J. Fluid Mech. (2021), *vol.* 919, A34, doi:10.1017/jfm.2021.398

Numerical investigation of minimum drag profiles in laminar flow using deep learning surrogates

Li-Wei Chen[1],†, Berkay A. Cakal[1], Xiangyu Hu[2] and Nils Thuerey[1]

[1]Department of Informatics, Technical University of Munich, D-85748 Garching, Germany

[2]Department of Mechanical Engineering, Technical University of Munich, D-85748 Garching, Germany

(Received 28 September 2020; revised 10 March 2021; accepted 29 April 2021)

Efficiently predicting the flow field and load in aerodynamic shape optimisation remains a highly challenging and relevant task. Deep learning methods have been of particular interest for such problems, due to their success in solving inverse problems in other fields. In the present study, U-net-based deep neural network (DNN) models are trained with high-fidelity datasets to infer flow fields, and then employed as surrogate models to carry out the shape optimisation problem, i.e. to find a minimal drag profile with a fixed cross-sectional area subjected to a two-dimensional steady laminar flow. A level-set method as well as Bézier curve method are used to parameterise the shape, while trained neural networks in conjunction with automatic differentiation are utilised to calculate the gradient flow in the optimisation framework. The optimised shapes and drag force values calculated from the flow fields predicted by the DNN models agree well with reference data obtained via a Navier–Stokes solver and from the literature, which demonstrates that the DNN models are capable not only of predicting flow field but also yielding satisfactory aerodynamic forces. This is particularly promising as the DNNs were not specifically trained to infer aerodynamic forces. In conjunction with a fast runtime, the DNN-based optimisation framework shows promise for general aerodynamic design problems.

Key words: computational methods, Navier–Stokes equations, general fluid mechanics

1. Introduction

Owing to its importance in a wide range of fundamental studies and industrial applications, significant effort has been made to study the shape optimisation for minimising aerodynamic drag over a bluff body (Bushnell & Moore 1991; Bushnell 2003). The deployment of computational fluid dynamics tools has played an important role in

† Email address for correspondence: liwei.chen@tum.de

these optimisation problems (Thévenin & Janiga 2008). While a direct optimisation via high-fidelity computational fluid dynamics models gives reliable results, the high computational cost of each simulation, e.g. for Reynolds-averaged Navier–Stokes formulations, and the large amount of evaluations needed lead to assessments that such optimisations are still not feasible for practical engineering (Skinner & Zare-Behtash 2018). When considering gradient-based optimisation, the adjoint method provides an effective way to calculate the gradients of an objective function with respect to design variables and alleviates the computational workload greatly (Jameson 1988; Giles & Pierce 2000; Economon, Palacios & Alonso 2013; Kline, Economon & Alonso 2016; Zhou *et al.* 2016), but the number of required adjoint computational fluid dynamics simulations is typically still prohibitively expensive when multiple optimisation objectives are considered (Mueller & Verstraete 2019). In gradient-free methods (e.g. genetic algorithm), the computational cost rises dramatically as the number of design variables is increased, especially when the convergence requirement is tightened (Zingg, Nemec & Pulliam 2008). Therefore, advances in terms of surrogate-based optimisation are of central importance for both gradient-based and gradient-free optimisation methods (Queipo *et al.* 2005; Sun & Wang 2019).

Recently, state-of-the-art deep learning methods and architectures have been successfully developed to achieve fast prediction of fluid physics. Among others, Bhatnagar *et al.* (2019) developed the convolutional neural network method for aerodynamics flow fields, while others studied the predictability of laminar flows (Chen, Viquerat & Hachem 2019), employed graph neural networks to predict transonic flows (de Avila Belbute-Peres, Economon & Kolter 2020) or learned reductions of numerical errors in partial differential equation fluid solvers (Um *et al.* 2020). For the inference of Reynolds-averaged Navier–Stokes solutions, a U-net-based deep learning model was proposed and shown to be significantly faster than a conventional computational fluid dynamics solver (Thuerey *et al.* 2018). These promising achievements open up new possibilities of applying deep neural network (DNN)-based flow solvers in aerodynamic shape optimisation. In the present study we focus on evaluating the accuracy and performance of DNN-base surrogates in laminar flow regimes.

Modern deep learning methods are also giving new impetus to aerodynamic shape optimisation research. For example, Eismann, Bartzsch & Ermon (2017) used a data-driven Bayesian approach to design optimisation and generated object shapes with an improved drag coefficient. Viquerat & Hachem (2019) evaluated quantitative predictions such as drag forces using a VGG-like convolutional neural network. To improve the surrogate-based optimisation, Li *et al.* (2020) proposed a new sampling method for airfoils and wings based on a generative adversarial network. Renganathan, Maulik & Ahuja (2020) designed a surrogate-based framework by training a DNN that was used for gradient-based and gradient-free optimisations. In these studies, the neural network is mainly trained to construct the mapping between shape parameters and aerodynamic quantities (e.g. lift and drag coefficients), but no flow-field information can be obtained from the network models. We instead demonstrate how deep learning models that were not specifically trained to infer the parameters to be minimised can be used in optimisation problems. The proposed deep learning model offers two advantages. First, the model is flexible as it predicts a full flow-field in a region of interest. Once trained, it can be used in different optimisation tasks with multiple objectives. This is of particular importance when considering problems such as compressor/turbine blade wake mixing (Michelassi *et al.* 2015). Second, as the model is differentiable, it can be seamlessly integrated with

https://doi.org/10.1017/jfm.2021.398 Published online by Cambridge University Press

deep learning algorithms (de Avila Belbute-Peres *et al.* 2018; Holl, Thuerey & Koltun 2020).

To understand the mechanisms underlying drag reduction and to develop optimisation algorithms, analytical and computational works have been specifically performed for Stokes flow and laminar steady flow over a body (Pironneau 1973, 1974; Glowinski & Pironneau 1975, 1976; Katamine *et al.* 2005; Kim & Kim 2005; Kondoh, Matsumori & Kawamoto 2012). As far back as the 1970s, Pironneau (1973) analysed the minimum drag shape for a given volume in Stokes flow, and later for the Navier–Stokes equations (Pironneau 1974). By using the adjoint variable approach, Kim & Kim (2005) investigated the minimal drag profile for a fixed cross-sectional area in two-dimensional laminar flow with Reynolds number range of $Re = 1$ to 40. More recently Katamine *et al.* (2005) studied the same problem at two Reynolds numbers, $Re = 0.1$ and $Re = 40$. With theoretical and numerical approaches, Glowinski & Pironneau (1975, 1976) looked for the axisymmetric profile of given area and smallest drag in a uniform incompressible laminar viscous flow at Reynolds numbers between 100 and 10^5 and obtained a drag-minimal shape with a wedge of angle 90° at the front end and a cusp rear end from an initial slender profile. Although the laminar flow regimes are well studied, due to the separation and nonlinear nature of the fluid, it can be challenging for surrogate models to predict a drag-minimal shape as well as aerodynamic forces. Moreover, with the Reynolds number approaching zero, the flow field experiences a dramatic change from a separated vortical flow towards a creeping flow, which poses additional difficulties for the learning task. To our knowledge, no previous studies exist that target training a ‘generalised model’ that performs well in such a Reynolds number range. We investigate this topic and quantitatively assess the results in the context of deep learning surrogates.

In the present paper, we adopt an approach for U-net-based flow-field inference (Thuerey *et al.* 2018) and use the trained DNN as a flow solver in shape optimisation. In comparison with conventional surrogate models (Yondo, Andrés & Valero 2018) and other optimisation work involving deep learning (Eismann *et al.* 2017; Viquerat & Hachem 2019; Li *et al.* 2020; Renganathan *et al.* 2020), we make use of a generic model that infers flow solutions: in our case it produces fluid pressure and velocity as field quantities. i.e. given encoded boundary conditions and shape, the DNN surrogate produces a flow-field solution, from which the aerodynamic forces are calculated. Thus, both the flow field and aerodynamic forces can be obtained during the optimisation. As we can fully control and generate arbitrary amounts of high-quality flow samples, we can train our models in a fully supervised manner. We use the trained DNN models in shape optimisation to find the minimal drag profile in the two-dimensional steady laminar flow regime for a fixed cross-sectional area, and evaluate results with respect to shapes obtained using a full Navier–Stokes flow solver in the same optimisation framework. We specifically focus on the challenging Reynolds number range from 1 to 40. Methods based on both level set and Bézier curve are employed for shape parameterisation. The implementation utilises the automatic differentiation package of the PyTorch package (Paszke *et al.* 2019), so the gradient flow driving the evolution of shapes can be directly calculated (Kraft 2017). Here DNN-based surrogate models show particular promise as they allow for a seamless integration into the optimisation algorithms that are commonly used for training DNNs.

The purpose of the present work is to demonstrate the capability of deep learning techniques for robust and efficient shape optimisation, and for achieving an improved understanding of the inference of the fundamental phenomena involved in low-Reynolds-number flows. This paper is organised as follows. The mathematical formulation and numerical method are briefly presented in § 2. The neural network architecture and training procedure are described in § 3. Details of the experiments and the results are then presented in § 4 and concluding remarks in § 5.

2. Methodology

We first explain and validate our approach for computing the fluid flow environment in which shapes should be optimised. Afterwards, we describe two different shape parameterisiations, based on level-set and Bézier curve, which we employ for our optimisation results.

2.1. *Numerical procedure*

We consider two-dimensional incompressible steady laminar flows over profiles of given area and look for the minimal drag design. The profile is initialised with a circular cylinder and updated by utilising steepest gradient descent as optimisation algorithm. The Reynolds number Re_D in the present work is based on the diameter of the initial circular cylinder. It can be also interpreted that the the length scale is defined as the equivalent diameter for given area S of an arbitrary shape, i.e. $D = 2\sqrt{S/\pi}$. In the present work, $D \approx 0.39424$ m is used.

To calculate the flow field around the profile at each iteration of the optimisation, two methods are employed in the present study. The first approach is a conventional steady solver of Navier–Stokes equations, i.e. simpleFoam within the open-source package OpenFOAM (from https://openfoam.org/). The second one is the deep learning model (Thuerey *et al.* 2018), which is trained with flow-field datasets generated by simpleFoam that consists of several thousand profiles at a chosen range of Reynolds numbers. More details of the architecture of the neural network, data generation, training and performance are discussed in § 3.

SimpleFoam is a steady-state solver for incompressible, turbulent flow using the semi-implicit method for pressure linked equations (known as SIMPLE) (Patankar & Spalding 1983). The governing equations are numerically solved by a second-order finite-volume method (Versteeg & Malalasekera 2007). The unstructured mesh in the fluid domain is generated using open-source code Gmsh version 4.4.1. To properly resolve the viscous flow, the mesh resolution is refined near the wall of the profile and the minimum mesh size is set as $\sim 6 \times 10^{-3}D$, where D is the equivalent circular diameter of the profile. The outer boundary, where the free-stream boundary condition is imposed, is set as 50 m ($\sim 32D$) away from the wall (denoted as OpenFOAM DOM50). The effects of domain size are assessed by performing additional simulations with domain sizes of 25 and 100 m away from the wall (denoted as OpenFOAM DOM25 and OpenFOAM DOM100, respectively). Here the drag coefficient C_d is defined as the drag force divided by the projected length and dynamic head. As shown in figure 1, from $Re_D = 0.1$ to $Re_D = 40$, the total C_d as well as the viscous $C_{d,v}$ and inviscid $C_{d,p}$ parts obtained from the three different domains almost collapse. Although small differences are observed when $Re_D < 0.5$, the predictions in the range of interest [1, 40] are consistent and not sensitive to the domain size. The computation runs for 6000 iterations to obtain a converged state.

To validate the set-up, we compare our numerical results and literature data in terms of the surface pressure coefficient and wall shear. As 'sanity checks' for the numerical set-up, we also run SU2 (see Economon *et al.* 2016) with the same mesh for comparison. Figure 2(*a*) shows the distribution of the surface pressure coefficient $(p_w - p_\infty)/0.5\rho_\infty U_\infty^2$ at $Re_D = 1$, 10 and 40. Here, θ is defined as the angle of the

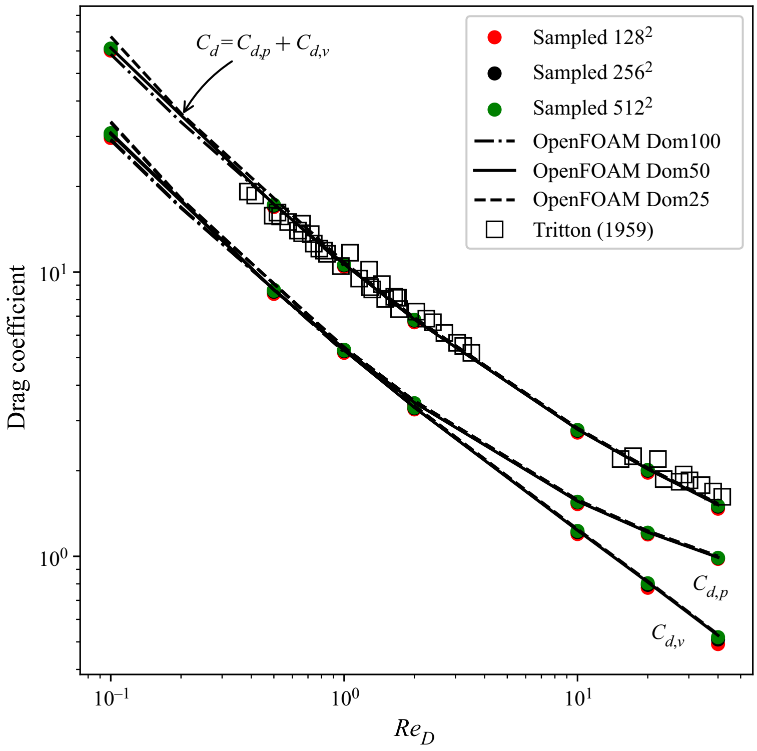

Figure 1. Drag coefficients from $Re_D = 0.1$ to 40. Surface integral values from OpenFOAM simulations are plotted as black curves. Results based on re-sampled points on Cartesian grids with resolutions of 128×128, 256×256 and 512×512 are plotted as red, black and green circles, respectively. All data are compared with the experimental measurements of Tritton (1959), which are shown as squares.

intersection of the horizontal line and the vector of the centre to a local surface point, so that $\theta = 0^\circ$ is the stagnation point in the upwind side and $\theta = 180^\circ$ that in the downwind side. Only half of the surface distribution is shown due to symmetry. The results agree well with the numerical results of Dennis & Chang (1970), and the results for OpenFOAM and SU2 collapse. In figure 2(*b*), the results for OpenFOAM compare well with those predicted by SU2. The drag coefficients for Reynolds numbers ranging from 0.1 to 40 agree well with the experimental data of Tritton (1959) in figure 1, which further supports that the current set-up and the solver produce reliable data.

To facilitate neural networks with convolution layers, the velocity and pressure field from OpenFOAM in the region of interest are re-sampled with a uniform Cartesian grid in a rectangular domain $[-1, 1]^2$ $(\approx [-1.27D, 1.27D]^2)$. A typical resolution used in the present study is 128×128, corresponding to a grid size of $0.02D$. As also shown in figure 1, the effect of the resolution of re-sampling on the drag calculation has been studied. The details of the force calculation on Cartesian grids are given in § 2.2.1. Results with three different resolutions shown as coloured symbols, i.e. 128^2, 256^2 and 512^2, compare favourably with the surface integral values based on the original mesh in OpenFOAM. Therefore, re-sampled fields on the 128×128 grid are used in the deep learning framework and optimisation unless otherwise noted.

https://doi.org/10.1017/jfm.2021.398 Published online by Cambridge University Press

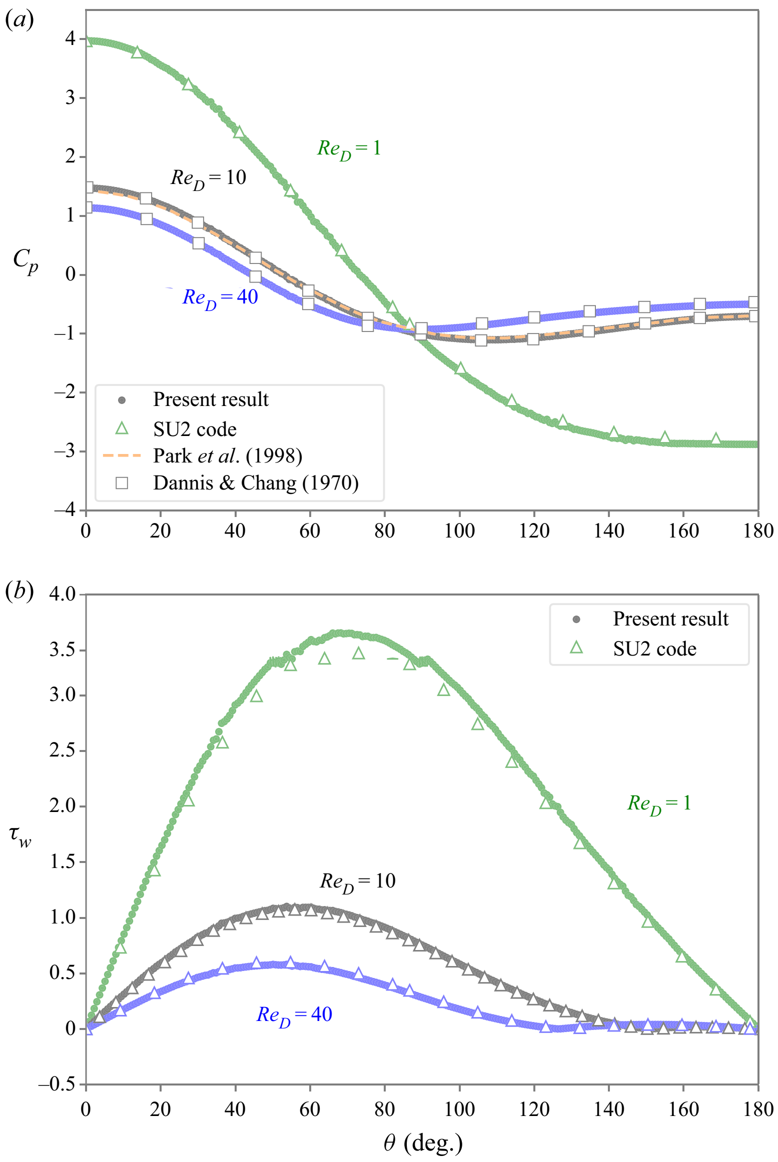

Figure 2. Pressure coefficient and wall shear stress distributions. (*a*) Pressure coefficient distribution and (*b*) wall shear stress distribution.

2.2. *Shape parameterisation*

2.2.1. *Level-set method*

The level-set method proposed by Osher & Sethian (1988) is a technique that tracks an interface implicitly and has been widely used in fluid physics, image segmentation, computer vision as well as shape optimisation (Sethian 1999*a*; Sethian & Smereka 2003; Baeza *et al.* 2008). The level-set function ϕ is a higher-dimensional auxiliary scalar function, the zero-level-set contour of which is the implicit representation of a time-dependent surface $\Gamma(t) = \{\boldsymbol{x} : \phi(\boldsymbol{x}) = 0\}$. Here, let $\mathcal{D} \in \mathcal{R}^{\mathcal{N}}$ be a reference domain, $\boldsymbol{x} \in \mathcal{D}$ and Ω is a body created by the enclosed surface Γ. Specifically in the present study, the domain $\mathcal{D}$ is referred to the sampled Cartesian grid in the rectangular region, and $\mathcal{N}$ is 2 as we focus on two-dimensional problems. The level-set function ϕ is defined by a signed distance function:

$$\phi = \begin{cases} -d(\varGamma(t)) & \boldsymbol{x} \in \varOmega \\ 0 & \boldsymbol{x} \in \partial\varOmega \ (\text{on } \varGamma) \\ d(\varGamma(t)) & \boldsymbol{x} \in \mathcal{D} - \varOmega, \end{cases} \tag{2.1}$$

where $d(\varGamma(t))$ denotes the Euclidean distance from $\boldsymbol{x}$ to $\varGamma$.

The arc length c and area S of the body are formulated as $c = \int_{\mathcal{D}} \delta_\epsilon(\phi)|\nabla\phi|\,\mathrm{d}s$ and $S = \int_{\mathcal{D}} H_\epsilon(-\phi)\,\mathrm{d}s$. To make the operators differentiable, in the above, we use smoothed Heaviside and Dirac delta functions $H_\epsilon(x) = 1/(1+\mathrm{e}^{-x/\epsilon})$ and $\delta_\epsilon(x) = \partial_x(1/(1+\mathrm{e}^{-x/\epsilon}))$, respectively. Here ϵ is a small positive number and chosen as twice the grid size (Zahedi & Tornberg 2010). Then, the aerodynamic forces due to pressure distribution and viscous effect are described as

$$\boldsymbol{F}_{pressure} = \int_{\partial\varOmega} (p\boldsymbol{n})\,\mathrm{d}l = \int_{\mathcal{D}} (p\boldsymbol{n})\delta_\epsilon(\phi)|\nabla\phi|\,\mathrm{d}s, \tag{2.2}$$

$$\boldsymbol{F}_{viscous} = \int_{\partial\varOmega} (\mu\boldsymbol{n}\times\boldsymbol{\omega})\,\mathrm{d}l = \int_{\mathcal{D}} (\mu\boldsymbol{n}\times\boldsymbol{\omega})\delta_\epsilon(\phi)|\nabla\phi|\,\mathrm{d}s. \tag{2.3}$$

Here, $\boldsymbol{n}$ is the unit normal vector, $\boldsymbol{n} = \nabla\phi/|\nabla\phi|$, p is the pressure, μ is the dynamic viscosity and $\boldsymbol{\omega} = \nabla\times\boldsymbol{v}$ is the vorticity with $\boldsymbol{v}$ being the velocity. A nearest-neighbour method is used to extrapolate values of pressure and vorticity inside the shape $\varOmega$. Then, the drag force is considered as the loss in the optimisation, i.e.

$$\mathcal{L} = \boldsymbol{F}_{pressure}\cdot\hat{\boldsymbol{i}}_x + \boldsymbol{F}_{viscous}\cdot\hat{\boldsymbol{i}}_x, \tag{2.4}$$

where $\hat{\boldsymbol{i}}_x$ is the unit vector in the direction of the x axis.

The minimisation of (2.4) is solved by the following equation:

$$\frac{\partial\phi}{\partial\tau} + V_n|\nabla\phi| = 0. \tag{2.5}$$

Here, the normal velocity is defined as $V_n = \partial\mathcal{L}/\partial\phi$. At every iteration, the Eikonal equation is solved numerically with the fast marching method to ensure $|\nabla\phi| \approx 1.0$ (Sethian 1999*b*). Then, we have $\partial\phi/\partial\tau \propto -\partial\mathcal{L}/\partial\phi$, which is a gradient flow that minimises the loss function $\mathcal{L}$ and drives the evolution of the profile (He, Kao & Osher 2007). For a more rigorous mathematical analysis we refer to Kraft (2017). In the present work, the automatic differentiation functionality of PyTorch is utilised to efficiently minimise (2.4) via gradient descent. Note that the level-set-based surface representation and optimisation algorithm are relatively independent modules, and can be coupled with any flow solver, such as OpenFOAM and SU2, so long as the solver provides a re-sampled flow field on the Cartesian grid (e.g. 128×128) at an iteration in the optimisation. We will leverage this flexibility by replacing the numerical solver with a surrogate model represented by a trained neural network below.

2.2.2. *Bézier-curve-based parameterisation*

Bézier-curve-based parametric shape parameterisation is a widely accepted technique in aerodynamic studies (Gardner & Selig 2003; Yang *et al.* 2018; Zhang *et al.* 2020).

This work utilises two Bézier curves, representing upper and lower surfaces of the profile denoted with superscript $k = \{u, l\}$. Control points $\boldsymbol{P}_i^k \in \mathcal{D}$ are the parameters of the optimisation framework. The Bézier curves are defined via the following equation:

$$\boldsymbol{B}^k(t) = \sum_{i=0}^{n} \binom{n}{i} t^i (1-t)^{n-i} \boldsymbol{P}_i^k, \tag{2.6}$$

where $t \in [0, 1]$ denotes the sample points along the curves. The first and last control points of each curve share the same parameters to construct the closure $\varOmega$ of the profile.

A binary labelling of the Cartesian grid $\mathcal{D}$ is performed as

$$\chi = \begin{cases} 1 & \boldsymbol{x} \in \varOmega \\ 0 & \boldsymbol{x} \in \mathcal{D} - \varOmega, \end{cases} \tag{2.7}$$

where χ is the binary mask of the profile and $\boldsymbol{x}$ is the coordinate of a point on the Cartesian grid. The normal vector $\boldsymbol{n}$ is obtained via applying a convolution with a 3×3 Sobel operator kernel on χ. Then, forces are calculated as

$$\boldsymbol{F}_{pressure} = \sum_{i \in \mathcal{D}-\varOmega} (p\boldsymbol{n})_i \Delta l_i, \tag{2.8}$$

$$\boldsymbol{F}_{viscous} = \sum_{i \in \mathcal{D}-\varOmega} (\mu \boldsymbol{n} \times \boldsymbol{\omega})_i \Delta l_i, \tag{2.9}$$

where i is the index of a point outside the profile and Δl_i is the grid size at the point i. Thereby, drag $\mathcal{L}$ is calculated using (2.4). As for the level set representation, the shape gradient $\partial \mathcal{L}/\partial \boldsymbol{P}_i^k$ is computed via automatic differentiation in order to drive the shape evolution to minimise $\mathcal{L}$.

3. Neural network architecture and training procedure

3.1. *Architecture*

The neural network model is based on a U-net architecture (Ronneberger, Fischer & Brox 2015), a convolutional network originally used for the fast and precise segmentation of images. Following the methodology of previous work (Thuerey *et al.* 2018), we consider the inflow boundary conditions (i.e. u_∞, v_∞) and the shape of profiles (i.e. the binary mask) on the 128×128 Cartesian grid as three input channels. In the encoding part, 7 convolutional blocks are used to transform the input (i.e. $128^2 \times 3$) into a single data point with 512 features. The decoder part of the network is designed symmetrically with another 7 layers in order to reconstruct the outputs with the desired dimension, i.e. $128^2 \times 3$, corresponding to the flow-field variables $[p, u, v]$ on the 128×128 Cartesian grid. Leaky ReLU activation functions with a slope of 0.2 are used in the encoding layers, and regular ReLU activations in the decoding layers.

In order to assess the performance of the deep learning models, we have tested three different models with varying weight counts of 122 000, 1.9×10^6 and 30.9×10^6, respectively, which are later referred to as small-, medium- and large-scale networks.

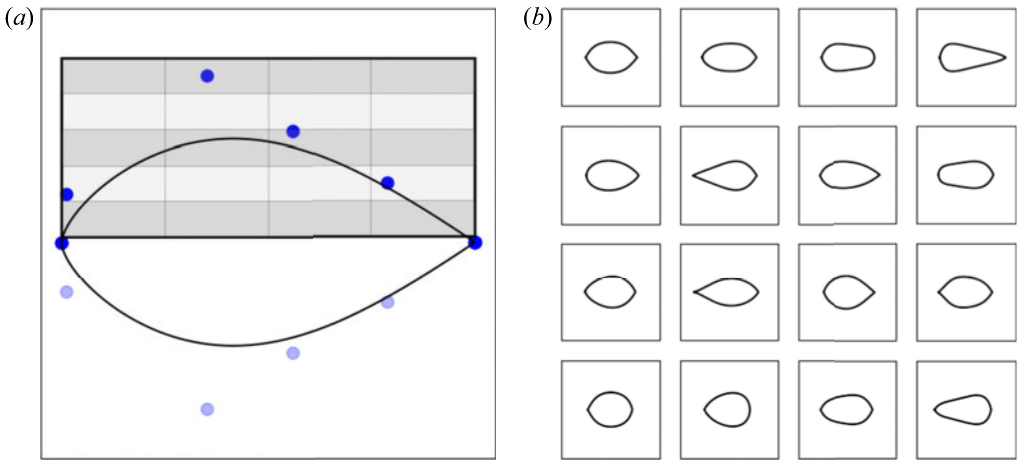

Figure 3. Shape generation using two Bézier curves. The region of interest is divided into four columns, and each column-wise region is further split into five subregions. (*a*) Bézier control points and (*b*) randomly generated shapes.

3.2. *Dataset generation*

For the training dataset, it is important to have a comprehensive coverage of the space of targeted solutions. In the present study, we utilise the parametric Bézier curve defined by (2.6) to generate randomised symmetric shape profiles subject to a fixed area constraint S.

To parameterise the upper surface of the profile, two points at the leading and trailing edges are fixed and four control points are positioned in different regions. As depicted in figure 3(*a*), the region of interest is divided into four columns separated by the border lines, and each control point of the upper Bézier curve is only allowed to be located within its corresponding column-wise region. Each column-wise region is further split into five subregions to produce diversified profiles. The subregions give $5^4 = 625$ possible permutations, with control points being placed randomly in each subregion. This procedure is repeated for four times, in total producing $4 \times 625 = 2500$ Bézier curves. Figure 3(*b*) shows some examples from this set.

Based on these 2500 geometries, we then generate three sets of training data, as summarised in table 1.

(1) We run OpenFOAM with fixed $Re_D = 1$ for all of the 2500 profiles to obtain 2500 flow fields, denoted as Dataset-1.
(2) The second dataset is similar but all of the 2500 simulations are conducted at $Re_D = 40$ (Dataset-40).
(3) The third dataset is generated to cover a continuous range of Reynolds numbers, in order to capture a space of solutions that not only varies over the immersed shapes, but additionally captures dimensions of varying flow physics with respect to a chosen Reynolds number. For this, we run a simulation by randomly choosing a profile Ω_i^* among 2500 geometries and a Reynolds number in the range $Re_D^* \in [0.5, 42.5]$. As we know that drag scales logarithmically with respect to Reynolds number, we similarly employ a logarithmic sampling for the Reynolds number dimension. We use a uniform distribution random variable $\kappa \in [\log 0.5, \log 42.5]$, leading to a $Re_D^* = 10^\kappa$ uniformly distributed in log scale. In total we have obtained 8640 flow fields, which we refer to as Dataset-Range. With this size of the training dataset,

Name	No. of flow fields	*Re*	Neural network model
Dataset-1	2500	1	Small, medium and large
Dataset-40	2500	40	Small, medium and large
Dataset-Range	8640	0.5–42.5	Large

Table 1. Three datasets for training the neural network models.

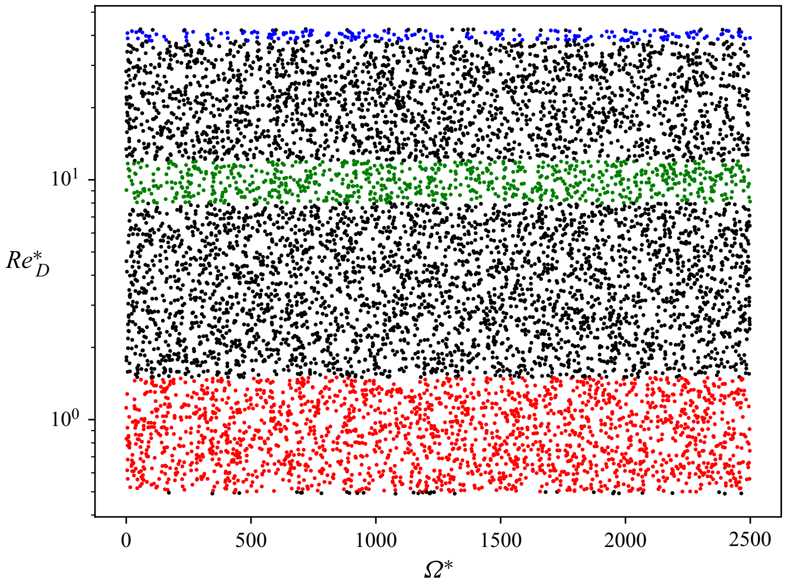

Figure 4. Distribution of flow-field samples from Dataset-Range on the Ω^*_i–Re^*_D map. The indices of geometries Ω^*_i are from 0 to 2499. The red symbols denote the flow-field samples with $Re^*_D \in [0.5, 1.5]$, the green ones with $Re^*_D \in [8, 12]$ and the blue ones with $Re^*_D \in [38, 42]$.

the model performance converges to a stable prediction accuracy for training and validation losses, as shown in the Appendix.

Shown in figure 4 is the distribution of all the flow-field samples from Dateset-Range on the Ω^*_i–Re^*_D map, with Re^*_D in log scale. It is worth noting that there are 2053 flow-field samples in the range $Re^*_D \in [0.5, 1.5]$ which are shown in red, 819 samples with $Re^*_D \in [8, 12]$ shown in green and 195 samples with $Re^*_D \in [38, 42]$ shown in blue.

3.3. *Preprocessing*

Proper preprocessing of the data is crucial for obtaining a high inference accuracy from the trained neural networks. Firstly, the non-dimensional flow-field variables are calculated using

$$\left.\begin{aligned} \hat{p}_i &= (p_i - p_{i,mean})/U^2_{\infty,i}, \\ \hat{u}_i &= u_i/U_{\infty,i}, \\ \hat{v}_i &= v_i/U_{\infty,i}. \end{aligned}\right\} \tag{3.1}$$

https://doi.org/10.1017/jfm.2021.398 Published online by Cambridge University Press

Here, i denotes the ith flow-field sample in the dataset, p_{mean} the simple arithmetic mean pressure and $U_\infty = \sqrt{u_\infty^2 + v_\infty^2}$ the magnitude of the free-stream velocity.

As the second step, all input channels and target flow-field data in the training dataset are normalised to the range $[-1, 1]$ in order to minimise the errors from limited precision in the training phase. To do so, we need to find the maximum absolute values for each flow variable in the entire training dataset, i.e. $|\hat{p}|_{max}$, $|\hat{u}|_{max}$ and $|\hat{v}_{max}|$. Similarly, the maximum absolute values of the free-stream velocity components are $|u_\infty|_{max}$ and $|v_\infty|_{max}$. Then we get the final normalised flow-field variables in the following form:

$$\left.\begin{aligned} \tilde{p}_i &= \hat{p}_i/|\hat{p}|_{max}, \\ \tilde{u}_i &= \hat{u}_i/|\hat{u}|_{max}, \\ \tilde{v}_i &= \hat{v}_i/|\hat{v}|_{max}, \end{aligned}\right\} \tag{3.2}$$

and the normalised free-stream velocities used for input channels are

$$\left.\begin{aligned} \tilde{u}_i &= u_i/\max(|u_\infty|_{max}, 1\times 10^{-18}), \\ \tilde{v}_i &= v_i/\max(|v_\infty|_{max}, 1\times 10^{-18}). \end{aligned}\right\} \tag{3.3}$$

The free-stream velocities appear in the boundary conditions, on which the solution globally depends, and should be readily available spatially and throughout the different layers. Thus, free-stream conditions and the shape of the profile are encoded in a $128^2 \times 3$ grid of values. The magnitude of the free-stream velocity is chosen such that it leads to a desired Reynolds number.

3.4. *Training details*

The neural network is trained with the Adam optimiser in PyTorch (Kingma & Ba 2014). A difference $L_1 = |y_{truth} - y_{prediction}|$ is used for the loss calculation. For most of the cases, the training runs converge after 100 000 iterations with a learning rate 6×10^{-4} and a batch size of 10 (unless otherwise mentioned). An 80 % to 20 % split is used for training and validation sets, respectively. The validation set allows for an unbiased evaluation of the quality of the trained model during training, for example, to detect overfitting. In addition, as learning rate decay is used, the variance of the learning iterations gradually decreases, which lets the training process fine-tune the final state of the model.

Figure 5 shows the training and validation losses for three models that are trained using Dataset-1, i.e. small-scale, medium-scale and large-scale models. All three models converge at stable levels of training and validation loss after 500 epochs. Looking at the training evolution for the small-scale model in figure 5(*a*), numerical oscillation can be seen in the early stage of the validation loss history, which is most likely caused by the smaller number of free parameters in the small-scale network. In contrast, the medium- and large-scale models show a smoother loss evolution, and the gap between validation and training losses indicates a slight overfitting as shown in figures 5(*b*) and 5(*c*). Although the training of the large-scale model exhibits a spike in the loss value at an early stage due to an instantaneous pathological configuration of mini-batch data and learned state, the network recovers, and eventually converges to lower loss values. Similar spikes can be seen in some of the other training runs, and could potentially be removed via gradient-clipping algorithms, which, however, we did not find necessary to achieve reliable convergence.

Figure 6 presents the training and validation losses for the three models trained with Dataset-40. Similarly, convergence can be achieved after 500 epochs. Compared with the training evolution at $Re_D = 1$, the models for $Re_D = 40$ have smaller gaps between training

https://doi.org/10.1017/jfm.2021.398 Published online by Cambridge University Press

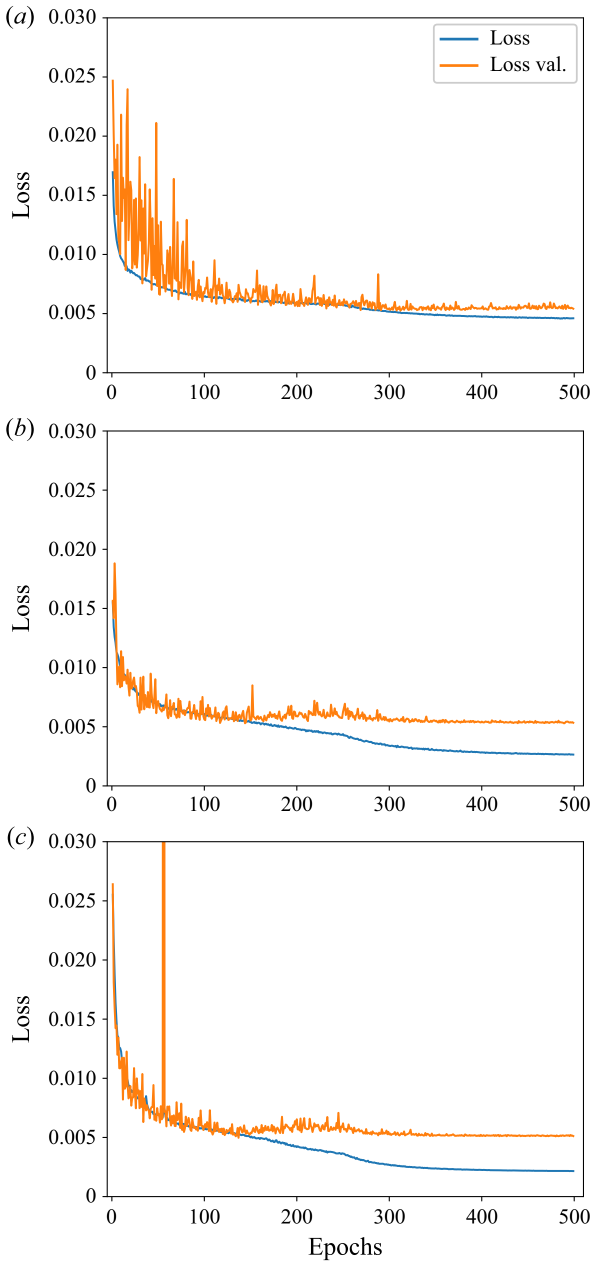

Figure 5. Training (in blue) and validation (in orange) losses of three different scales of models trained with Dataset-1. (*a*) Small-scale neural network, (*b*) medium-scale neural network and (*c*) large-scale neural network.

and validation losses, indicating that the overfitting is less pronounced than for $Re_D = 1$. We believe this is caused by the smoother and more diffusive flow fields at $Re_D = 1$ (close to Stokes flow), in contrast to the additional complexity of the solutions at $Re_D = 40$, which already exhibit separation bubbles.

We use Dataset-Range to train the model for a continuous range of Reynolds numbers. As this task is particularly challenging, we directly focus on the large-scale network that

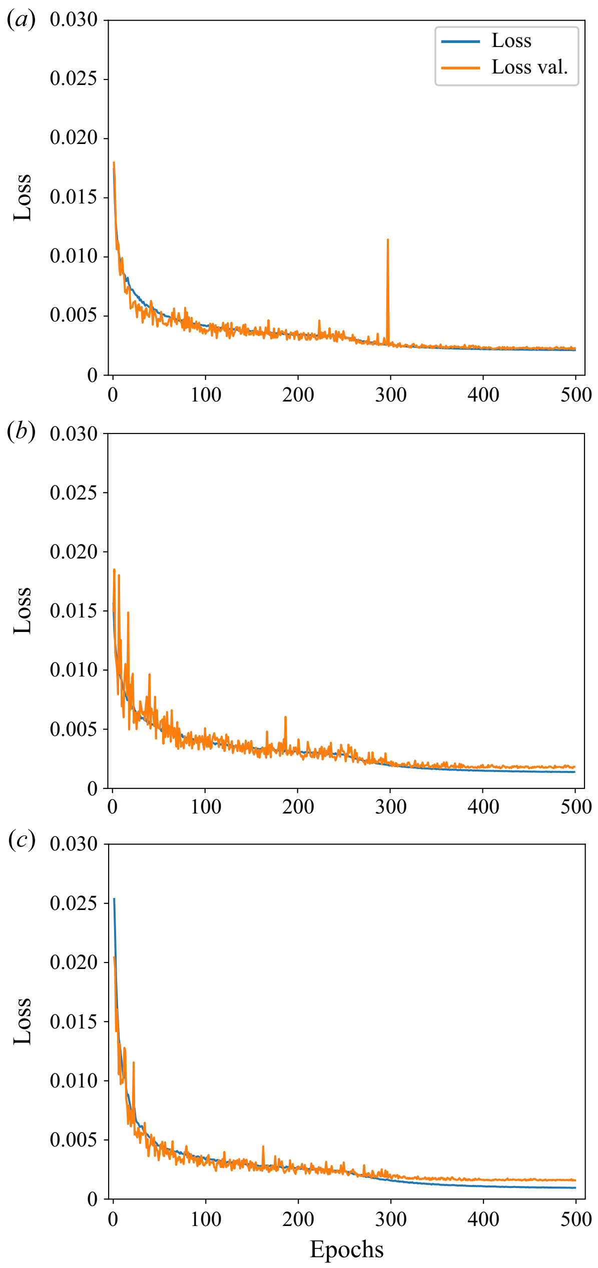

Figure 6. Training (in blue) and validation (in orange) losses of three different scales of models trained with Dataset-40. (*a*) Small-scale neural network, (*b*) medium-scale neural network and (*c*) large-scale neural network.

has 30.9×10^6 weights. To achieve better convergence for this case, we run 800 000 iterations with a batch size of 5, which leads to more than 485 epochs. As shown in figure 7, training and validation losses converge to stable levels, and do not exhibit overfitting over the course of the training iterations. The final loss values are 1.01×10^{-3} and 1.31×10^{-3}, respectively.

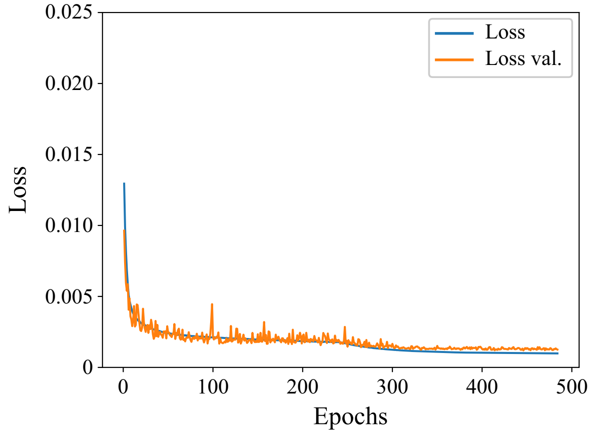

Figure 7. Training (in blue) and validation (in orange) losses of large-scale model trained with Dataset-Range.

To summarise, having conducted the above-mentioned training, we obtain seven neural network models, i.e. models of three network sizes each for Dataset-1 and Dataset-40 and a ranged model trained with Dataset-Range, as listed in table 1. These neural networks are used as surrogate models in the optimisation in the next section. We also compare the results from neural network models with corresponding optimisations conducted with the OpenFOAM solver, and evaluate the performance and accuracy of the optimisation runs.

4. Shape optimisation results

The initial shape for the optimisation is a circular cylinder with a diameter $D \approx 0.39424$ m. The integral value of the drag force using (2.4) is adopted as the objective function. The mathematical formula of the optimisation for the shape $\varOmega$ bounded by curve $\varGamma$, the surface of the profile, is expressed as

$$\left.\begin{array}{c} \min\ Drag(\varOmega) \\ \text{subject to area } S(\varOmega) = S_0 \\ \text{barycentre } \boldsymbol{b}(\varOmega) = \dfrac{1}{S(\varOmega)} \displaystyle\int_{\varOmega} \boldsymbol{x}\, \mathrm{d}s = (0, 0). \end{array}\right\} \tag{4.1}$$

For the level-set representation, the profile $\varOmega$ is the region where $\phi \leq 0$ and the constrained optimisation problem is solved as follows:

(1) Initialise level-set function ϕ such that the initial shape (i.e. a circular cylinder) corresponds to $\phi = 0$.
(2) For a given ϕ, calculate drag (i.e. loss $\mathcal{L}$) using (2.2)–(2.4). Terminate if the optimisation converges, e.g. drag history reaches a statistically steady state.
(3) Calculate the gradient $\partial\mathcal{L}/\partial\phi$. Consider an unconstrained minimisation problem and solve (2.5) as follows:

$$\phi^{n+1} \Longleftarrow \phi^n - \Delta\tau \frac{\partial\mathcal{L}}{\partial\phi}|\nabla\phi|. \tag{4.2}$$

In practice, we update ϕ using the second-order Runge–Kutta method, and discretise the convection term with a first-order upwind scheme (Sethian & Smereka 2003).

https://doi.org/10.1017/jfm.2021.398 Published online by Cambridge University Press

We assume derivatives of the flow-field variables (i.e. pressure and velocity) are significantly smaller than those with respect to the shape. Hence, we treat both fields as constants for each step of the shape evolution. To ensure the correct search direction for optimisation, we use a relatively small pseudo time step $\Delta\tau$, which is calculated with a Courant number of 0.8.

(4) To ensure $|\nabla\phi| \approx 1$, a fast marching method is used to solve the Eikonal equation (Sethian 1999*b*).

(5) The area of the shape Ω is obtained by $S = \int_{\mathcal{D}} H_\epsilon(-(\phi+\eta))\,\mathrm{d}s$, where η is an adjustable constant. We optimise η such that $|S - S_0| < \epsilon$. Then, we update $\phi^{n+1} \Longleftarrow \phi^{n+1} + \eta$.

(6) Check if the barycentre is at the origin: $|\boldsymbol{b} - \boldsymbol{o}| < \epsilon$. If not, solve (2.5) to update ϕ^{n+1} by replacing V_n with a translating constant velocity so that the barycentre of the shape Ω moves towards the origin. Continue with step (2).

While our main focus lies on level-set representations, the Bézier curve parameterisation with reduced degrees of freedom is used for comparison purposes. This highlights how differences in the shape parameterisation can influence the optimisation results. Thus, we include the Bezier parameterisation with very few control points and the level-set representation with a dense grid sampling as two extremes of the spectrum of shape representations. When Bézier curves are used, the constrained optimisation differs from the above-mentioned loop in the following way. In (1)–(3), the coordinates of Bézier curve control points are used as the design variables to be initialised and updated. In (5) and (6), the area of Ω and barycentre are calculated based on the region enclosed by the Bézier curve, where the binary mask of inner region is 1 and of outer region is 0.

In the optimisation experiments, the flow-field solvers used are OpenFOAM (as baseline) and small-, medium- and large-scale neural network models. As additional validation for the optimisation procedure, we also compare with additional runs based on the Bézier curve parameterisation with a large-scale neural network model. If the flow solver is OpenFOAM, Gmsh is automatically called to generate an unstructured mesh based on the curve $\phi = 0$ at every iteration. To update ϕ and calculate drag in step (2), as mentioned in §2.1, the flow-field variables are re-sampled on the 128×128 Cartesian grid.

4.1. *Optimisation experiment at $Re_D = 1$*

Figure 8 presents the drag coefficients over 200 optimisation iterations using OpenFOAM solver and the three neural network models. Here, the drag coefficient C_d is defined as drag divided by the projected length of the initial cylinder and dynamic head. The same definition is used for all other experiments in the present paper. As the ground truth, figure 8(*a*) shows the case which uses the OpenFOAM solver in the optimisation. The history of drag values, shown in blue, is calculated based on the re-sampled data on the Cartesian grid (i.e. 128^2). For comparison, the drag values obtained from the surface integral in the OpenFOAM native postprocessing code are shown with red markers. As can be seen in figure 8(*a*), after convergence of the optimisation, the total drag drops 6.3 % from 10.43 to 9.78. To further break it down, the inviscid part decreases significantly from 5.20 to 2.50 ($\sim$ 51.8 %) while the viscous part gradually increases from 5.23 to 7.27 ($\sim$ 31.0 %). This is associated with the elongation of the shape from a circular cylinder to an 'oval', eventually becoming a rugby-ball shape as shown in figure 9(*b*).

https://doi.org/10.1017/jfm.2021.398 Published online by Cambridge University Press

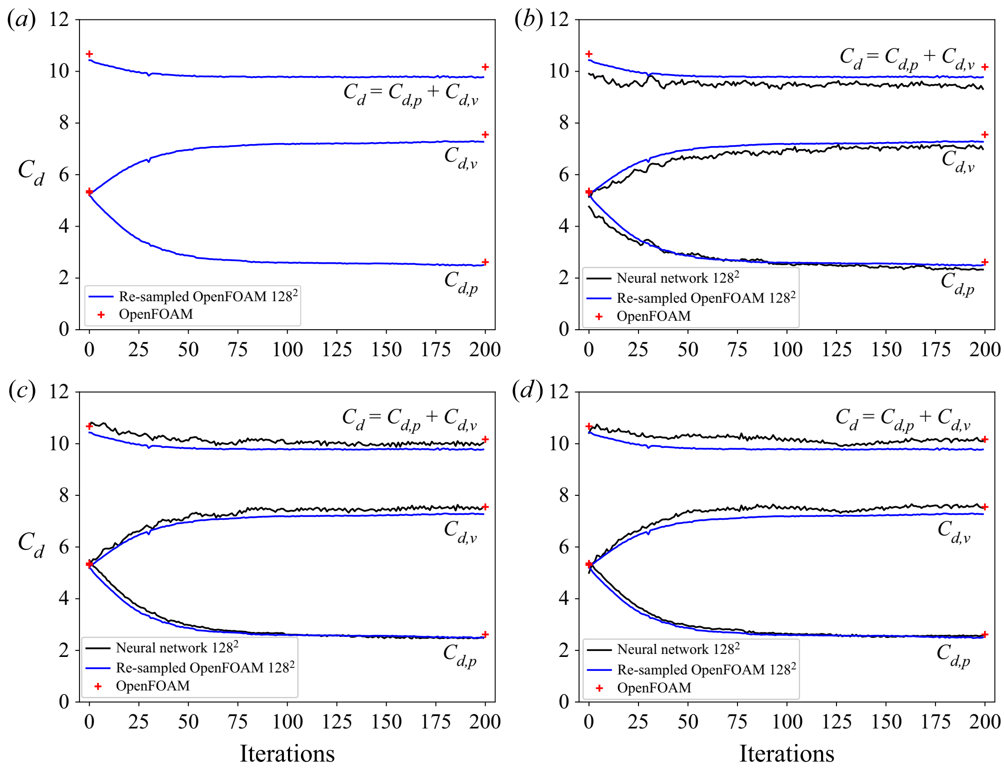

Figure 8. Optimisation histories at $Re_D = 1$. The black solid lines denote the results using neural network models trained with Dataset-1 and the blue solid lines denote the results from OpenFOAM. Results calculated with the re-sampled flow fields on the 128 × 128 Cartesian grid are denoted by 128^2. The red cross symbols represent the OpenFOAM results obtained with its native postprocessing tool. (*a*) OpenFOAM, (*b*) small-scale neural network, (*c*) medium-scale neural network and (*d*) large-scale neural network.

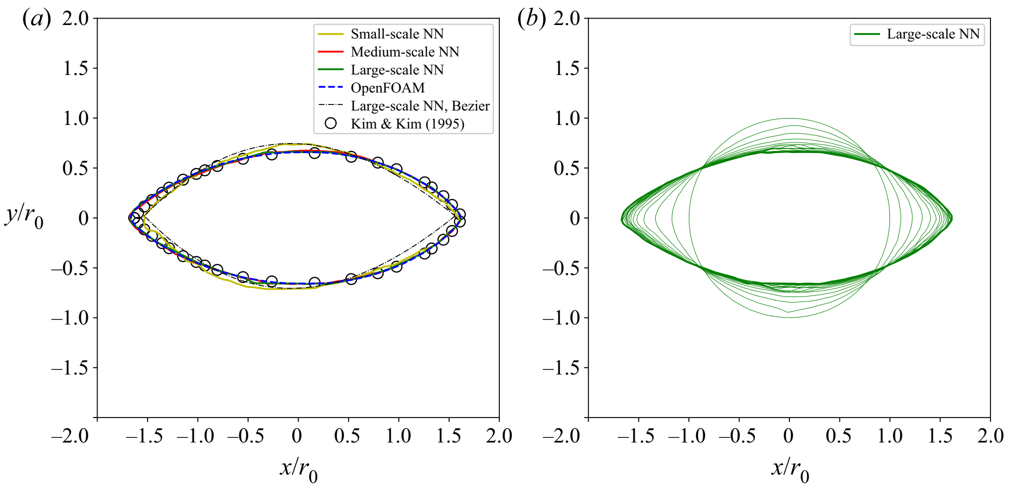

Figure 9. The converged shapes at $Re_D = 1$ (*a*) and the intermediate states at every 10th iteration predicted by the large-scale neural network (NN) model (*b*).

From figures 8(*b*) and 8(*c*), one can observe the histories of the drag values are reasonably well predicted by the neural network models and agree with the OpenFOAM solution in figure 8(*a*). Despite the small-scale model exhibiting noticeable oscillations in the optimisation procedure, the medium- and large-scale neural network models provide smoother predictions, and the drag of both initial and final shapes agrees well with that from re-sampled data (blue lines) and that from the OpenFOAM native postprocessing code (red symbols).

Figure 9(*a*) depicts the converged shapes of all four solvers. The ground truth result using OpenFOAM ends up with a rugby-ball shape which achieves a good agreement with the data of Kim & Kim (2005). The medium- and large-scale neural network models collapse and compare favourably with the ground truth result. In contrast, the small-scale neural network model's prediction is slightly off, which is not surprising as one can observe oscillation and offset of the drag history in figure 8(*b*) as discussed before. A possible reason is that the small-scale model has fewer weights so that the complexity of the flow evolution cannot be fully captured. It is worth noting that the reduced performance of the Bézier representation in the present work is partly due to the discretisation errors when calculating the normal vectors in combination with a reduced number of degrees of freedom.

The x-component velocity fields with streamlines for the optimised shapes are shown in figure 10. The flow fields and the patterns of streamlines in all three cases with neural networks show no separation, which is consistent with the ground truth result in figure 10(*a*). Considering the final shape obtained using the three neural network surrogates, the medium- and large-scale models give satisfactory results that are close to the OpenFOAM result.

4.2. *Optimisation experiment at $Re_D = 40$*

As the Reynolds number increases past the critical Reynolds number $Re_D \approx 47$, the circular cylinder flow configuration loses its symmetry and becomes unstable, which is known as the Karman vortex street. We consider optimisations for the flow regime at $Re_D = 40$ which is of particular interest because it exhibits a steady-state solution, yet is close to the critical Reynolds number. The steady separation bubbles behind the profile further compound the learning task and the optimisation, making it a good test case for the proposed method.

The ground truth optimisation result using OpenFOAM is shown in figure 11(*a*). The shape is initialised with a circular cylinder and is optimised to minimise drag over 200 iterations. As a result, the total drag, processed on the Cartesian grid, drops from 1.470 to 1.269 ($\sim$13.7 % reduction). Associated with the elongation of the shape, the inviscid drag decreases 41.3 % while the viscous drag increases 41.3 %. The initial and the final results of the OpenFOAM native postprocessing are shown in red, indicating good agreement. Figure 11(*b–d*) presents the drag histories over 200 optimisation iterations with three neural network models that are trained with Dataset-40. Although larger oscillations are found in the drag history of the small-scale model, the medium- and large-scale models predict smoother drag history and compare well with the ground truth data using OpenFOAM.

The final converged shapes are compared to a reference result (Katamine *et al.* 2005) in figure 12(*a*). The evolution of intermediate shapes from the initial circular cylinder towards the final shape is shown in figure 12(*b*). The upwind side forms a sharp leading edge while the downwind side of the profile develops into a blunt trailing edge. Compared with the

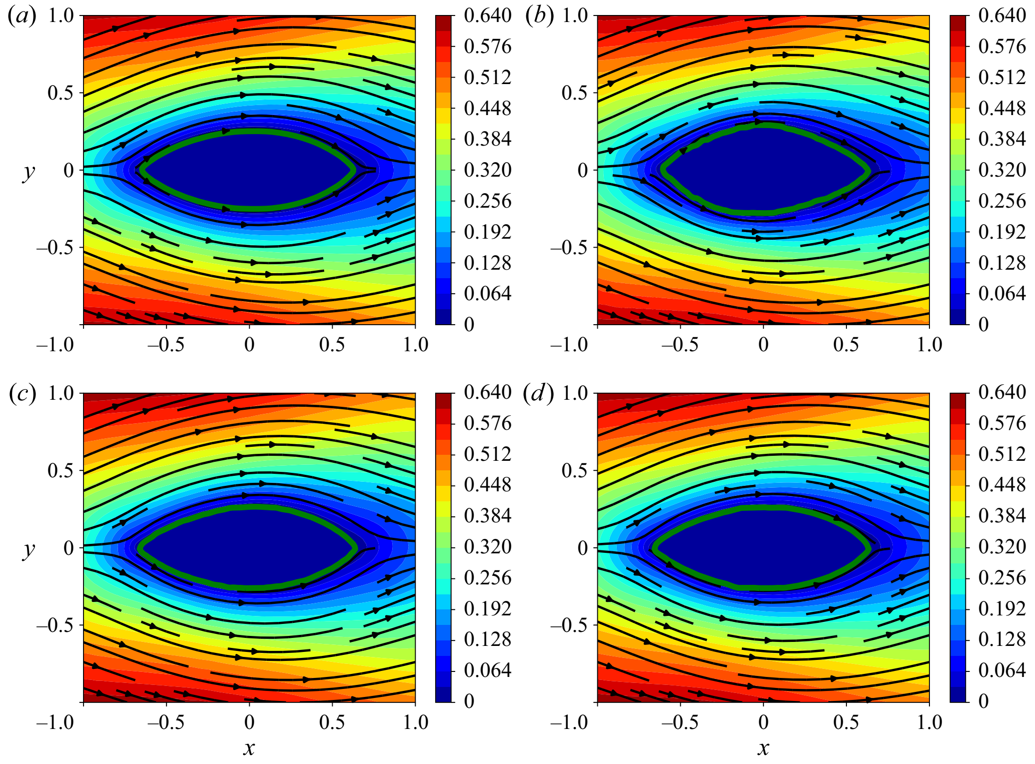

Figure 10. Streamlines and the x-component velocity fields u/U_∞ at $Re_D = 1$. (*a*) OpenFOAM, (*b*) small-scale neural network, (*c*) medium-scale neural network and (*d*) large-scale neural network.

reference data (Katamine *et al.* 2005) and the result using the Bézier-curve-based method, the use of level-set-based method leads to a slightly flatter trailing edge, probably because more degrees of freedom for the shape representation are considered in level-set-based method.

Further looking at the details of shapes in figure 13, it can be seen that the more weights the neural network model contains, the closer it compares with the ground truth result using OpenFOAM. The large-scale model, which has the largest weight count, is able to resolve the fine feature of the flat trailing edge as shown in figure 13(*d*). In contrast, in figure 13(*b*), the small-scale model does not capture that and even the the surface of the profile exhibits pronounced roughness. Nonetheless, all three DNN models predict similar flow patterns compared with the ground truth result depicted with streamlines, which are characterised with re-circulation regions downstream of the profiles.

It should be mentioned that the optimised shape at $Re_D = 40$ of Kim & Kim (2005) differs from the one in the present study and the one of Katamine *et al.* (2005). In the former (Kim & Kim 2005), the optimised profile converges at an elongated slender shape with an even smaller drag force. Most likely, this is caused by an additional wedge angle constraint being imposed at both leading and trailing edges, which is not adopted in our work and that of Katamine *et al.* (2005). As we focus on deep learning surrogates in the present study, we believe the topic of including additional constraints will be an interesting avenue for future work. In comparison with the ground truth from OpenFOAM, the current results are deemed to be in very good agreement.

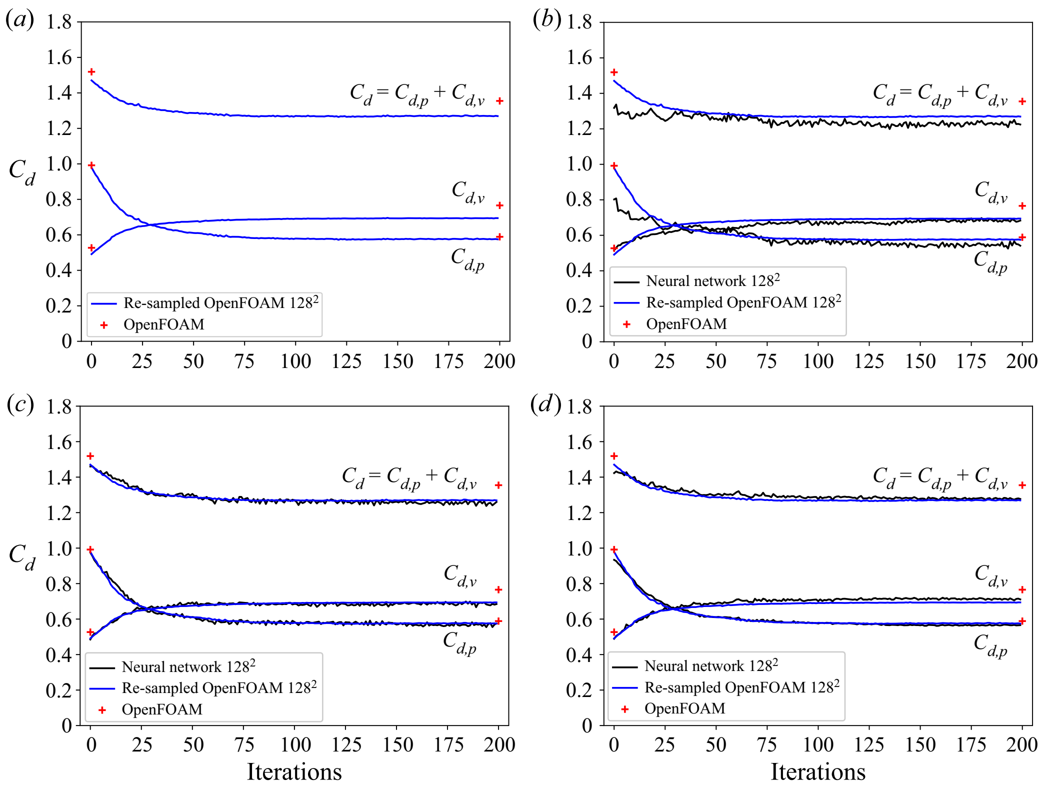

Figure 11. Optimisation histories at $Re_D = 40$. The black solid lines denote the results using neural network models trained with Dataset-40 and the blue solid lines denote the results from OpenFOAM. Results calculated with the re-sampled flow fields on the 128 × 128 Cartesian grid are denoted by 128^2. The red cross symbols represent the OpenFOAM results obtained with its native postprocessing tool. (*a*) OpenFOAM, (*b*) small-scale neural network, (*c*) medium-scale neural network and (*d*) large-scale neural network.

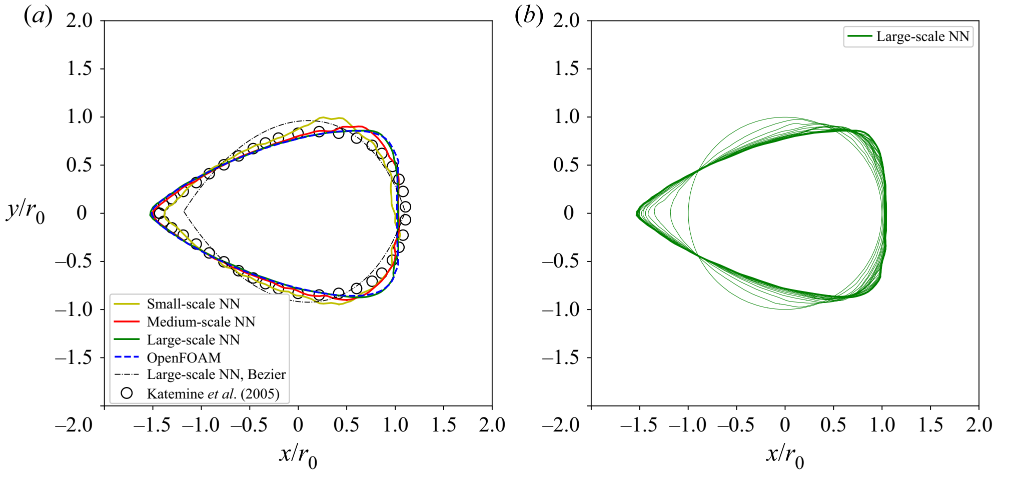

Figure 12. The converged shapes at $Re_D = 40$ (*a*) and the intermediate states at every 10th iteration predicted by the large-scale neural network (NN) model (*b*).

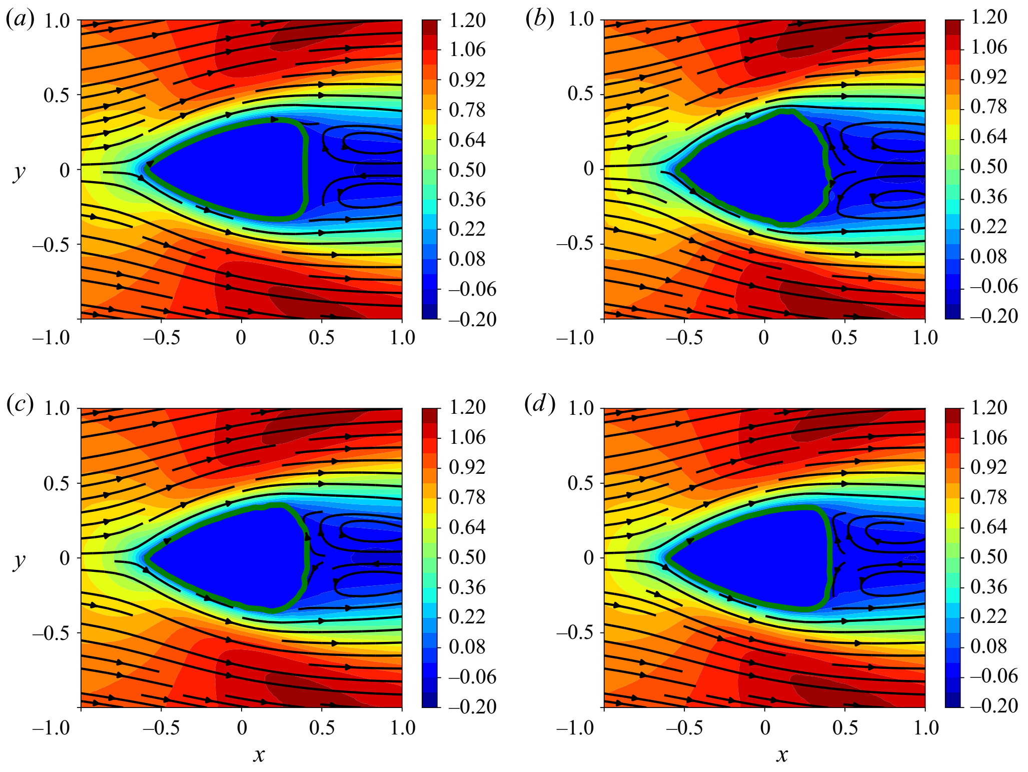

Figure 13. Streamlines and the x-component velocity fields u/U_∞ at $Re_D = 40$ obtained with different solvers, i.e. OpenFOAM and three neural network models trained with Dataset-40. (a) OpenFOAM, (b) small-scale neural network, (c) medium-scale neural network and (d) large-scale neural network.

4.3. *Shape optimisations for an enlarged solution space*

The generalising capabilities of neural networks are a challenging topic (Ling, Kurzawski & Templeton 2016). To evaluate their flexibility in our context, we target shape optimisations in the continuous range of Reynolds numbers from $Re_D = 1$ to 40, over the course of which the flow patterns change significantly (Tritton 1959; Sen, Mittal & Biswas 2009). Hence, in order to succeed, a neural network not only has to encode changes of the solutions with respect to immersed shape but also the changing physics of the different Reynolds numbers. In this section, we conduct four tests at $Re_D = 1$, 5, 10 and 40 with the ranged model in order to quantitatively assess its ability to make accurate flow-field predictions over the chosen change of Reynolds numbers. The corresponding OpenFOAM runs are used as ground truth for comparisons.

The optimisation histories for the four cases are plotted in figure 14. Despite some oscillations, the predicted drag values as well as the inviscid and viscous parts agree well with the ground truth values from OpenFOAM. The total drag force as objective function has been reduced and reaches a stable state in each case. The performance of the ranged model at $Re_D = 40$ is reasonably good, although it is slightly outperformed by the specialised neural network model trained with Dataset-40.

In line with the previous runs, the overall trend of optimisation for the four cases shows that the viscous drag increases while the inviscid part decreases as shown in figure 14, which is associated with an elongation of the profile and the formation of a sharp

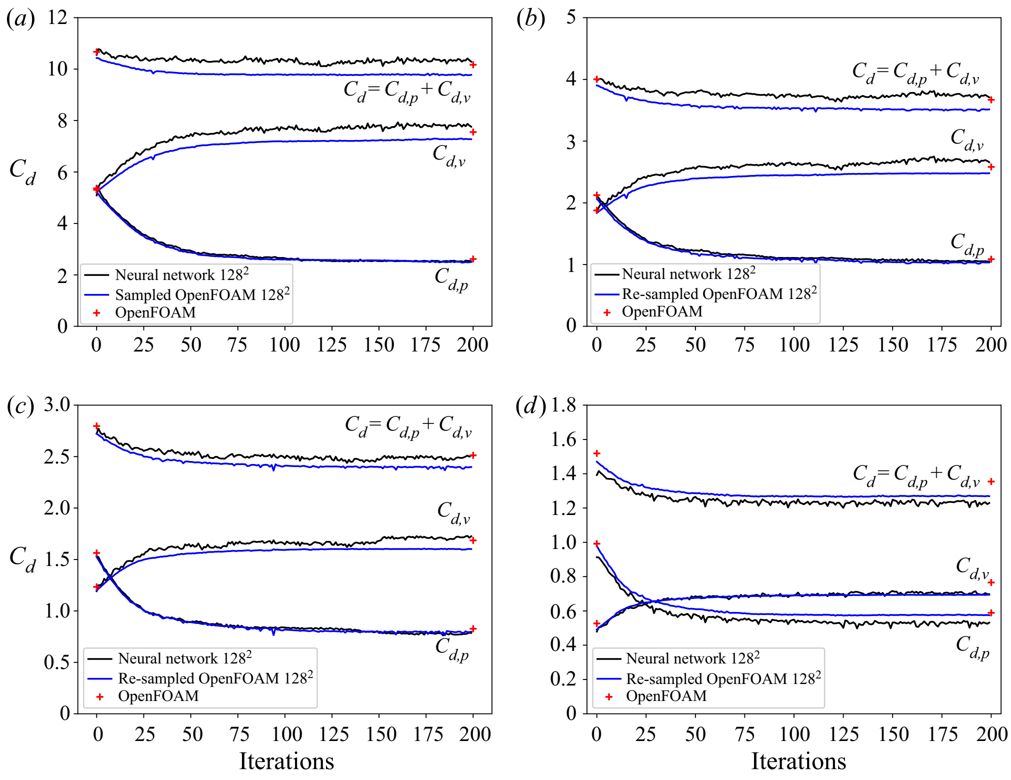

Figure 14. Optimisation history for the four cases at $Re_D = 1$, 5, 10 and 40. The black solid lines denote the results using neural network models (i.e. the ranged model) and the blue solid lines denote the results from OpenFOAM. Results calculated with the re-sampled flow fields on the 128×128 Cartesian grid are denoted by 128^2. The red cross symbols represent the OpenFOAM results obtained with its native postprocessing tool. (*a*) $Re_D = 1$, (*b*) $Re_D = 5$, (*c*) $Re_D = 10$ and (*d*) $Re_D = 40$.

leading edge. The final shapes after optimisation for the four Reynolds numbers are summarised in figure 15. For the four cases, there eventually develops a sharp leading edge, while the trailing edge shows a difference. At $Re_D = 1$ and 5, the profiles converge with sharp trailing edges as depicted in figures 15(*a*) and 15(*b*). The corresponding flow fields also show no separations in figures 16(*a*) and 16(*b*).

As shown in figure 15(*c*) at $Re_D = 10$ and figure 15(*d*) at $Re_D = 40$, blunt trailing edges become the final shapes and the profile for $Re_D = 10$ is more slender than that for $Re_D = 40$. The higher Reynolds number leads to a flattened trailing edge, associated with the occurrence of the recirculation region shown in figures 16(*c*) and 16(*d*), and the gradient of the objective function becoming relatively weak in these regions. In terms of accuracy, the converged shapes at $Re_D = 1$, 5 and 10 compare favourably with the results from OpenFOAM. Compared with ground truth shapes, only the final profile at $Re_D = 40$ predicted by the ranged model shows slight deviations near the trailing edge. Thus, given the non-trivial changes of flow behaviour across the targeted range or Reynolds numbers, the neural network yields a robust and consistent performance.

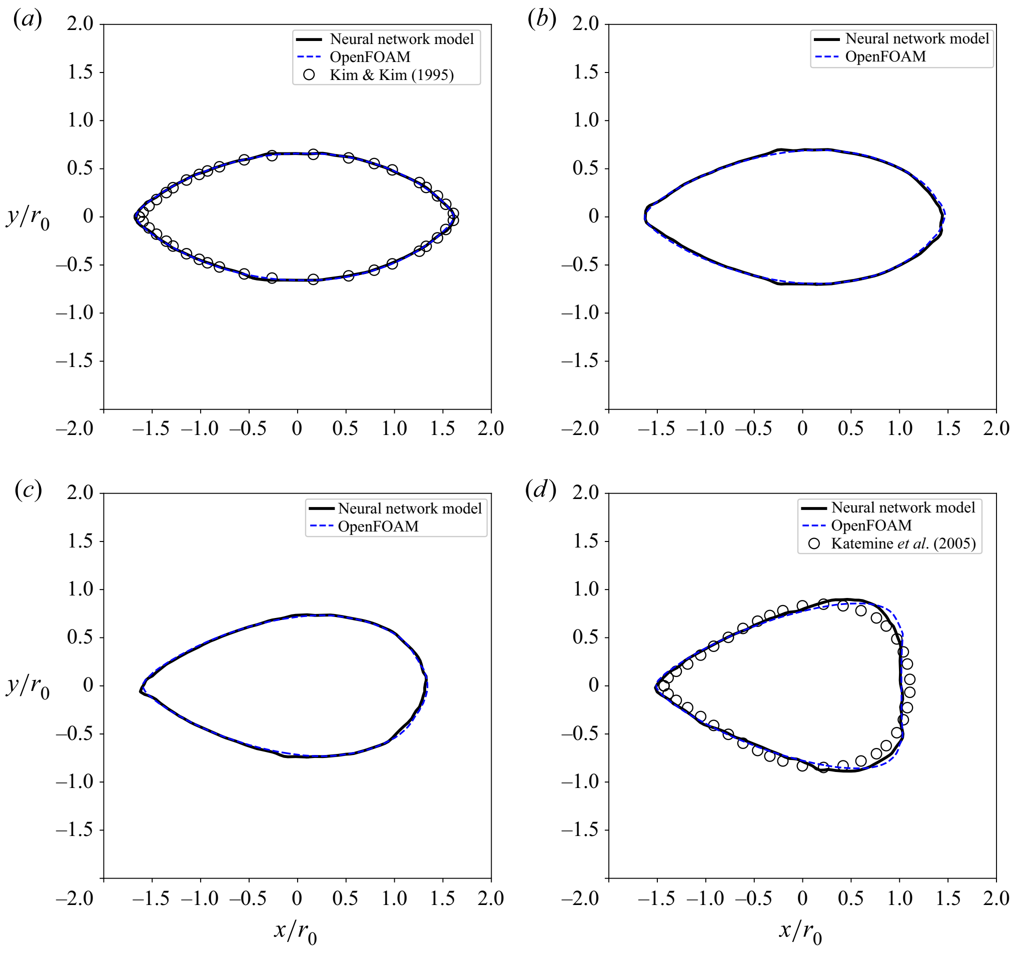

Figure 15. Shapes after optimisation at $Re_D = 1$, 5, 10 and 40. The black solid lines denote the results using neural network models (i.e. the ranged model), the blue dashed lines denote the results from OpenFOAM and the symbols denote the corresponding reference data. (*a*) $Re_D = 1$, (*b*) $Re_D = 5$, (*c*) $Re_D = 10$ and (*d*) $Re_D = 40$.

4.4. *Performance*

The performance of trained DNN models is one of the central factors motivating their use. We evaluate our models in a standard workstation with 12 cores, i.e. Intel® Xeon® W-2133 CPU at 3.60 GHz, with an NVidia GeForce RTX 2060 GPU. The optimisation run at $Re_D = 1$ which consists of 200 iterations is chosen for evaluating the runtimes using different solvers, i.e. OpenFOAM and DNN models of three sizes trained with Dataset-1. Due to the strongly differing implementations, we compare the different solvers in terms of elapsed wall clock time. As listed in table 2, it takes 16.3 h using nine cores (or 147 core-hours) for OpenFOAM to complete such a case. Compared with OpenFOAM, the DNN model using the GPU reduces the computational cost significantly. The small-scale model requires 97 s and even the large-scale model takes less than 200 s to accomplish the task. Therefore, relative to OpenFOAM, the speed-up factor is between 600 and 300 times. Even when considering a factor of approximately 10 in terms of GPU advantage due

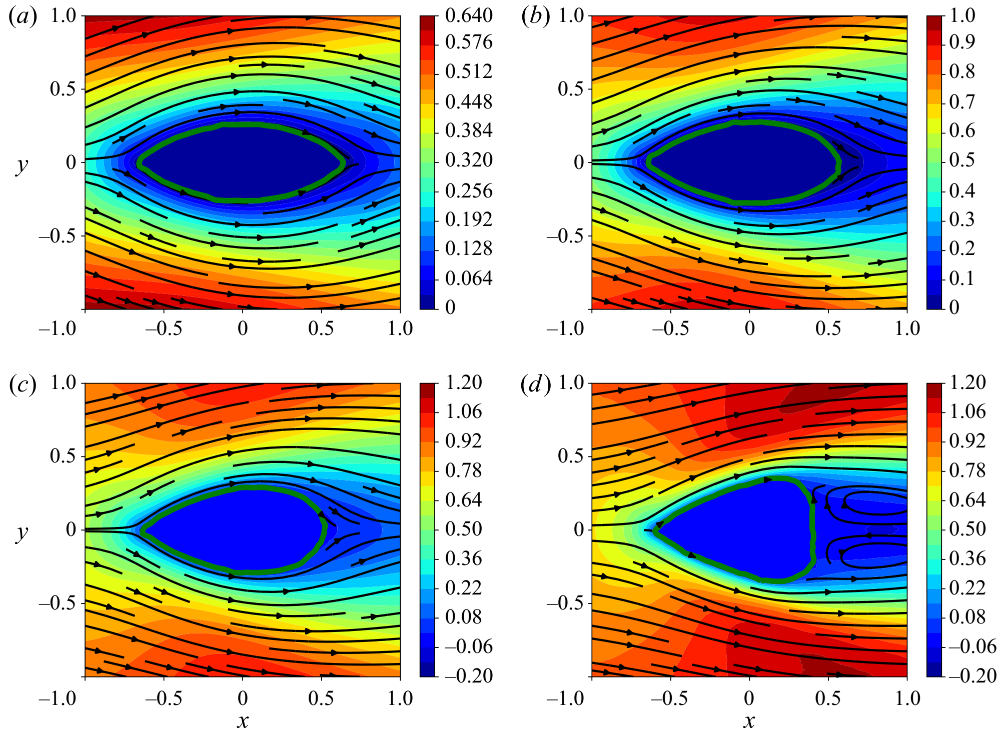

Figure 16. Streamlines and the x-component velocity fields u/U_∞ at $Re_D = 1$, 5, 10 and 40 using the ranged model. (*a*) $Re = 1$, (*b*) $Re = 5$, (*c*) $Re = 10$ and (*d*) $Re = 40$.

Solver	Wall time	Platform
OpenFOAM	16.3 h	CPU only, 9 cores
Small-scale DNN	97 s	CPU, 1 core and GPU
Medium-scale DNN	106 s	CPU, 1 core and GPU
Large-scale DNN	196 s	CPU, 1 core and GPU

Table 2. Runtimes for 200 optimisation iterations at $Re_D = 1$.

to an improved on-chip memory bandwidth, these measurements indicate the significant reductions in terms of runtime that can potentially be achieved by employing trained neural networks.

The time to train the DNN models varies with neural network size and the amount of training data. Taking Dataset-1 as an example, the time of training starts with 23 min for the small-scale model, up to 124 min for the large-scale model. When using Dataset-Range (8640 samples), it takes 252 min to train a large-scale ranged model.

Note that we aim at providing a possible choice rather than downplaying the alternatives, e.g. optimisation using discrete adjoints (Zhou *et al.* 2016). Given the potentially large one-time cost for training a model, learned approaches bear particular promise in settings where similar optimisation problems need to be solved repeatedly. Considering the cost of data generation and training, we also believe that hybrid methods that combine deep learning and traditional numerical methods represent very promising avenues for

https://doi.org/10.1017/jfm.2021.398 Published online by Cambridge University Press

future algorithms. An example would be to employ learned models as fast initialisers for nonlinear optimisations (Holl *et al.* 2020).

5. Concluding remarks

In this paper, DNNs are trained to infer flow fields, and used as surrogate models to carry out shape optimisation for drag minimisation of the flow past a profile with a given area subjected to a two-dimensional incompressible fluid at low Reynolds numbers. Both level-set and Bézier curve representations are adopted to parameterise the shape, and the integral values on the re-sampled Cartesian mesh are used as the optimisation objective. The gradient flow that drives the evolution of shape profile is calculated by automatic differentiation in a deep learning framework, which seamlessly integrates with trained neural network models.

Through optimisation, the drag values predicted by neural network models agree well with the OpenFOAM results showing consistent trends. Although the total drag decreases, it is observed that the inviscid drag decreases while the contribution of the viscous part increases, which is associated with the elongation of the shape. It is demonstrated that the present DNN model is able to predict satisfactory drag forces and the proposed optimisation framework is promising for use in general aerodynamic design. Moreover, the DNN model stands out with respect to its flexibility, as it predicts a full flow field in a region of interest and, once trained, can potentially be used in other optimisation tasks with multiple objectives. In conjunction with the low runtime of the trained DNN, we believe the proposed method showcases the possibilities of using DNNs as surrogates for optimisation problems in physical sciences.

Acknowledgements. The authors are grateful to O. Karakaya and H. Ma for valuable discussions.

Funding. This work was supported by European Research Council (ERC) grants 637014 (realFlow) and 838342 (dataFlow).

Declaration of interests. The authors report no conflict of interest.

Author ORCIDs.
Li-Wei Chen https://orcid.org/0000-0002-0309-2284;
Nils Thuerey https://orcid.org/0000-0001-6647-8910.

Appendix

To assess the effects of the number of samples in the datasets on the training and validation losses, six training runs are conducted with various amounts of data, which are listed in table 3. The method to generate those datasets is discussed in §3.2. Note that in table 3 any smaller dataset is a subset of a larger dataset and all follow the same probability distribution (see figure 4).

We found validation sets of several hundred samples to yield stable estimates, and hence use an upper limit of 400 as the maximal size of the validation dataset. The typical number of epochs, $Epoch_{max}$, for training ranges from 500 to 750.

In figure 17, all models converge to stable levels of training and validation losses, and do not exhibit overfitting despite the reduced amount of data for some of the runs. For all graphs, the onset of learning rate decay can be seen in the middle of the plot. It can be seen that with an increased number of samples, the training and validation losses follow an overall declining trend, and the variance of training losses noticeably decreases. Additionally, looking at the models trained with Dataset-Range-5815 and

Name	No. of flow fields	Training	Validation
Dataset-Range-400	400	320	80
Dataset-Range-800	800	640	160
Dataset-Range-1660	1660	1330	330
Dataset-Range-3315	3315	2915	400
Dataset-Range-5815	5815	5415	400
Dataset-Range-8640	8640	8240	400

Table 3. Different dataset sizes used for training runs with corresponding splits into training and validation sets.

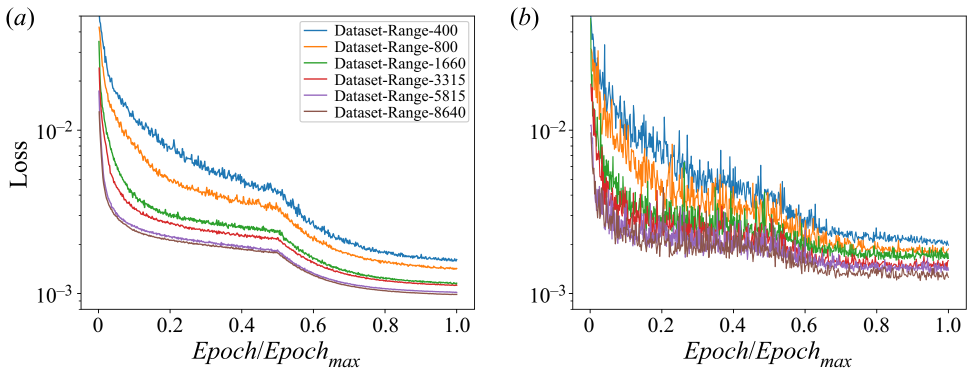

Figure 17. Comparison of training histories for different amounts of training data. (*a*) Training loss and (*b*) validation loss.

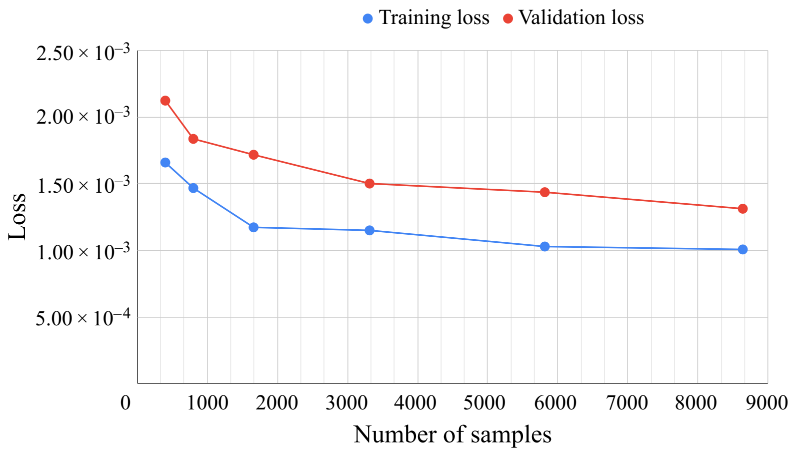

Figure 18. Training and validation losses for different amounts of training data.

Dataset-Range-8640, the two curves are very close in terms of the loss values, which implies that further increasing the amount of data does not yield significant improvements in terms of inference accuracy.

Figure 18 shows the values of training and validation losses (averaged in the last 100 epochs). It can be observed that the models with small amounts of data exhibit

https://doi.org/10.1017/jfm.2021.398 Published online by Cambridge University Press

larger losses. Both training and validation loss curves show notable drops over the course of the first data points with small amounts of data. This indicates that adding samples leads to marked improvements when the amount of data is smaller than 3000. The behaviour stabilises with larger amounts of data being available for training, especially when the number of samples is greater than 5000.

Based on the above test results, Dataset-Range-8640 is chosen for the optimisation study at the Reynolds number range $Re \in [0, 40]$. For the sake of brevity, Data-Range is used to denote this dataset in the main text.

References

de Avila Belbute-Peres, F., Economon, T. & Kolter, Z. 2020 Combining differentiable PDE solvers and graph neural networks for fluid flow prediction. In *Proceedings of the 37th International Conference on Machine Learning* (ed. H. Daumé III & A. Singh), pp. 2402–2411, vol. 119. PMLR.

de Avila Belbute-Peres, F., Smith, K., Allen, K., Tenenbaum, J. & Kolter, J.Z. 2018 End-to-end differentiable physics for learning and control. In *Advances in Neural Information Processing Systems*. (ed. S. Bengio *et al.*) vol. 31. Curran Associates, Inc.

Baeza, A., Castro, C., Palacios, F. & Zuazua, E. 2008 2D Navier–Stokes shape design using a level set method. *AIAA Paper* 2008-172.

Bhatnagar, S., Afshar, Y., Pan, S., Duraisamy, K. & Kaushik, S. 2019 Prediction of aerodynamic flow fields using convolutional neural networks. *Comput. Mech.* **64** (2), 525–545.

Bushnell, D.M. 2003 Aircraft drag reduction–a review. *J. Aerosp. Engng* **217** (1), 1–18.

Bushnell, D.M. & Moore, K.J. 1991 Drag reduction in nature. *Annu. Rev. Fluid Mech.* **23** (1), 65–79.

Chen, J., Viquerat, J. & Hachem, E. 2019 U-net architectures for fast prediction of incompressible laminar flows. arXiv:1910.13532.

Dennis, S.C. & Chang, G. 1970 Numerical solutions for steady flow past a circular cylinder at Reynolds numbers up to 100. *J. Fluid Mech.* **42**, 471–489.

Economon, T.D., Palacios, F. & Alonso, J.J. 2013 A viscous continuous adjoint approach for the design of rotating engineering applications. In *21st AIAA Computational Fluid Dynamics Conference*. American Institute of Aeronautics and Astronautics.

Economon, T.D., Palacios, F., Copeland, S.R., Lukaczyk, T.W. & Alonso, J.J. 2016 SU2: an open-source suite for multiphysics simulation and design. *AIAA J.* **54** (3), 828–846.

Eismann, S., Bartzsch, S. & Ermon, S. 2017 Shape optimization in laminar flow with a label-guided variational autoencoder. arXiv:1712.03599.

Gardner, B.A. & Selig, M.S. 2003 Airfoil design using a genetic algorithm and an inverse method. *AIAA Paper* 2003-0043.

Giles, M.B. & Pierce, N.A. 2000 An introduction to the adjoint approach to design. *Flow Turbul. Combust.* **65**, 393–415.

Glowinski, R. & Pironneau, O. 1975 On the numerical computation of the minimum-drag profile in laminar flow. *J. Fluid Mech.* **72**, 385–389.

Glowinski, R. & Pironneau, O. 1976 Towards the computation of minimum drag profiles in viscous laminar flow. *Z. Angew. Math. Model.* **1**, 58–66.

He, L., Kao, C.-Y. & Osher, S. 2007 Incorporating topological derivatives into shape derivatives based level set methods. *Comput. Fluids* **225**, 891–909.

Holl, P., Thuerey, N. & Koltun, V. 2020 Learning to control PDEs with differentiable physics. In *International Conference on Learning Representations, 2020*. https://openreview.net/forum?id=HyeSin4FPB.

Jameson, A. 1988 Aerodynamic design via control theory. *J. Sci. Comput.* **3**, 233–260.

Katamine, E., Azegami, H., Tsubata, T. & Itoh, S. 2005 Solution to shape optimisation problems of viscous fields. *Intl J. Comput. Fluid Dyn.* **19** (1), 45–51.

Kim, D.W. & Kim, M.U. 2005 Minimum drag shape in two dimensional viscous flow. *Intl J. Numer. Meth. Fluids* **21** (2), 93–111.

Kingma, D.P. & Ba, J. 2014 Adam: a method for stochastic optimization. arXiv:1412.6980.

Kline, H.L., Economon, T.D. & Alonso, J.J. 2016 Multi-objective optimization of a hypersonic inlet using generalized outflow boundary conditions in the continuous adjoint method. In *54th AIAA Aerospace Sciences Meeting*. *AIAA Paper* 2016-0912.

KONDOH, T., MATSUMORI, T. & KAWAMOTO, A. 2012 Drag minimization and lift maximization in laminar flows via topology optimization employing simple objective function expressions based on body force integration. *Struct. Multidiscipl. Optim.* **45**, 693–701.

KRAFT, D. 2017 Self-consistent gradient flow for shape optimization. *Optim. Meth. Softw.* **32** (4), 790–812.

LI, J., ZHANG, M., MARTINS, J.R.R.A. & SHU, C. 2020 Efficient aerodynamic shape optimization with deep-learning-based geometric filtering. *AIAA J.* **58** (10), 4243–4259.

LING, J., KURZAWSKI, A. & TEMPLETON, J. 2016 Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *J. Fluid Mech.* **807**, 155–166.

MICHELASSI, V., CHEN, L., PICHLER, R. & SANDBERG, R.D. 2015 Compressible direct numerical simulation of low-pressure turbines-part II: effect of inflow disturbances. *Trans. ASME: J. Turbomach.* **137**, 071005.

MUELLER, L. & VERSTRAETE, T. 2019 Adjoint-based multi-point and multi-objective optimization of a turbocharger radial turbine. *Intl J. Turbomach. Propul. Power* **2**, 14–30.

OSHER, S. & SETHIAN, J.A. 1988 Fronts propagating with curvature dependent speed: algorithms based on Hamilton–Jacobi formulations. *J. Comput. Phys.* **79**, 12–49.

PASZKE, A., *et al.* 2019 PyTorch: an imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32* (ed. H. Wallach *et al.*), pp. 8024–8035. Curran Associates, Inc.

PATANKAR, S.V. & SPALDING, D.B. 1983 A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. In *Numerical Prediction of Flow, Heat Transfer, Turbulence and Combustion*, pp. 54–73. Elsevier.

PIRONNEAU, O. 1973 On optimum profiles in Stokes flow. *J. Fluid Mech.* **59**, 117–128.

PIRONNEAU, O. 1974 On optimum design in fluid mechanics. *J. Fluid Mech.* **64**, 97–110.

QUEIPO, N.V., HAFTKA, R.T., SHYY, W., GOEL, T., VAIDYANATHAN, R. & KEVIN TUCKER, P. 2005 Surrogate-based analysis and optimization. *Prog. Aerosp. Sci.* **41** (1), 1–28.

RENGANATHAN, S.A., MAULIK, R. & AHUJA, J. 2020 Enhanced data efficiency using deep neural networks and gaussian processes for aerodynamic design optimization. arXiv:2008.06731.

RONNEBERGER, O., FISCHER, P. & BROX, T. 2015 U-net: convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015* (ed. N. Navab, J. Hornegger, W.M. Wells & A.F. Frangi), pp. 234–241. Springer International Publishing.

SEN, S., MITTAL, S. & BISWAS, G. 2009 Steady separated flow past a circular cylinder at low Reynolds numbers. *J. Fluid Mech.* **620**, 89–119.

SETHIAN, J.A. 1999*a* *Computational Methods for Fluid Flow*, 2nd edn, chap. 16, 17. Cambridge University Press.

SETHIAN, J.A. 1999*b* Fast marching methods. *SIAM Rev.* **41**, 199–235.

SETHIAN, J.A. & SMEREKA, P. 2003 Level set methods for fluid interface. *Annu. Rev. Fluid Mech.* **35**, 341–372.

SKINNER, S.N. & ZARE-BEHTASH, H. 2018 State-of-the-art in aerodynamic shape optimisation methods. *Appl. Softw. Comput.* **62**, 933–962.

SUN, G. & WANG, S. 2019 A review of the artificial neural network surrogate modeling in aerodynamic design. *J. Aerosp. Engng* **233** (16), 5863–5872.

THÉVENIN, D. & JANIGA, G. (Ed.) 2008 *Optimization and Computational Fluid Dynamics*. Springer-Verlag.

THUEREY, N., WEISSENOW, K., PRANTL, L. & HU, X. 2018 Deep learning methods for Reynolds-averaged Navier–Stokes simulations of airfoil flows. arXiv:1810.08217.

TRITTON, D.J. 1959 Experiments on the flow past a circular cylinder at low Reynolds numbers. *J. Fluid Mech.* **6** (4), 547–567.

UM, K., BRAND, R., HOLL, P., FEI, R. & THUEREY, N. 2020 Solver-in-the-loop: learning from differentiable physics to interact with iterative PDE-solvers. In *Advances in Neural Information Processing Systems* (ed. H. Larochelle *et al.*), pp. 6111–6122. vol. 33. Curran Associates, Inc.

VERSTEEG, H.K. & MALALASEKERA, W. 2007 *An Introduction to Computational Fluid Dynamics*, 2nd edn, chap. 6. Pearson Education Limited.

VIQUERAT, J. & HACHEM, E. 2019 A supervised neural network for drag prediction of arbitrary 2D shapes in low Reynolds number flows. arXiv:1907.05090.

YANG, F., YUE, Z., LI, L. & YANG, W. 2018 A new curvature-controlled stacking-line method for optimization design of compressor cascade considering surface smoothness. *J. Aerosp. Engng* **232**, 459–471.

YONDO, R., ANDRÉS, E. & VALERO, E. 2018 A review on design of experiments and surrogate models in aircraft real-time and many-query aerodynamic analyses. *Prog. Aerosp. Sci.* **96**, 23–61.

ZAHEDI, S. & TORNBERG, A.-K. 2010 Delta function approximations in level set methods by distance function extension. *J. Comput. Phys.* **229**, 2199–2219.

Zhang, X., Qiang, X., Teng, J. & Yu, W. 2020 A new curvature-controlled stacking-line method for optimization design of compressor cascade considering surface smoothness. *J. Aerosp. Engng* **234**, 1061–1074.

Zhou, B.Y., Albring, T., Gauger, N.R., da Silva, C.R.I., Economon, T.D. & Alonso, J.J. 2016 An efficient unsteady aerodynamic and aeroacoustic design framework using discrete adjoint. *AIAA Paper* 2016-3369.

Zingg, D.W., Nemec, M. & Pulliam, T.H. 2008 A comparative evaluation of genetic and gradient-based algorithms applied to aerodynamic optimization. *Eur. J. Comput. Mech.* **17** (1–2), 103–126.

https://doi.org/10.1017/jfm.2021.398 Published online by Cambridge University Press