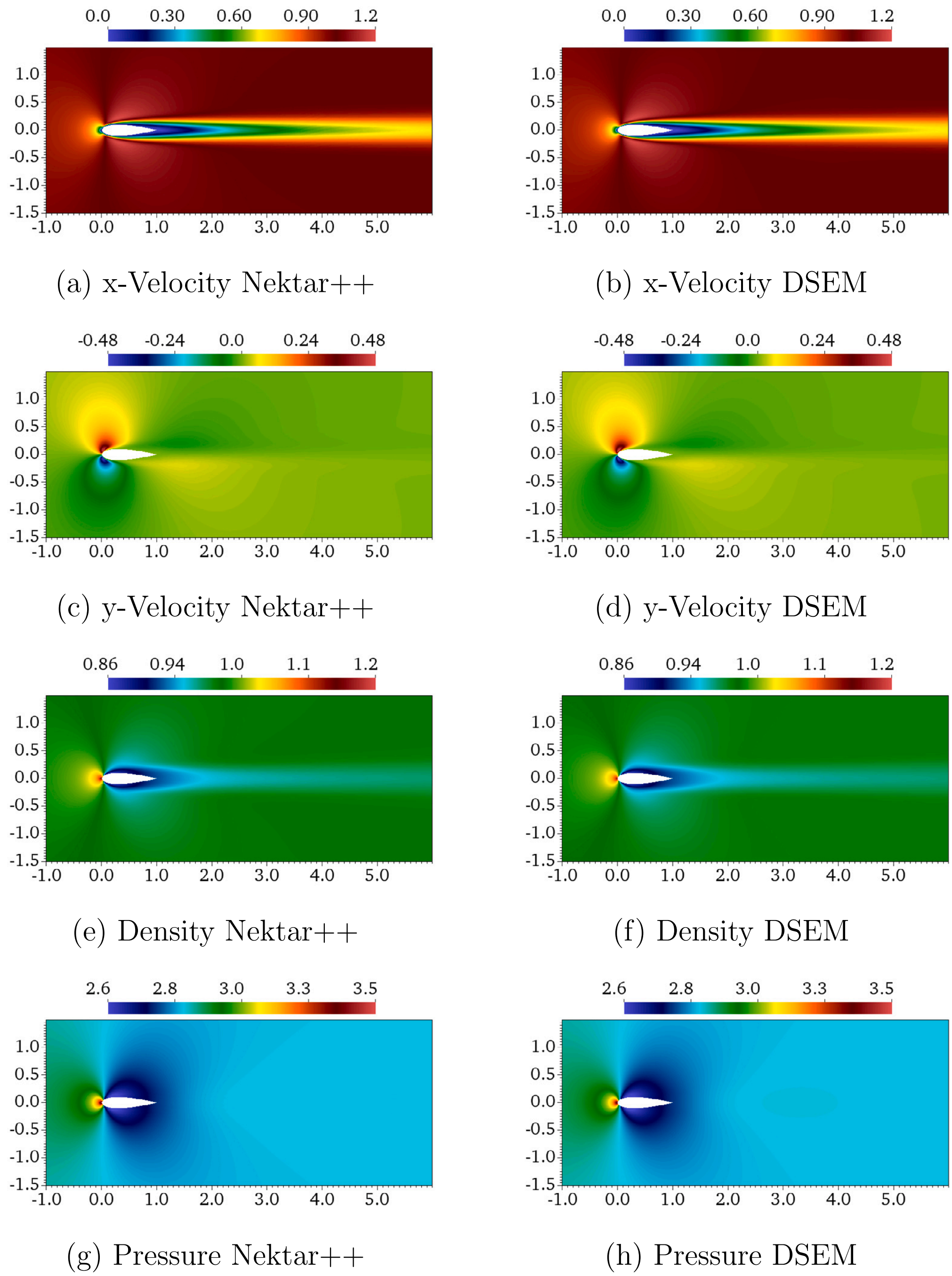


Fig. 20. Cross verification of steady state flow around NACA0020 obtained by the DSEM code and Nektar++. The flow parameters are set as $Re = 500$ and $M = 0.5$.

Table 6
Rel. L^2 norm of ρ.

	w = 20	w = 50	w = 100
d = 2	9.94E−03	6.13E−03	5.18E−03
d = 4	4.97E−03	2.88E−03	2.74E−03
d = 6	4.41E−03	3.49E−03	3.55E−03

Next, we perform an experiment to investigate the convergence of the model by changing the learning rate. In this study, we consider Adam optimizer with learning rates = 10^{-2}, 10^{-3} and 10^{-4}.

The DeepONet model that approximates the density field is trained for 200000 epochs. From Fig. 21, we observe that learning rate = 10^{-2} and 10^{-3} are too large for the approximation task, causing the Mean Squared Error computed between the true and predicted density fields to oscillate without converging. The results suggest that a lower learning rate could be a better choice to minimize oscillations.

A.3. Geometry optimization validation

To validate the main geometry optimization findings in the manuscript using Dakota, we also evaluate the objective function using brute force and SciPy's (Virtanen et al., 2020) dual-annealing method. Brute force is evaluated using a 10×10 grid on the geometric parameter space $(p, m) \in [0.2, 0.5] \times [0.0, 0.09]$, and therefore requires 100 evaluations of the objective. The dual-annealing method is set to have a maximum amount of 50 evaluations but likely could be set to fewer. As seen in Table 7, all methods discover the same optimal set

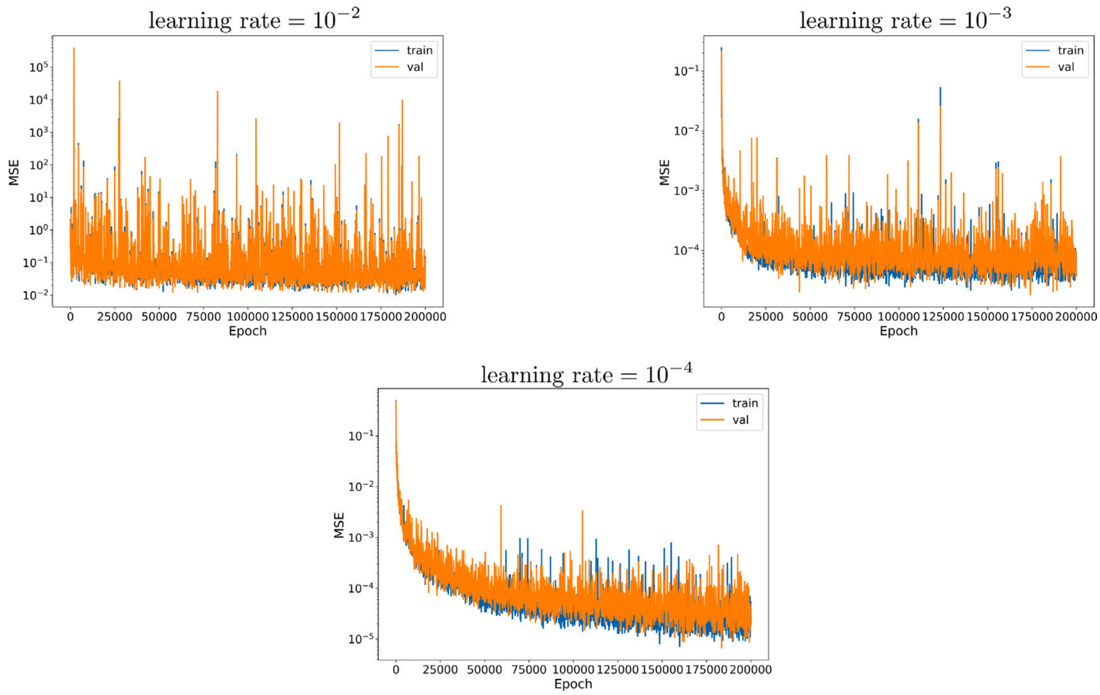


Fig. 21. The error convergence plot of DeepONet models with learning rates = 10^{-2}, 10^{-3} and 10^{-4}.

Table 7
Geometry optimization results for different DeepONet methodologies. (A) Flowfield mapping with finite-difference approximation. (B) Flowfield mapping with automatic-differentiation approximation and DAKOTA framework uses a gradient-free approach for the optimization. (**C**) and (**D**) uses local and global gradient based optimization algorithm for shape optimization. The cost of these methods was obtained on an eight-core CPU (2.3 GHz Intel core i9) with 16 GB 2667 MHz DDR4 of MacBook Pro 2019.

Model type	Optimized parameters (p,m)	Optimized $\frac{L}{D}$	Computation cost (s)
Brute Force (**A**)	(0.2, 0.070)	0.272	332.63
Brute Force (**B**)	(0.2, 0.070)	0.281	264.12
SciPy dual-annealing (**A**)	(0.2, 0.064)	0.269	196.09
SciPy dual-annealing (**B**)	(0.2, 0.064)	0.281	165.57
Dakota (**A**)	(0.2, 0.067)	0.269	157.71
Dakota (**B**)	(0.2, 0.063)	0.282	105.00
Dakota (**C**)	(0.2, 0.068)	0.269	296.81
Dakota (**D**)	(0.2, 0.068)	0.269	1278.6

of parameters and $\frac{L}{D}$, regardless of optimizer or partial approximation given by (A) and (B).

In the Table 7, we have also shown the results for optimization using local and global gradient-based methods adopted from the DAKOTA framework. For local gradient-based optimization, we utilize the Fletcher–Reeves conjugate gradient algorithm (Vanderplaats, 1973). This method can effectively utilize the bound constraint provided on (m, p). However, for the global gradient method, we use the multistart strategy with the method of feasible directions (MFD) (Chen and Kostreva, 2000). This is to be noted that gradient-based optimizers are best suited for efficient convergence to a local minimum in the vicinity of the initial point and are not intended to find global optima in nonconvex design spaces. Therefore, gradient-based methods are suitable to offer the best convergence rates, of all of the local optimization methods, and are chosen when the cost function is smooth, unimodal, and well-behaved. However, gradient based methods are not the first choice when the underlying problem exhibits non-smooth, discontinuous, or multi-modal. In such cases, the derivative-free methods are more appropriate and therefore chosen at first place for the present work.

A.4. *WSS from automatic-differentiation*

The derivation for Eq. (11), which defines the term $\frac{dU}{d\vec{n}}$ used in the wall shear stress (WSS) calculation, is provided here. The expression is a function of partials (u_x, u_y, v_x, v_y), which are readily available using AD and θ, which is the angle between the x-y axis and the normal-tangent axis of each airfoil segment. Let us write the unit normal and unit tangent as

$$\vec{n} = -\sin(\theta)\hat{i} + \cos(\theta)\hat{j} \tag{13}$$

$$\vec{t} = \cos(\theta)\hat{i} + \sin(\theta)\hat{j} \tag{14}$$

To get U, which is the directional flow speed relative to the surface of the airfoil section, we take the dot product of the velocity vector and the unit tangent

$$U = (u\hat{i} + v\hat{j}) \cdot (\cos(\theta)\hat{i} + \sin(\theta)\hat{j}) = u\cos(\theta) + v\sin(\theta) \tag{15}$$

Finally, we take the derivative of U with respect to the unit normal $\vec{n}$

$$\begin{aligned} \frac{dU}{d\vec{n}} &= \vec{\nabla} U . \vec{n} = -u_x \sin(\theta)\cos(\theta) - v_x \sin^2(\theta) + u_y \cos^2(\theta) + v_y \sin(\theta)\cos(\theta) \\ &\Rightarrow (v_y - u_x)\sin\theta\cos\theta - v_x \sin^2\theta + u_y \cos^2\theta \end{aligned} \tag{16}$$

and recover the expression in Eq. (11) of the manuscript.

References

Adams, B., Bohnhoff, W., Dalbey, K., Ebeida, M., Eddy, J., Eldred, M., Hooper, R., Hough, P., Hu, K., Jakeman, J., Khalil, M., Maupin, K., Monschke, J.A., Ridgway, E., Rushdi, A., Seidl, D., Stephens, J., Swiler, L.P., Tran, A., Winokur, J., 2022. Dakota, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis: Version 6.16 user's manual.. http://dx.doi.org/10.2172/1868142, URL https://www.osti.gov/biblio/1868142.

Akram, M.T., Kim, M.-H., 2021. CFD analysis and shape optimization of airfoils using class shape transformation and genetic algorithm—Part I. Appl. Sci. 11 (9), 3791.

Amsallem, D., Zahr, M., Choi, Y., Farhat, C., 2015. Design optimization using hyper-reduced-order models. Struct. Multidiscip. Optim. 51 (4), 919–940.

Anosri, S., Panagant, N., Champasak, P., Bureerat, S., Thipyopas, C., Kumar, S., Pholdee, N., Yıldız, B.S., Yildiz, A.R., 2023. A comparative study of state-of-the-art metaheuristics for solving many-objective optimization problems of fixed wing unmanned aerial vehicle conceptual design. Arch. Comput. Methods Eng. 1–15.
Aye, C.M., Pholdee, N., Yildiz, A.R., Bureerat, S., Sait, S.M., 2019. Multi-surrogate-assisted metaheuristics for crashworthiness optimisation. Int. J. Veh. Design 80 (2–4), 223–240.
Baydin, A.G., Pearlmutter, B.A., Radul, A.A., Siskind, J.M., 2018. Automatic differentiation in machine learning: a survey. J. March. Learn. Res. 18, 1–43.
Benner, P., Gugercin, S., Willcox, K., 2015. A survey of projection-based model reduction methods for parametric dynamical systems. SIAM Rev. 57 (4), 483–531.
Benner, P., Ohlberger, M., Patera, A., Rozza, G., Urban, K., 2017. Model reduction of parametrized systems. Springer.
Bingol, O.R., Krishnamurthy, A., 2019. NURBS-Python: An open-source object-oriented NURBS modeling framework in Python. SoftwareX 9, 85–94. http://dx.doi.org/10.1016/j.softx.2018.12.005.
Bui-Thanh, T., Willcox, K., Ghattas, O., 2008. Model reduction for large-scale systems with high-dimensional parametric input space. SIAM J. Sci. Comput. 30 (6), 3270–3288.
Candler, G., Johnson, H., Nompelis, I., Gidzak, V., Subbareddy, P., Barnhardt, M., 2015. Development of the US3D code for advanced compressible and reacting flow simulations. In: 53rd AIAA Aerospace Sciences Meeting. http://dx.doi.org/10.2514/6.2015-1893.
Cantwell, C., Moxey, D., Comerford, A., Bolis, A., Rocco, G., Mengaldo, G., De Grazia, D., Yakovlev, S., Lombard, J.-E., Ekelschot, D., Jordi, B., Xu, H., Mohamied, Y., Eskilsson, C., Nelson, B., Vos, P., Biotto, C., Kirby, R., Sherwin, S., 2015. Nektar++: An open-source spectral/hp element framework. Comput. Phys. Comm. 192, 205–219. http://dx.doi.org/10.1016/j.cpc.2015.02.008, URL https://www.sciencedirect.com/science/article/pii/S0010465515000533.
Carlberg, K., Farhat, C., 2008. A compact proper orthogonal decomposition basis for optimization-oriented reduced-order models. In: 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference. p. 5964.
Carpentieri, G., Koren, B., van Tooren, M.J., 2007. Adjoint-based aerodynamic shape optimization on unstructured meshes. J. Comput. Phys. 224 (1), 267–287.
Chen, T., Chen, H., 1995. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. IEEE Trans. Neural Netw. 6 (4), 911–917.
Chen, G., Fidkowski, K., 2017. Airfoil shape optimization using output-based adapted meshes. In: 23rd AIAA Computational Fluid Dynamics Conference. p. 3102.
Chen, X., Kostreva, M.M., 2000. Methods of feasible directions: A review. Progress in Optimization: Contributions from Australasia 205–219.
Chernukhin, O., Zingg, D.W., 2013. Multimodality and global optimization in aerodynamic design. AIAA J. 51 (6), 1342–1354.
Chiavazzo, E., Gear, C.W., Dsilva, C.J., Rabin, N., Kevrekidis, I.G., 2014. Reduced models in chemical kinetics via nonlinear data-mining. Processes 2 (1), 112–140.
Choi, Y., Boncoraglio, G., Anderson, S., Amsallem, D., Farhat, C., 2020. Gradient-based constrained optimization using a database of linear reduced-order models. J. Comput. Phys. 423, 109787.
De, S., Hassanaly, M., Reynolds, M., King, R.N., Doostan, A., 2022. Bi-fidelity modeling of uncertain and partially unknown systems using DeepONets. http://dx.doi.org/10.48550/ARXIV.2204.00997, Preprint at https://arxiv.org/abs/2204.00997.
Di Leoni, P.C., Lu, L., Meneveau, C., Karniadakis, G.E., Zaki, T.A., 2023. Neural operator prediction of linear instability waves in high-speed boundary layers. J. Comput. Phys. 474, 111793.
Du, X., He, P., Martins, J.R., 2021. Rapid airfoil design optimization via neural networks-based parameterization and surrogate modeling. Aerosp. Sci. Technol. 113, 106701.
Eberhart, R., Kennedy, J., 1995. A new optimizer using particle swarm theory. In: MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science. Ieee, pp. 39–43.
Finkel, D., Kelley, C.T., 2004. Convergence analysis of the DIRECT algorithm. Tech. rep., North Carolina State University. Center for Research in Scientific Computation.
Geuzaine, C., Remacle, J.-F., 2020-06-22. Gmsh. URL http://http://gmsh.info/.
Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., He, K., 2017. Accurate, large minibatch sgd: Training imagenet in 1 hour. arXiv preprint arXiv:1706.02677.
Hao, Z., Wang, Z., Su, H., Ying, C., Dong, Y., Liu, S., Cheng, Z., Song, J., Zhu, J., 2023. GNOT: A general neural operator transformer for operator learning. In: Proceedings of the 40th International Conference on Machine Learning. In: Proceedings of Machine Learning Research, vol. 202, PMLR, pp. 12556–12569.
He, X., Li, J., Mader, C.A., Yildirim, A., Martins, J.R., 2019. Robust aerodynamic shape optimization—from a circle to an airfoil. Aerosp. Sci. Technol. 87, 48–61.
Hesthaven, J.S., Rozza, G., Stamm, B., et al., 2016. Certified Reduced Basis Methods for Parametrized Partial Differential Equations, Vol. 590. Springer.
Hesthaven, J.S., Ubbiali, S., 2018. Non-intrusive reduced order modeling of nonlinear problems using neural networks. J. Comput. Phys. 363, 55–78.
Hicks, R.M., Henne, P.A., 1978. Wing design by numerical optimization. J. Aircr. 15 (7), 407–412.
Howard, A.A., Perego, M., Karniadakis, G.E., Stinis, P., 2022. Multifidelity deep operator networks. http://dx.doi.org/10.48550/ARXIV.2204.09157, Preprint at https://arxiv.org/abs/2204.09157.
Jacobs, E.N., Ward, K.E., Pinkerton, R.M., 1933. The characteristics of 78 related airfoil sections from tests in the variable-density wind tunnel. Natl. Advis. Comm. Aeronaut..
Jin, P., Meng, S., Lu, L., 2022. MIONet: Learning multiple-input operators via tensor product. arXiv preprint arXiv:2202.06137.
Kammeyer, M., Gillum, M., 1994. Design validation tests on a realistic hypersonic waverider at mach 10, 14, and 16.5 in the naval surface warfare center hypervelocity wind tunnel no. 9. Naval Surface Warfare Center NSWCDD/TR-93/198.
Kontolati, K., Goswami, S., Karniadakis, G.E., Shields, M.D., 2023. Learning in latent spaces improves the predictive accuracy of deep neural operators. arXiv preprint arXiv:2304.07599.
Kontolati, K., Goswami, S., Shields, M.D., Karniadakis, G.E., 2022. On the influence of over-parameterization in manifold based surrogates and deep neural operators. arXiv preprint arXiv:2203.05071.
Kopriva, D.A., Kolias, J.H., 1996. A conservative staggered-grid Chebyshev multidomain method for compressible flows. J. Comput. Phys. 125 (1), 244–261.
Krige, D.G., 1951. A statistical approach to some basic mine valuation problems on the witwatersrand. J. South. Afr. Inst. Min. Metall. 52 (6), 119–139.
Kumar, S., Yildiz, B.S., Mehta, P., Panagant, N., Sait, S.M., Mirjalili, S., Yildiz, A.R., 2023. Chaotic marine predators algorithm for global optimization of real-world engineering problems. Knowl.-Based Syst. 261, 110192.
Kunz, P.J., 2003. Aerodynamics and Design for Ultra-Low Reynolds Number Flight. Stanford University.
Lepine, J., Guibault, F., Trepanier, J.-Y., Pepin, F., 2001. Optimized nonuniform rational B-spline geometrical representation for aerodynamic design of wings. AIAA J. 39 (11), 2033–2041.
Li, J., Cai, J., Qu, K., 2019. Surrogate-based aerodynamic shape optimization with the active subspace method. Struct. Multidiscip. Optim. 59 (2), 403–419.
Li, Z., Huang, D.Z., Liu, B., Anandkumar, A., 2022. Fourier neural operator with learned deformations for pdes on general geometries. arXiv preprint arXiv:2207.05209.
Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., Anandkumar, A., 2020. Neural operator: Graph kernel network for partial differential equations. arXiv:2003.03485.
Liao, P., Song, W., Du, P., Zhao, H., 2021. Multi-fidelity convolutional neural network surrogate model for aerodynamic optimization based on transfer learning. Phys. Fluids 33 (12), 127121.
Lieberman, C., Willcox, K., Ghattas, O., 2010. Parameter and state model reduction for large-scale statistical inverse problems. SIAM J. Sci. Comput. 32 (5), 2523–2542.
Liu, J., Song, W.-P., Han, Z.-H., Zhang, Y., 2017. Efficient aerodynamic shape optimization of transonic wings using a parallel infilling strategy and surrogate models. Struct. Multidiscip. Optim. 55 (3), 925–943.
Lu, L., Jin, P., Karniadakis, G.E., 2019. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. arXiv preprint arXiv:1910.03193.
Lu, L., Jin, P., Pang, G., Zhang, Z., Karniadakis, G.E., 2021. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. Nat. Mach. Intell. 3 (3), 218–229.
Lu, L., Meng, X., Cai, S., Mao, Z., Goswami, S., Zhang, Z., Karniadakis, G.E., 2022a. A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data. Comput. Methods Appl. Mech. Engrg. 393, 114778.
Lu, L., Pestourie, R., Johnson, S.G., Romano, G., 2022b. Multifidelity deep neural operators for efficient learning of partial differential equations with application to fast inverse design of nanoscale heat transport. Phys. Rev. Res. 4, 023210.
Mark, d.B., Otfried, C., Marc, v.K., Mark, O., 2008. Computational geometry algorithms and applications. Spinger.
Meng, Z., Qian, Q., Xu, M., Yu, B., Yıldız, A.R., Mirjalili, S., 2023a. PINN-FORM: A new physics-informed neural network for reliability analysis with partial differential equation. Comput. Methods Appl. Mech. Engrg. 414, 116172.
Meng, X., Yang, L., Mao, Z., del Águila Ferrandis, J., Karniadakis, G.E., 2022. Learning functional priors and posteriors from data and physics. J. Comput. Phys. 457, 111073.
Meng, Z., Yıldız, B.S., Li, G., Zhong, C., Mirjalili, S., Yildiz, A.R., 2023b. Application of state-of-the-art multiobjective metaheuristic algorithms in reliability-based design optimization: a comparative study. Struct. Multidiscip. Optim. 66 (8), 191.
Mengaldo, G., De Grazia, D., Witherden, F., Farrington, A., Vincent, P., Sherwin, S., Peiro, J., 2014. A guide to the implementation of boundary conditions in compact high-order methods for compressible aerodynamics. In: 7th AIAA Theoretical Fluid Mechanics Conference. p. 2923.
Mishra, R., Choudhary, A., Fatima, S., Mohanty, A., Panigrahi, B., 2022a. A fault diagnosis approach based on 2D-vibration imaging for bearing faults. J. Vib. Eng. Technol. 1–14.
Mishra, R.K., Choudhary, A., Fatima, S., Mohanty, A.R., Panigrahi, B.K., 2022b. A self-adaptive multiple-fault diagnosis system for rolling element bearings. Meas. Sci. Technol. 33 (12), 125018.
Mishra, R.K., Choudhary, A., Mohanty, A., Fatima, S., 2022c. An intelligent bearing fault diagnosis based on hybrid signal processing and henry gas solubility optimization. Proc. Inst. Mech. Eng. C 236 (19), 10378–10391.

Moxey, D., Cantwell, C.D., Bao, Y., Cassinelli, A., Castiglioni, G., Chun, S., Juda, E., Kazemi, E., Lackhove, K., Marcon, J., Mengaldo, G., Serson, D., Turner, M., Xu, H., Peiró, J., Kirby, R.M., Sherwin, S.J., 2020. Nektar++: Enhancing the capability and application of high-fidelity spectral/hp element methods. Comput. Phys. Comm. 249, 107110. http://dx.doi.org/10.1016/j.cpc.2019.107110, URL https://www.sciencedirect.com/science/article/pii/S0010465519304175.

Nadarajah, S., Jameson, A., 2001. Studies of the continuous and discrete adjoint approaches to viscous automatic aerodynamic shape optimization. In: 15th AIAA Computational Fluid Dynamics Conference. p. 2530.

Painchaud-Ouellet, S., Tribes, C., Trépanier, J.-Y., Pelletier, D., 2006. Airfoil shape optimization using a nonuniform rational b-splines parametrization under thickness constraint. AIAA J. 44 (10), 2170–2178.

Papadimitriou, D.I., Papadimitriou, C., 2016. Aerodynamic shape optimization for minimum robust drag and lift reliability constraint. Aerosp. Sci. Technol. 55, 24–33.

Peyvan, A., Komperda, J., Li, D., Ghiasi, Z., Mashayek, F., 2021. Flux reconstruction using Jacobi correction functions in discontinuous spectral element method. J. Comput. Phys. 435, 110261.

Raissi, M., Perdikaris, P., Karniadakis, G.E., 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J. Comput. Phys. 378, 686–707.

Renganathan, S.A., Maulik, R., Ahuja, J., 2021. Enhanced data efficiency using deep neural networks and Gaussian processes for aerodynamic design optimization. Aerosp. Sci. Technol. 111, 106522.

Reuther, J., Jameson, A., Farmer, J., Martinelli, L., Saunders, D., 1996. Aerodynamic shape optimization of complex aircraft configurations via an adjoint formulation. In: 34th Aerospace Sciences Meeting and Exhibit. p. 94.

Serrano, L., Vittaut, J.-N., et al., 2023. Operator learning on free-form geometries. In: ICLR 2023 Workshop on Physics for Machine Learning.

Srinath, D., Mittal, S., 2010. An adjoint method for shape optimization in unsteady viscous flows. J. Comput. Phys. 229 (6), 1994–2008.

Tao, J., Sun, G., 2019. Application of deep learning based multi-fidelity surrogate model to robust aerodynamic design optimization. Aerosp. Sci. Technol. 92, 722–737.

Vanderplaats, G.N., 1973. CONMIN: A FORTRAN program for constrained function minimization: User's manual. Tech. rep.

Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S.J., Brett, M., Wilson, J., Millman, K.J., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C.J., Polat, İ., Feng, Y., Moore, E.W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A.H., Pedregosa, F., van Mulbregt, P., SciPy 1.0 Contributors, 2020. SciPy 1.0: Fundamental algorithms for scientific computing in python. Nature Methods 17, 261–272. http://dx.doi.org/10.1038/s41592-019-0686-2.

Wang, K., Yu, S., Wang, Z., Feng, R., Liu, T., 2019. Adjoint-based airfoil optimization with adaptive isogeometric discontinuous Galerkin method. Comput. Methods Appl. Mech. Engrg. 344, 602–625.

Williams, M.O., Kevrekidis, I.G., Rowley, C.W., 2015. A data–driven approximation of the koopman operator: Extending dynamic mode decomposition. J. Nonlinear Sci. 25 (6), 1307–1346.

Wu, X., Zhang, W., Peng, X., Wang, Z., 2019. Benchmark aerodynamic shape optimization with the POD-based CST airfoil parametric method. Aerosp. Sci. Technol. 84, 632–640.

Wu, X., Zuo, Z., Ma, L., 2022. Aerodynamic data-driven surrogate-assisted teaching-learning-based optimization (TLBO) framework for constrained transonic airfoil and wing shape designs. Aerospace 9 (10), 610.

Xiao, Z., Xu, X., Xing, H., Luo, S., Dai, P., Zhan, D., 2021. RTFN: a robust temporal feature network for time series classification. Inf. Sci. 571, 65–86.

Xing, H., Xiao, Z., Zhan, D., Luo, S., Dai, P., Li, K., 2022. SelfMatch: Robust semisupervised time-series classification with self-distillation. Int. J. Intell. Syst. 37 (11), 8583–8610.

Yıldız, B.S., Mehta, P., Panagant, N., Mirjalili, S., Yildiz, A.R., 2022. A novel chaotic runge kutta optimization algorithm for solving constrained engineering problems. J. Comput. Design Eng. 9 (6), 2452–2465.

Yu, Y., Lyu, Z., Xu, Z., Martins, J.R., 2018. On the influence of optimization algorithm and initial design on wing aerodynamic shape optimization. Aerosp. Sci. Technol. 75, 183–199.

Zhang, X., Xie, F., Ji, T., Zhu, Z., Zheng, Y., 2021. Multi-fidelity deep neural network surrogate model for aerodynamic shape optimization. Comput. Methods Appl. Mech. Engrg. 373, 113485.

Zhao, T., Qian, W., Lin, J., Chen, H., Ao, H., Chen, G., He, L., 2023. Learning mappings from iced airfoils to aerodynamic coefficients using a deep operator network. J. Aerosp. Eng. 36 (5), 04023035.

Zhiwei, S., Chen, W., Zheng, Y., Junqiang, B., Zheng, L., Qiang, X., Qiujun, F., 2020. Non-intrusive reduced-order model for predicting transonic flow with varying geometries. Chin. J. Aeronaut. 33 (2), 508–519.

Zhu, M., Zhang, H., Jiao, A., Karniadakis, G.E., Lu, L., 2022. Reliable extrapolation of deep neural operators informed by physics or sparse observations. arXiv preprint arXiv:2212.06347.

Zou, Z., Meng, X., Psaros, A.F., Karniadakis, G.E., 2022. Neuraluq: A comprehensive library for uncertainty quantification in neural differential equations and operators. arXiv preprint arXiv:2208.11866.