



A Machine Learning Approach to Predict Near-optimal Meshes for Turbulent Compressible Flow Simulations

Sergi Sanchez-Gamero, Oubay Hassan & Ruben Sevilla

To cite this article: Sergi Sanchez-Gamero, Oubay Hassan & Ruben Sevilla (2024) A Machine Learning Approach to Predict Near-optimal Meshes for Turbulent Compressible Flow Simulations, International Journal of Computational Fluid Dynamics, 38:2-3, 221-245, DOI: 10.1080/10618562.2024.2306941

To link to this article: <https://doi.org/10.1080/10618562.2024.2306941>



© 2024 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group.



Published online: 29 Jan 2025.



Submit your article to this journal [↗](#)



Article views: 2



View related articles [↗](#)



View Crossmark data [↗](#)

A Machine Learning Approach to Predict Near-optimal Meshes for Turbulent Compressible Flow Simulations

Sergi Sanchez-Gamero, Oubay Hassan and Ruben Sevilla

Zienkiewicz Centre for Computational Engineering, Faculty of Science and Engineering, Swansea University, Swansea, Wales, UK

ABSTRACT

This work presents a methodology to predict a near-optimal spacing function, which defines the element sizes, suitable to perform steady RANS turbulent viscous flow simulations. The strategy aims at utilising existing high fidelity simulations to compute a target spacing function and train an artificial neural network (ANN) to predict the spacing function for new simulations, either unseen operating conditions or unseen geometric configurations. Several challenges induced by the use of highly stretched elements are addressed. The final goal is to substantially reduce the time and human expertise that is nowadays required to produce suitable meshes for simulations. Numerical examples involving turbulent compressible flows in two dimensions are used to demonstrate the ability of the trained ANN to predict a suitable spacing function. The influence of the NN architecture and the size of the training dataset are discussed. Finally, the suitability of the predicted meshes to perform simulations is investigated.

ARTICLE HISTORY

Received 3 October 2023
Accepted 12 January 2024

KEYWORDS

Mesh generation; spacing function; machine learning; artificial neural network; turbulent compressible viscous flow

1. Introduction

Generating a mesh that represents the geometry of a domain is a requirement for the majority of numerical methods used to solve partial differential equations. Fully automatic unstructured mesh generators are nowadays common and available in many commercial, research and open-source codes. However, generating tailored meshes for specific applications such as computational fluid dynamics (CFD) is still a time consuming task that requires substantial human intervention and expertise (Dawes et al. 2001; Slotnick et al. 2014). This is mainly due to the need to generate tailored meshes for different geometric configurations and/or operating conditions during a design optimisation process. An obvious alternative to alleviate the need for excessive human interaction is to produce a highly refined mesh that is capable of capturing the solution features for all the configurations to be tested (Lock et al. 2023b; Michal 2019). This removes the cost associated with human intervention, but leads to undesired large execution times. In addition, it leads to substantially larger carbon emissions associated to the simulation process, as discussed in Lock et al. (2023b).

Automatic mesh adaptive algorithms have gained popularity in the last decades (George et al. 2017). The user only needs to generate an initial mesh and an iterative process automatically adapts the mesh based on an error estimation. Depending on the error estimator used, the initial mesh requires a level of pre-adaptation to be performed by a human, because if the initial mesh fails to capture some features of the solution, the error estimator driving the adaptive process will be unable to correctly identify the areas where refinement is necessary. Furthermore, error indicators that are based on a single entity will fail to capture many phenomena. Hence, to ensure sensing all flow features, multiple indicators that uses the primitive variables and derived variables are required to enable capturing features such as shock waves, contact discontinuity, flow separation and turbulence-related quantities. In addition, it has been shown that, for complex three dimensional simulations, the complete adaptation process can involve between 20 and 30 iterations (Loseille, Dervieux, and Alauzet 2010). Each iteration requires the computation of a solution, the estimation of the error, the generation of an adapted mesh and the interpolation of the solution from the original to the adapted mesh.

CONTACT Ruben Sevilla  r.sevilla@swansea.ac.uk

Machine learning (ML) techniques for simulating physical phenomena have gained significant popularity within the computational engineering community in the last years. However, there is a fundamental concern about the potential uncertainty of predictions which, for some applications, cannot be tolerated. Instead of predicting the physical phenomena directly, ML approaches are nowadays being increasingly applied for predicting near-optimal meshes for simulations. This is motivated by two main facts. First, mesh generation is still referred to as one of the main bottlenecks in the simulation pipeline and, second, the uncertainty associated to ML predictions is tolerated when the output is a mesh that can be further improved before obtaining a high fidelity simulation.

The earliest use of ML to aid the mesh generation stage is attributed to Chang-Hoi et al. (1991). Self-organising maps were used to train an artificial neural network (ANN) capable of predicting the position of the interior mesh nodes, given the boundary discretisation and a desired mesh density function. During the 1990s many approaches followed this work by proposing several enhancements (Chedid and Najjar 1996; Dyck, Lowther, and McFee 1992; Lowther and Dyck 1993; Manevitz, Yousef, and Givoli 1997). Most of the work in this decade focussed on two dimensional problems with simple geometries and in the context of electromagnetic simulations. Further work during the second half of the 1990s and the early 2000s introduced the use of let-it-grow or adaptive ANNs to solve some of the limitations of previous works based on self-organising maps (Alfonzetti 1998; Alfonzetti et al. 1996; Fritzsche 1994; Martinetz and Schulten 1994; Triantafyllidis and Labridis 2002) and the first three dimensional results started to emerge (Alfonzetti 2003).

In the last two decades the work in the area of ML for mesh generation has gained momentum, with numerous approaches to aid the mesh generation stage. In Zhang et al. (2020) and Zhang, Jimack, and Wang (2021) the authors proposed employing ANNs to predict the spacing at a certain location for a given set of parameters associated to a partial differential equation, geometric parameters and boundary conditions. Another approach for predicting a suitable spacing for CFD applications was recently proposed in Huang et al. (2021). This work produces highly accurate CFD data using adaptively refined meshes. The spacing from the adaptive computation is

transferred to a Cartesian grid that covers the domain of interest and translated into a grey-scale image that is used to train an ANN. These approaches are related to the current work in the sense that they involve the prediction of a spacing function that is suitable for generating meshes suitable for unseen cases.

Other works have also presented the potential of using ANNs to assist mesh adaptivity or degree adaptivity strategies (Bohn and Feischl 2021; Foucart, Charous, and Lermusiaux 2023; Manevitz, Bitar, and Givoli 2005; Tlaes et al. 2022; Wallwork et al. 2022; Yang et al. 2022), to predict near-optimal spacing for complex three dimensional CFD applications (Lock et al. 2023a, 2023b) or to predict the mesh anisotropy (Fidkowski and Chen 2021). A complete review, up to the first half of 2022, can be found in Lei et al. (2023).

This work presents a new approach to predict the near-optimal spacing for steady RANS turbulent compressible flow simulations involving unseen geometric configurations or operating flow conditions. The proposed approach follows the rationale introduced in Lock et al. (2023a, 2023b) and presents solutions to the challenges that appear when considering highly stretched elements, as required to capture boundary and shear layers, and when working with more than one key variable to determine the desired spacing. In addition, for problems involving variable geometric configurations, a strategy is presented that is able to provide a tight coupling with the computer aided design (CAD) model by considering the control points of the non-uniform rational B-splines (NURBS) curves as the inputs of the ANN.

The proposed methodology consists of the following stages. First, with a computed solution, a strategy to compute the so-called *target spacing* is presented. The target spacing, defined as the spacing function that will induce a mesh capable of capturing all the features present in a solution, is obtained by using standard concepts of error analysis. The strategy is based on the computation of the eigenvalues of the Hessian matrix of a selected key variable. Second, the target spacing is transferred to a background coarse mesh. This is done to provide a discrete representation of the spacing with the same number of values for any solution, even if the meshes used to perform different simulations have different number of nodes and/or elements. When problems with variable geometry are considered, mesh morphing of the background mesh is

performed using a simple elasticity analogy. The third step consists of training a feed-forward ANN to predict the spacing at the nodes of the background mesh for a given set of inputs, which could be operating conditions or geometric parameters. Finally, the trained ANN can be used to predict a near-optimal spacing function suitable for new simulations.

The trained ANN can be used to produce a near-optimal mesh in the sense that is expected to be a very close representation of the mesh that an adaptive process would provide. As such, the predicted meshes can be integrated into an adaptive process and it is anticipated that the number of adaptive iterations will be significantly reduced when compared to starting the process with a very coarse mesh. This approach has already proved to be beneficial when compared to the current industrial practice of generating a single refined mesh to perform a range of simulations for different flow conditions (Lock et al. 2023b). It has also the potential to substantially accelerate optimisation and inverse design processes because in such scenarios it is often required to evaluate the objective functions, i.e. to run a CFD simulation, hundreds or thousands of times (Balla et al. 2022). In this context, manually generating a mesh for each configurations is clearly unfeasible and having a tool to predict a near-optimal mesh is particularly beneficial. It is worth remarking that the proposed approach also provides a path to utilise the vast amount of data that is available in industry to inform mesh generation.

The remainder of the paper is organised as follows. Section 2 briefly recalls the concepts necessary to introduce the proposed methodology. These include the governing partial differential equations for a compressible turbulent viscous flow, NURBS curves, the definition of the spacing function using a background mesh, the use of an elastic analogy to morph an existing mesh and feed-forward ANNs. In Section 3 the main contribution of the work is presented. First, the computation of the target spacing for a given simulation is presented. Three important aspects discussed in detail are the computation of the Hessian of a key variable, the choice of the key variables for compressible turbulent viscous flows and the challenges introduced by the use of highly stretched elements. A smoothing procedure is proposed to avoid that small variations in the pressure within the inflation layer having a major effect on the computed target spacing. This section also presents the strategy utilised to transfer

the spacing from a given mesh to a coarse background mesh. Particular attention is paid to the transfer of the spacing computed with two different key variables. In Section 4 two numerical examples are considered. The first example involves the prediction of near-optimal meshes for variable flow conditions, whereas in the second example the case of varying geometries is considered. The parameters for the second example are the position of the control points of the NURBS curves defining an aerofoil geometry and the example considers 23 geometric parameters. This Section discusses the effect of using different ANN architectures as well as the effect of using an increasing number of cases in the training dataset. The predicted meshes are compared to the target meshes and also utilised to perform simulations in order to assess the suitability of the predicted meshes. Finally, Section 5 summarises the conclusions of the work that has been presented.

2. Background

This Section summarises the fundamental concepts that are required to present the proposed approach to predict the near-optimal spacing suitable to generate meshes for unseen simulations.

2.1. Governing Equations

This work considers problems governed by the compressible Reynolds-averaged Navier-Stokes (RANS) equations coupled with the Spalart-Allmaras (SA) turbulence model (Spalart and Allmaras 1992). The non-dimensional conservative form of the RANS equations, without the turbulence equations, can be written as

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot (\mathbf{F}(\mathbf{U}) - \mathbf{G}(\mathbf{U}, \nabla \mathbf{U})) = \mathbf{0}, \quad (1)$$

where \mathbf{U} is the vector of conserved quantities, \mathbf{F} is the advection flux tensor and \mathbf{G} is the diffusion flux tensor, given by

$$\mathbf{U} = \begin{Bmatrix} \rho \\ \rho \mathbf{v} \\ \rho E \end{Bmatrix} \quad \mathbf{F}(\mathbf{U}) = \begin{bmatrix} \rho \mathbf{v}^T \\ \rho \mathbf{v} \otimes \mathbf{v} + p \mathbf{I} \\ (\rho E + p) \mathbf{v}^T \end{bmatrix},$$

$$\mathbf{G}(\mathbf{U}, \nabla \mathbf{U}) = \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\sigma} \\ (\boldsymbol{\sigma} \mathbf{v} + \mathbf{q})^T \end{bmatrix}. \quad (2)$$

In the above expressions, ρ denotes the density, \mathbf{v} the velocity vector, E the total specific energy, p the pressure and \mathbf{I} the identity matrix. The viscous stress tensor $\boldsymbol{\sigma}$ and the heat flux vector \mathbf{q} are given by

$$\begin{aligned}\boldsymbol{\sigma} &= \frac{\mu + \mu_t}{Re_\infty} \left(\nabla \mathbf{v} + \nabla \mathbf{v}^T - \frac{2}{3} (\nabla \cdot \mathbf{v}) \mathbf{I} \right), \\ \mathbf{q} &= \frac{1}{Re_\infty} \left(\frac{\mu}{Pr_\infty} + \frac{\mu_t}{Pr_t} \right) \nabla T,\end{aligned}\quad (3)$$

where μ is the viscosity, T the temperature, Re_∞ is the Reynolds number and Pr_∞ and Pr_t are the laminar and turbulent Prandtl numbers, respectively.

It is assumed that the viscosity varies with temperature according to the Sutherland's law, i.e. $\mu = (T/T_\infty)^{3/2} (T_\infty + S)/(T + S)$, where the free-stream temperature is $T_\infty = 1/((\gamma - 1)M_\infty^2)$ and the Sutherland constant is $S = S_0/((\gamma - 1)T_0M_\infty^2)$, with $S_0 = 110$ K and $T_0 = 273$ K. Assuming the fluid to be calorically perfect, the pressure is $p = (\gamma - 1)\rho(E - \|\mathbf{v}\|^2/2)$, where $\gamma = c_p/c_v$ (equal to 1.4 for air in standard conditions) is the ratio of specific heats at constant pressure, c_p , and constant volume, c_v .

The flow is typically characterised by the Prandtl, Reynolds and Mach numbers, given by

$$Pr_\infty = \frac{c_p \mu_\infty}{\kappa}, \quad Re_\infty = \frac{\rho_\infty v_\infty L}{\mu_\infty}, \quad M_\infty = \frac{v_\infty}{c_\infty}.\quad (4)$$

where $c = \sqrt{\gamma p/\rho}$ is the speed of sound, L a characteristic length and κ the thermal conductivity.

In this work the eddy viscosity μ_t appearing in Equation (3) is calculated from the SA turbulence model (Spalart and Allmaras 1992), which is described in detail in many references and it is therefore not repeated here.

It is worth noting that only steady cases are considered in this work. Therefore, the time t is actually an artificial pseudo-time that is simply utilised to facilitate convergence towards the steady state.

2.2. Non-uniform Rational B-splines

NURBS are commonly used in CAD to provide a boundary representation of the computational domain (Piegl and Tiller 1996). This work considers well known aerofoils such as the RAE2828 for problems with fixed geometry and NURBS-shaped aerofoils for problems with variable geometry.

A NURBS curve of degree q is a piecewise rational function defined in parametric form as

$$\mathbf{C}(\lambda) = \left(\sum_{i=0}^{n_{cp}} v_i \mathbf{B}_i C_i^q(\lambda) \right) / \left(\sum_{i=0}^{n_{cp}} v_i C_i^q(\lambda) \right) \quad 0 \leq \lambda \leq 1,$$

where the $n_{cp} + 1$ control points are denoted by $\{\mathbf{B}_i\}$ and the corresponding control weights are $\{v_i\}$. The normalised B-spline basis functions, $\{C_i^q(\lambda)\}$, are defined using the recursion

$$\begin{aligned}C_i^0(\lambda) &= \begin{cases} 1 & \text{if } \lambda \in [\lambda_i, \lambda_{i+1}[\\ 0 & \text{elsewhere,} \end{cases} \\ C_i^k(\lambda) &= \frac{\lambda - \lambda_i}{\lambda_{i+k} - \lambda_i} C_i^{k-1}(\lambda) + \frac{\lambda_{i+k+1} - \lambda}{\lambda_{i+k+1} - \lambda_{i+1}} C_{i+1}^{k-1}(\lambda),\end{aligned}$$

for $k = 1 \dots q$, where λ_i , for $i = 0, \dots, n_k$, are the knots, where it is assumed that they are ordered $0 \leq \lambda_i \leq \lambda_{i+1} \leq 1$. The knots are commonly arranged into the so-called knot vector

$$\mathcal{N} = \underbrace{\{0, \dots, 0\}}_{q+1}, \lambda_{q+1}, \dots, \lambda_{n_k - q - 1}, \underbrace{\{1, \dots, 1\}}_{q+1},$$

which uniquely defines the B-spline basis functions. The number of control points, $n_{cp} + 1$, the number of knots, $n_k + 1$, and the degree, q , are related using the expression $n_k = n_{cp} + q + 1$.

Remark 2.1: NURBS curves possess a variety of attractive properties (Piegl and Tiller 1996), but for the purpose of this work only two of the properties are briefly recalled here. For a given NURBS curve with control points $\{\mathbf{B}_i\}_{i=0, \dots, n_{cp}}$ the end point interpolation property means that $\mathbf{C}(0) = \mathbf{B}_0$ and $\mathbf{C}(1) = \mathbf{B}_{n_{cp}}$. In addition the tangent to the initial (end) point is aligned with the two first (last) control points, namely $\mathbf{C}'(0) = \mu \mathbf{v}_{0,1}$ and $\mathbf{C}'(1) = \kappa \mathbf{v}_{n_{cp}-1, n_{cp}}$, where μ and κ are real numbers and $\mathbf{v}_{i,j}$ is the vector formed by connecting control points \mathbf{B}_i and \mathbf{B}_j . In this work these two properties are relevant when building NURBS-shaped aerofoils as they provide a simple guideline to join NURBS curves whilst ensuring \mathcal{G}^1 continuity.

2.3. Spacing Function Using a Background Mesh

The generation of suitable meshes for turbulent compressible flow simulations usually relies on a combination of human expertise and techniques to define

a user-controlled local refinement. Among all the available techniques, the use of a background mesh and/or mesh sources provide great flexibility. Previous works on machine learning for predicting near-optimal meshes for inviscid compressible flows (Lock et al. 2023a, 2023b) have shown that, despite the greater flexibility of mesh sources, the use of a background mesh leads to a faster training of the ANN and, at the same time, a higher accuracy of the predictions. This is mainly due to the reduced number of outputs of the ANN. Therefore, this work only considers the use of a background mesh to define the mesh spacing.

A background mesh, denoted by \mathcal{B}_h , is defined as a collection of disjoint elements $\{B_e\}_{e=1, \dots, n_{e1}^{\mathcal{B}}}$, where $n_{e1}^{\mathcal{B}}$ is the number of elements. A discrete nodal spacing field is then defined on the background mesh, namely $\{\delta_i^{\mathcal{B}}\}_{i=1, \dots, n_{no}^{\mathcal{B}}}$, where $n_{no}^{\mathcal{B}}$ is the number of mesh nodes. The background mesh covers the whole computational domain Ω , namely $\Omega \subseteq \cup_{e=1}^{n_{e1}^{\mathcal{B}}} B_e$, and it is used during the mesh generation stage to query the required spacing at a point $\mathbf{x} \in \Omega$. This is done by identifying the element of the background mesh, B_e , that contains the point \mathbf{x} and interpolating the spacing at \mathbf{x} from the nodal spacing values of the background element B_e .

In addition, to capture the high gradient of the velocity field in the normal direction to a wall, an inflation layer of quadrilateral elements is typically employed. A common approach involves defining the non-dimensional height of the first layer of quadrilateral elements, h_1 , based on the Reynolds number, for instance $h_1 = CRe_{\infty}^{-3/4}$, where C is a user-defined constant, and building a geometric progression to successively increase the height of the next layers. A growing factor G , which usually takes value between 1.1 and 1.5, is commonly employed to define the height of the next layer of quadrilateral elements, namely $h_i = Gh_{i-1}$ for $i = 2, \dots, n_q$, where n_q is the maximum number of layers of quadrilateral elements. In the proposed implementation, n_q can be selected to be an arbitrarily large number because the mesh generator employed is capable of creating different number of quadrilateral elements in the normal direction in different regions. The mesh generator stops creating a new layer automatically when the spacing of the layer to be generated is approximately half of the spacing given by the background mesh. If the mesh generator

does not provide such capability, the approach presented here is still valid, but a structured layer with a fixed number of elements might need to be used.

In this work, the insertion of the inflation layer is considered to be independent upon the spacing of the background mesh. Although the background mesh does not need to contain an inflation layer, it is sometimes beneficial to include such a layer to ensure that a better resolution of the spacing is provided near the obstacle. In the numerical examples, both options are considered.

2.4. Mesh Morphing

When problems with varying geometric configurations are considered, a fixed fitted background mesh cannot be used. To ensure that a background mesh with the same topology is available for each geometric configuration, a mesh morphing approach based on linear elasticity is considered in this work. The strategy follows the rationale employed to generate high-order curved meshes in Xie et al. (2013).

First, a background mesh is generated for a geometry with a pre-defined set of control points, namely $\{\mathbf{B}_i^{\mathcal{B}}\}_{i=0, \dots, n_{cp}}$ for each NURBS curve $\mathcal{C}^{\mathcal{B}}$. The parametric coordinates of each node of the background mesh on the wall boundary, \mathbf{x}^w , namely λ_i such that $\mathbf{x}_i^w = \mathcal{C}^{\mathcal{B}}(\lambda_i)$, are computed using a standard NURBS inversion algorithm (Piegl and Tiller 1996). For a different geometric configuration with NURBS boundary parametrised by \mathcal{C} , with control points $\{\mathbf{B}_i\}_{i=0, \dots, n_{cp}}$, the desired position of the background mesh nodes on the wall boundary is computed as $\hat{\mathbf{x}}_i^w = \mathcal{C}(\lambda_i)$. This means that the parametric position of background mesh nodes on the wall boundary is kept constant and only the position of the control points is altered between different geometric configurations. Finally, using the initial and desired position of the background mesh nodes on the wall boundary, a boundary value problem governed by the linear elastic equation is defined. Homogeneous Dirichlet boundary conditions are imposed on the far field and Dirichlet boundary conditions, corresponding to $\hat{\mathbf{x}}_i^w - \mathbf{x}_i^w$, are imposed on each node on the wall. The solution of the elastic problem provides the desired position of all the nodes of the background mesh for a different geometric configuration.

It is worth noting that maintaining a good quality of the deformed background mesh is not a requirement as this mesh is only used to define the spacing function, not to perform a simulation. If required, multiple steps can be employed during the deformation, to ensure the validity of the deformed mesh (Xie et al. 2013).

The computational cost of deforming a background mesh is negligible due to the use of coarse background meshes. However, if desired, it is possible to deform the mesh layer-by-layer, starting with the elements in contact with a wall boundary (Xie et al. 2013). This strategy substantially reduces the computational cost of mesh deformation and also enables to stop deforming the mesh when the deformation computed in one layer is negligible.

2.5. Feed-forward Neural Networks

A feed-forward ANN is an arrangement of neurons organised in layers, where each neuron is only connected to all the neurons of the previous and next layers. A weight is associated to each connection between two neurons and a bias, which can be seen as an extra neuron with fixed value, is assigned to each layer. The first layer contains the so-called inputs of the network, whereas the neurons of the last layer are called outputs. It is common to number the remaining layers, named hidden layers, using an index $l = 1, \dots, N_l$, where N_l is the number of hidden layers (Hagan, Demuth, and Beale 1997).

The forward propagation involves the calculation of the value associated to each neuron from the values of the neurons from the previous layer, the weights of the associated connections and an activation function. More precisely, the value of the j th neuron of the $l + 1$ layer is computed as

$$z_j^{l+1} = F^{l+1} \left(\sum_{i=1}^{N_n^l} \theta_{ij}^l z_i^l + b_j^l \right), \quad (5)$$

where F^l is the activation function of the l th hidden layer, θ_{ij}^l is the weight of the connection between the i th neuron of layer l and the j th neuron of layer $l + 1$, b^l is the bias of the l th layer and N_n^l is the number of neurons of the l th layer. Figure 1 shows a sketch of a feed-forward neural network.

For a given case, the inputs and outputs of the ANN are arranged in two vectors, $\mathbf{x} = \{x_1, \dots, x_N\}^T$ and

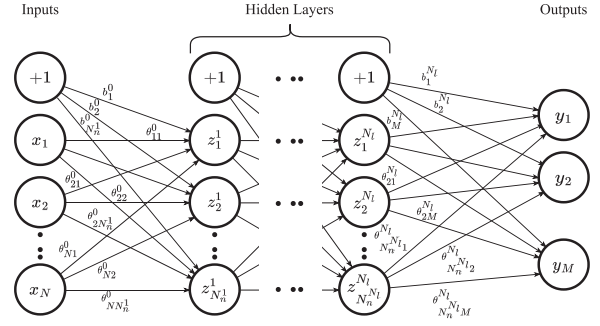


Figure 1. Sketch of a multi-layer feed-forward ANN.

$\mathbf{y} = \{y_1, \dots, y_M\}^T$, respectively, where N is the number of inputs and M is the number of outputs. The so-called cost or error function is defined as

$$E(\boldsymbol{\theta}, \mathbf{b}) = \frac{1}{N_{tr}M} \sum_{k=1}^{N_{tr}} \sum_{i=1}^M [y_i^k(\mathbf{x}) - h_i^k(\mathbf{x}, \boldsymbol{\theta}, \mathbf{b})]^2, \quad (6)$$

where $h_i^k(\mathbf{x}, \boldsymbol{\theta})$ denotes the output predicted by the ANN during the forward propagation and N_{tr} is the number of training cases. The training process consists of finding the weights $\boldsymbol{\theta}$ and biases \mathbf{b} that minimise the error function. Several approaches are available to perform this optimisation. The current implementation, based on TensorFlow 2.7.0 (Abadi et al. 2016), utilises the ADAM optimiser (Kingma and Ba 2014). The specific parameters of the optimiser considered here are the ones reported in Lock et al. (2023a, 2023b), where a similar approach was used to devise an ANN to predict near-optimal meshes for inviscid compressible flow simulations.

As usual in the context of ANNs, the hyperparameters, i.e. the number of neurons and hidden layers, can influence substantially the accuracy of the predictions. In the numerical examples considered, a simple grid search is performed in this work to select the hyperparameters. Further numerical experiments have shown that for the relative reduced datasets considered here, the rectified linear unit (ReLU) activation function produced more accurate results than other popular activation functions. Therefore, in all the numerical examples the ReLU activation function is used in all hidden layers and a linear function is employed in the output layer, namely

$$F^l(x) = \begin{cases} x & \text{if } l = N_l, \\ \max\{0, x\} & \text{otherwise.} \end{cases} \quad (7)$$

3. Near-optimal Spacing Function Prediction Using ANNs

This Section presents a novel strategy to predict a near-optimal spacing function to generate meshes suitable for performing turbulent compressible flow simulations. The strategy consists of training an ANN to predict a near-optimal discrete spacing in a background mesh. The training data is made of historical data available from previous high fidelity analysis. As an example, Figure 2 shows the pressure, Mach number and density for one simulation using a reference mesh. The geometry corresponds to the RAE2822 aerofoil and the flow conditions are $Re_\infty = 6.5 \times 10^6$, $M_\infty = 0.811$, and $\alpha = 0.27^\circ$. The objective is to predict the spacing for unseen operating flow conditions or geometric configurations.

The proposed strategy consists of the following stages:

- (1) For each available solution, a discrete target spacing function is computed on the same mesh that was used to perform the simulation.
- (2) The discrete spacing function is transferred from the computational mesh to a coarse background mesh.
- (3) A feed-forward ANN is trained to predict the discrete spacing function for the background mesh. The inputs of the ANN are flow conditions or geometric parameters, whereas the output of the ANN is a discrete spacing function, i.e. the spacing at the nodes of the background mesh.

It is worth noting that, for different simulations, the computational mesh is usually different, whereas the same background mesh is considered for all cases. When geometric parameters are considered, the mesh morphing strategy described in Section 2.4 is applied

to the background mesh to ensure that the topology of the background mesh is maintained.

3.1. Computation of the Target Spacing in the Computational Mesh

Following principles traditionally employed in error analysis (Peraire et al. 1987), the procedure to compute a discrete spacing function from a given solution, requires the evaluation of the Hessian matrix of a chosen key variable at every point of the computational mesh. More precisely, the spacing at a node \mathbf{x}_i along a generic direction given by a unit vector $\boldsymbol{\beta}$ is related to the derivatives of the key variable as

$$\delta_{i,\boldsymbol{\beta}}^2 \left(\sum_{k,l=1}^{n_{sd}} (H_i)_{kl} \beta_k \beta_l \right) = K, \quad (8)$$

where n_{sd} is the number of spatial dimensions, K is a user-defined constant and

$$(H_i)_{kl} = \frac{\partial^2 \sigma_i}{\partial x_k \partial x_l} \quad (9)$$

are the components of the Hessian matrix at node \mathbf{x}_i , \mathbf{H}_i , for the key variable σ . The evaluation of the derivatives of the selected key variable, σ , is described in detail in Section 3.1.1. After the derivatives are evaluated, the spacing at a node \mathbf{x}_i of the computational mesh is computed as

$$\delta_i = \begin{cases} \delta_{\min} & \text{if } \lambda_{i,\max} > K/\delta_{\min}^2, \\ \delta_{\max} & \text{if } \lambda_{i,\max} < K/\delta_{\max}^2, \\ \sqrt{K/\lambda_{i,\max}} & \text{otherwise,} \end{cases} \quad (10)$$

where $\{\lambda_{i,j}\}_{j=1,\dots,n_{sd}}$ are the eigenvalues of \mathbf{H}_i and $\lambda_{i,\max} = \max_{j=1,\dots,n_{sd}} \{\lambda_{i,j}\}$.

It can be seen that Equation (10) imposes that $\delta_i \in [\delta_{\min}, \delta_{\max}]$. The minimum value δ_{\min} is used to avoid an excessive local refinement in regions with very steep gradients, for instance in the vicinity of strong shocks.

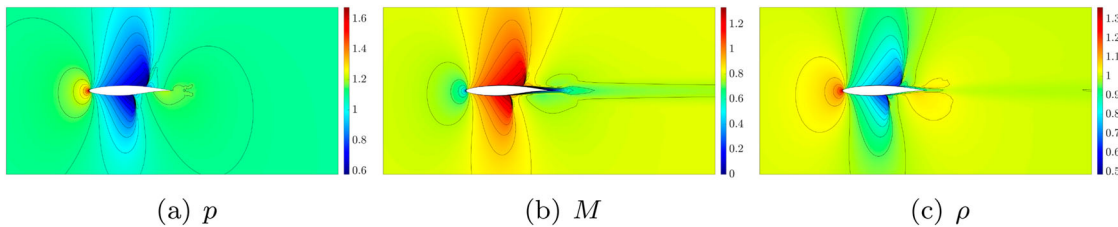


Figure 2. Solution obtained on a fine mesh for $Re_\infty = 6.5 \times 10^6$, $M_\infty = 0.811$, and $\alpha = 0.27^\circ$. (a) p . (b) M and (c) ρ .

Analogously, the maximum value δ_{\max} is used to avoid an excessive large value in regions of undisturbed flow, where the key variable is smooth or nearly constant. To simplify the implementation, this work considers the values of δ_{\min} and δ_{\max} as the minimum and maximum spacing in the computational mesh, respectively.

In the current implementation, the value of K is defined as

$$K = S^2 \delta_{\min}^2 \lambda_{\max}, \quad (11)$$

where $S \in (0, 1]$ is a user-defined scaling factor and $\lambda_{\max} = \max_{i=1, \dots, n_{\text{no}}} \{\lambda_{i, \max}\}$. For a value of $S = 1$, the minimum spacing is only assigned to the node/s \mathbf{x}_i with a maximum eigenvalue of the Hessian matrix, $\lambda_{i, \max}$, equal to the overall maximum λ_{\max} . The lower the scaling factor is, the more nodes in the mesh will be assigned the minimum spacing. This is done in practice to ensure that the refinement is not just concentrated in one region with a very steep gradient of the key variable, missing most of the other features of the solution. Figure 3 illustrates the effect of the scaling factor. The spacing is computed using the pressure field of Figure 2(a) as the key variable. Unless otherwise stated, the value of the scaling factor used in this work is $S = 0.2$.

The spacing defined by Equation (10) is a single scalar per node. Although it would be possible to compute an anisotropic metric per node, this is beyond the scope of the current work.

The next three Sections discuss the evaluation of the Hessian, the choice of the key variable, σ , and the proposed strategy to compute the spacing in the inflation layer, where the elements are quadrilaterals, with a significant stretching, designed to capture the large gradient of the velocity in the normal direction to the wall.

3.1.1. Computation of the Hessian Matrix

The components of the Hessian are computed as the derivatives of the derivatives. Therefore, it is only necessary to detail how to compute a first order derivative.

Three different strategies to compute the first derivatives of the selected key variable on an unstructured mesh have been considered. The first strategy considers a vertex-centred FV approximation, i.e. using a dual mesh, of the derivatives at the nodes (Sørensen et al. 2003) given by

$$\nabla \sigma_i \approx \frac{1}{|V_i|} \left[\sum_{j \in \mathcal{N}_i} \frac{1}{2} (\sigma_i + \sigma_j) \mathbf{C}_{ij} + \sum_{j \in \mathcal{N}_i^\partial} \sigma_i \mathbf{D}_{ij} \right] \quad (12)$$

where V_i is the the control volume associated to node \mathbf{x}_i , \mathcal{N}_i is the set of nodes connected to \mathbf{x}_i by an interior edge (i.e. an edge not on the boundary) and \mathcal{N}_i^∂ is the set of nodes connected to \mathbf{x}_i by a boundary edge. The geometric coefficient vectors \mathbf{C}_{ij} and \mathbf{D}_{ij} are defined as

$$\mathbf{C}_{ij} = \sum_{\Gamma_{ij}^k \in \mathcal{F}_{ij}} |\Gamma_{ij}^k| \mathbf{n}_i^k, \quad \mathbf{D}_{ij} = \sum_{\Gamma_{ij}^k \in \mathcal{F}_{ij}^\partial} |\Gamma_{ij}^k| \mathbf{n}_i^k, \quad (13)$$

where \mathcal{F}_{ij} and $\mathcal{F}_{ij}^\partial$ are the sets of interior and boundary facets of control volume V_i , respectively, that intersect the edge Γ_{ij} connecting nodes \mathbf{x}_i and \mathbf{x}_j , and \mathbf{n}_i^k is the unit outward normal to Γ_{ij}^k from the point of view of node \mathbf{x}_i .

The second approach considers the standard recovery process employed in the finite element method (Zienkiewicz and Zhu 1992a, 1992b). In this case the derivative at a node \mathbf{x}_i is evaluated as a weighted average of the derivative computed at the barycentre of the elements that contain node \mathbf{x}_i , namely

$$\nabla \sigma_i \approx \frac{\sum_{\Omega_e \in \mathcal{P}_i} |\Omega_e| \nabla \sigma_e}{\sum_{\Omega_e \in \mathcal{P}_i} |\Omega_e|}, \quad (14)$$

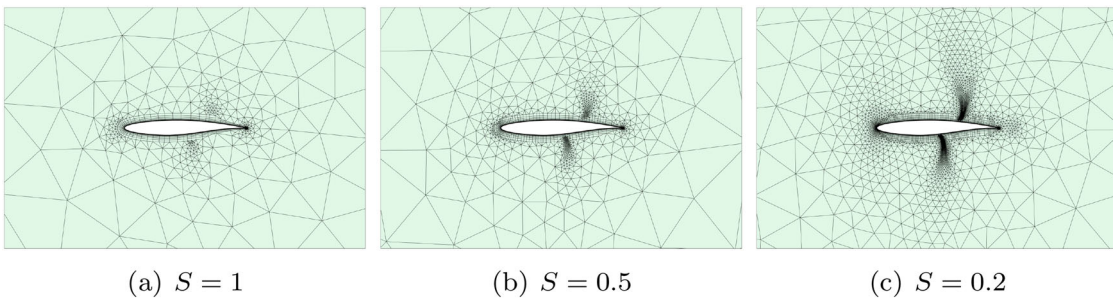


Figure 3. Meshes obtained by varying the minimum spacing using the pressure as key variable for the solution of Figure 2(a). (a) $S = 1$. (b) $S = 0.5$ and (c) $S = 0.2$.

where the gradient at the barycentre of the element, σ_e , is evaluated using the finite element shape functions of degree one and \mathcal{P}_i is the patch of elements that contain the node x_j .

The third alternative considered here employs again the finite element recovery process, but after splitting the quadrilateral elements into two triangular elements.

Figure 4 shows the meshes obtained with the three approaches described above. The results show that the spacing computed with the finite element recovery process to evaluate the Hessian matrix is slightly smaller in the regions of interest, i.e. at the leading edge and near the shocks, compared to the spacing obtained by using a finite volume approximation of the derivatives in the Hessian matrix. The finite volume approach requires a smaller scaling in order to produce a similar refinement as the one obtained with a finite element approximation of the derivatives. In this example the spacing computed after evaluating the derivatives with the finite volume formulae utilised a scaling of $S = 0.1$, whereas the finite element recovery approaches employed $S = 0.2$. To illustrate the differences between the three strategies, Figure 5 shows a detailed view of the meshes in the vicinity of the region where the shock on the upper surface interacts with

the boundary layer. As it can be observed, the finite element recovery process performed after splitting the quadrilateral elements into triangles produces a very localised refinement in the region where the shock on the upper surface interacts with the boundary layer. This refinement is not achieved by the finite volume approximation of the derivatives or the finite element recovery process on hybrid meshes. For this reason, in all the remaining examples, the finite element recovery process after splitting the quadrilateral elements into triangles is considered.

As seen in Figure 5, when the required element size near the obstacle is below the element size of the inflation layer, triangular elements are generated close to the obstacle. With the mesh generator employed, this does not pose any issue as the number of quadrilateral elements in the normal direction can vary in different regions.

3.1.2. Choice of the Key Variables

For inviscid compressible flow simulations, previous works (Lock et al. 2023a, 2023b) have shown the effectiveness of using the pressure as the only key variable to capture flow features in the vicinity of an aerodynamic object. However, for viscous compressible flows, the second derivatives of the pressure in the region of

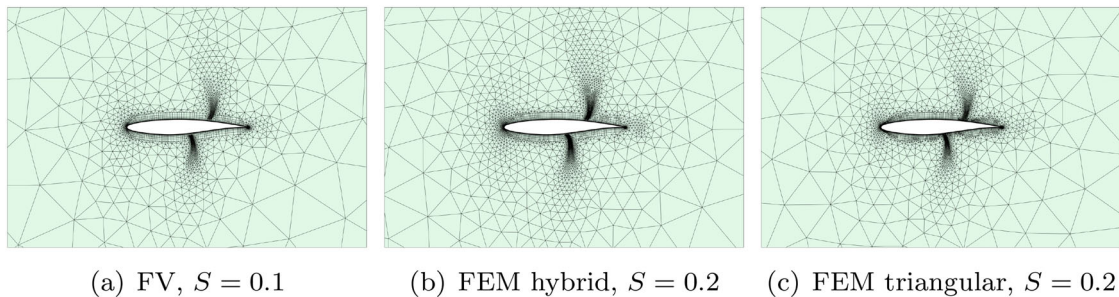


Figure 4. Meshes obtained with the minimum spacing provided by pressure as key variable and with three different approaches to evaluate the Hessian for the solution of Figure 2(a). (a) FV, $S = 0.1$. (b) FEM hybrid, $S = 0.2$ and (c) FEM triangular, $S = 0.2$.

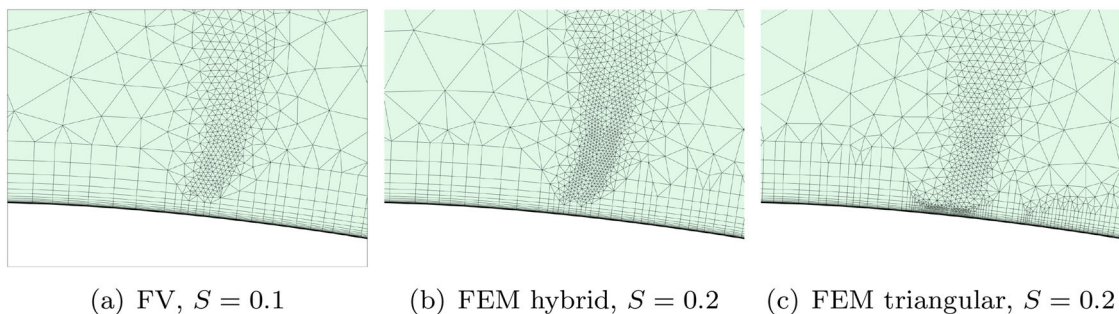


Figure 5. Detail of the three meshes of Figure 4 near the region where the shock of the upper surface interacts with the boundary layer. (a) FV, $S = 0.1$. (b) FEM hybrid, $S = 0.2$ and (c) FEM triangular, $S = 0.2$.

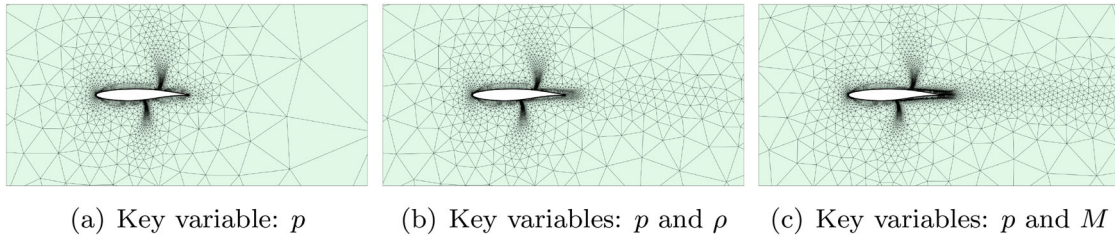


Figure 6. Meshes obtained with the minimum spacing provided by different sets of key variables for the solution of Figure 2. (a) Key variable: p . (b) Key variables: p and ρ and (c) Key variables: p and M .

the wake are not strong enough to provide relevant information. For this reason, a second key variable is considered to provide the required information that enables a local refinement of the wake. Two alternatives have been considered, namely the density and the Mach number. Figure 6 shows the meshes obtained by considering only the pressure and the pressure combined with the density and the Mach number as key variables.

As it can be observed, using the Mach number as the second key variable produces an extra refinement in the region of the wake. Further numerical experiments, not reported here for brevity confirm this observation and, therefore, for the numerical examples in this work, the pressure and the Mach number are the key variables considered. Obviously, when two key variables are employed, namely σ^1 and σ^2 , Equation (10) is used to compute the spacing at each node for both variables, namely δ_i^1 and δ_i^2 , and the minimum of the two spacings computed at each node is kept, $\delta_i = \min\{\delta_i^1, \delta_i^2\}$.

It is worth noting that in the example of Figure 6, the spacing was computed using Equation (10), but only for the nodes belonging to a triangular element. The next section details the computation of the spacing on the quadrilateral elements within the inflation layer.

3.1.3. Computation of the Target Spacing in the Inflation Layer

There are a number of difficulties that arise when computing the spacing in the inflation layer, caused by a significant variation of some quantities in the normal direction to the wall and the large stretching of the quadrilateral elements near the wall.

First, the Mach number presents a huge variation in the normal direction to the wall, due to the no-slip boundary condition. If Equation (10) is applied, the minimum spacing is completely dominated by the large values of the second derivatives of the velocity

or Mach number in the normal direction to the wall. However, as described in Section 2.3, there are effective and easy-to-implement strategies to define the spacing in the normal direction to the wall that are widely adopted and do not require any human intervention. Therefore, this work assumes that there is no need to devise a strategy to predict the spacing in the normal direction to a wall within the inflation layer. To this end, the use of the Mach number as a key variable is restricted to nodes that belong to a triangular element.

Ignoring the computation of the spacing with the Mach number as key variable in the quadrilateral elements of the inflation layer might lead to a large discrepancy between the spacing computed by the pressure and the Mach number in the interface between the unstructured mesh and inflation layer. This ultimately can lead to an unfeasible discrete spacing distribution that cannot be realised by a mesh generator. To avoid this incompatibility, the spacing function obtained by using the Mach number as key variable is extended into the inflation layer by considering a constant extension in the normal direction.

Assuming that the nodes on the wall boundary are denoted by $\mathbf{x}_{i,1}^w$ for $i = 1, \dots, n_{\text{no}}^w$, the nodes on the inflation layer that follow the normal direction to the wall from the i th node $\mathbf{x}_{i,1}^w$ are denoted by $\mathbf{x}_{i,j}^w$ for $j = 1, \dots, n_{\text{no}}^{i,w}$, as illustrated in Figure 7.

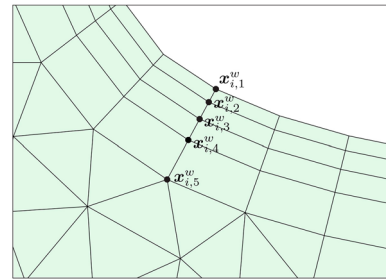


Figure 7. Sketch of a hybrid mesh showing the numbering of the nodes in the inflation layer region.

The spacing is computed at all nodes of the mesh that belong to a triangular element using Equation (10), with the Mach number as key variable. Then the spacing associated to the rest of the nodes in the inflation layer is simply given by

$$\delta(\mathbf{x}_{i,j}^w) = \delta(\mathbf{x}_{i,n_{no}^{i,w}}^w), \quad (15)$$

for $j = 1, \dots, n_{no}^{i,w} - 1$.

A second challenge to be considered when computing the spacing in the inflation layer using the pressure as key variable is that small variations in the normal direction might occur that, due to the very high stretching of the elements, can lead to extremely large derivatives. Figure 8 shows a close up view of the pressure field and contour lines for two different calculations near the leading edge, using the finite difference NASA code Wind-US (Cook, McDonald, and Firmin 1979) and the finite volume FLITE system (Sørensen et al. 2003) using the same computational mesh. Although the variation of the pressure on the area displayed is below 0.05, these small variations lead to very large derivatives when using any of the formulae presented in Section 3.1.1, due to the large stretching of the elements, with a height of the first element in the inflation layer of 9.9×10^{-6} and an area of 2.4×10^{-9} .

To ensure that these small and non-desired variations of the pressure do not skew the definition of the target spacing, a pressure smoothing is proposed in this work. Given the ordering of the nodes in the inflation layer previously described, and illustrated in Figure 7, the smoothed pressure, \tilde{p} in the normal direction to the wall at node $\mathbf{x}_{i,1}^w$ is assumed to be a cubic polynomial that satisfies the following four conditions:

- $\tilde{p}(0) = p(\mathbf{x}_{i,1}^w)$,
- $\tilde{p}'(0) = 0$,
- $\tilde{p}(d_{i,n_{no}^{i,w}-1}) = p(\mathbf{x}_{i,n_{no}^{i,w}-1}^w)$,
- $\tilde{p}'(d_{i,n_{no}^{i,w}-1}) = \frac{p(\mathbf{x}_{i,n_{no}^{i,w}}^w) - p(\mathbf{x}_{i,n_{no}^{i,w}-1}^w)}{d_{i,n_{no}^{i,w}} - d_{i,n_{no}^{i,w}-1}}$,

where $d_{ij} = \|\mathbf{x}_{i,1}^w - \mathbf{x}_{i,j}^w\|_2$.

These conditions impose the computed value of the pressure at the wall and zero normal derivative. In addition, to produce a smooth transition with the computed pressure outside the inflation layer, it imposes the value at the last node that does not belong to a triangular element and its derivative in the normal direction. The smoothing approach presented here simply aims at removing the negative influence of the undesired variations of the pressure in the spacing function computed using the Hessian. Although other, potentially more complex, smoothing approaches might be devised, for all the examples considered in this work, the cubic polynomial smoothing presented here led to satisfactory results. This strategy can be easily extended to three dimensional problems as the same type of smoothing can be defined in the normal direction to the wall.

Figure 9 shows the meshes obtained by considering the pressure and the smoothed pressure as key variable, respectively. It can be clearly observed that, without the proposed smoothing, the refinement is concentrated in the vicinity of the aerofoil, due to the gradients of the pressure in the normal direction. By smoothing the pressure, the variations in the normal direction do not play a relevant role and the spacing obtained provides the desired refinement in the critical zones such as leading and trailing edges and near the shocks.

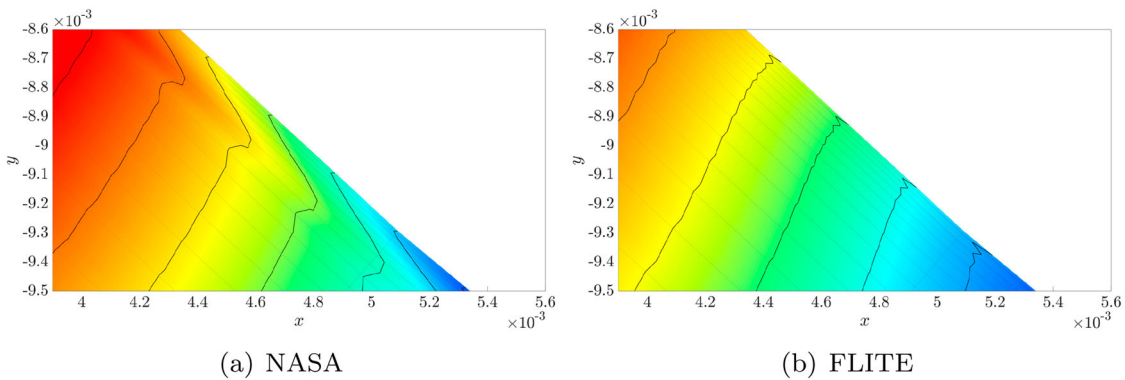


Figure 8. Close view of the pressure field and contour lines near the wall for $Re_\infty = 6.5 \times 10^6$, $M_\infty = 0.729$ and $\alpha = 2.31^\circ$ using two different solvers. (a) NASA and (b) FLITE.

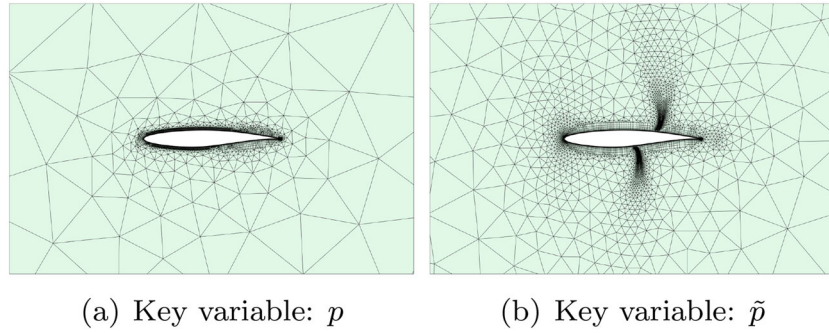


Figure 9. Meshes obtained with the minimum spacing provided by considering the pressure and the smoothed pressure as key variable. (a) Key variable: p and (b) Key variable: \tilde{p} .

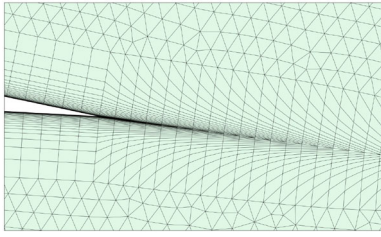


Figure 10. Detail of an unstructured mesh around an aerofoil, showing the highly stretched elements downstream of the aerofoil trailing edge.

Remark 3.1: The meshes considered for compressible turbulent flows normally present a high level of stretching, not only in the inflation layer, but also downstream of the aerofoil trailing edge, as shown in Figure 10. This is usually beneficial to capture the shear layer at low angles of attack, but it also results in very high gradients of the pressure, as in the vicinity of the wall. To avoid this region dominating the computed spacing, the smoothing of the pressure detailed in this section is also applied to the stretched elements in this region.

3.2. Computation of the Target Spacing in the Background Mesh

As the proposed strategy aims at training a feed-forward ANN to predict the target spacing, it is necessary to design an ANN architecture where the number of inputs and outputs is fixed.

For problems involving a fixed geometry and variable operating conditions, the inputs will be the flow conditions (i.e. Mach number and angle of attack), whereas for problems involving variable geometries the inputs considered in this work are the control points of the NURBS curves describing the boundary.

Considering the outputs as the spacing at each node of the computational mesh is not a viable solution for two main reasons. First, each computational mesh might have a different number of nodes. Second, even if all the computational meshes have the same number of nodes, this is expected to be a large number, because the solutions might have been computed in an over-refined mesh, leading to large ANN training times. The strategy considered here consists of defining a coarse background mesh and transferring the spacing function from the computational mesh to the background mesh. This will be done for every available training case, so that the ANN can be trained to predict the spacing at the nodes of the background mesh.

To describe the proposed approach, let us consider a background mesh \mathcal{B}_h with elements $\{B_e\}_{e=1,\dots,n_{e1}^B}$ and nodes $\{\mathbf{x}_i^B\}_{i=1,\dots,n_{no}^B}$. Similarly, let us consider a computational mesh, for one of the training cases, \mathcal{C}_H with elements $\{\Omega_e\}_{e=1,\dots,n_{e1}}$ and nodes $\{\mathbf{x}_i\}_{i=1,\dots,n_{no}}$. In addition, a nodal field, corresponding to the nodal spacing, is defined in the computational mesh, namely $\{\delta_i\}_{i=1,\dots,n_{no}}$. The objective is to transfer the discrete spacing δ from the computational mesh \mathcal{C}_H to the background mesh \mathcal{B}_h .

The strategy to transfer the spacing is adopted from Lock et al. (2023a) and it is briefly summarised here. For a given node of the background mesh \mathbf{x}_i^B , the patch of elements containing this node is denoted by \mathcal{P}_i^B . The set of nodes of the computational mesh that belong to \mathcal{P}_i^B is denoted by $\mathcal{X}_{\mathcal{P}_i}$. With this information, the spacing at the node \mathbf{x}_i^B of the background mesh is defined as

$$\delta_i^B = \min_{j \in \mathcal{X}_{\mathcal{P}_i}} \{\delta_j\}. \quad (16)$$

To illustrate the procedure, let us consider the solution computed on a reference mesh shown in Figure 11(a).

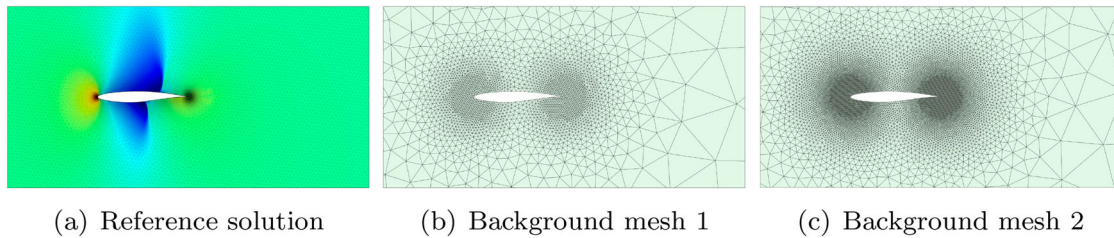


Figure 11. Solution on a reference mesh and two background meshes. (a) Reference solution. (b) Background mesh 1 and (c) Background mesh 2.

Two background meshes, with different levels of refinement are utilised, as shown in Figure 11(b,c).

First, the spacing is computed on the mesh where the reference solution is available, using the procedure described in the previous sections. The spacing computed using the smoothed pressure and the Mach number is shown in Figure 12(a). Using the procedure described in this Section, the spacing is transferred to the two background meshes of Figure 11(b,c) and depicted in Figure 12(b,c), respectively. It can be observed that transferring the spacing to a background mesh not only enables a consistent number of variables to be used in an ANN, but also introduces a desired smoothing in the discrete spacing function, when compared to the spacing function obtained in the reference mesh.

Finally, utilising the spacing functions of Figure 12, three meshes with the desired spacing are generated and displayed in Figure 13. The results show the similarity of the meshes obtained and also illustrate that for coarser background meshes the resulting

generated meshes are finer. This is due to the conservative approach utilised to transfer the spacing from the reference mesh to a background mesh, given by Equation (16). This conservative approach produces the desired outcome as it ensures that any local feature of the reference solution is captured by the spacing function defined in a coarser, background, mesh.

Remark 3.2: Two special situations need to be carefully handled when using this approach. First, when considering domains with curved boundaries, it is possible to have several nodes of the computational mesh that do not belong to any element, and therefore any patch, of the background mesh. To ensure that the spacing of such nodes is not ignored, they are associated to the closest elements of the background mesh. Second, it is possible to encounter a patch associated to a node of the background mesh that does not contain any node of the computational mesh. The spacing associated to these *isolated* nodes is taken as the minimum of the spacing of the nodes of the element of the

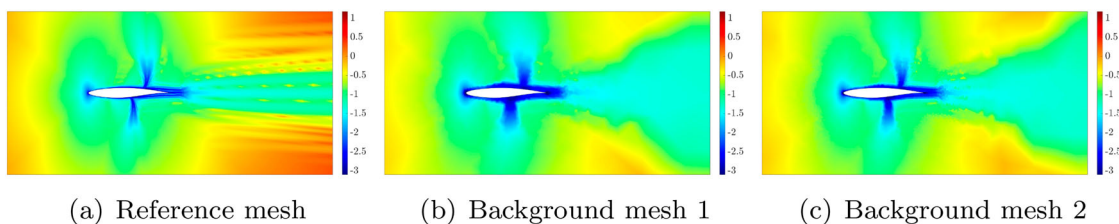


Figure 12. Spacing, in logarithmic scale, on the mesh where the solution is available and on the two background meshes of Figure 11(b,c). (a) Reference mesh. (b) Background mesh 1 and (c) Background mesh 2.

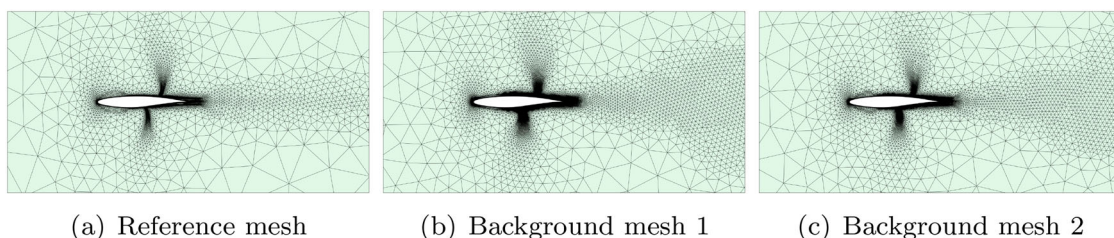


Figure 13. Meshes generated using the spacing function defined on a reference mesh and on the two background meshes of Figure 11(b,c). (a) Reference mesh. (b) Background mesh 1 and (c) Background mesh 2.

computational mesh that contains the isolated node of the background mesh.

4. Numerical Results

Two numerical studies are considered. The first involves the prediction of near-optimal meshes for variable operating conditions, namely the free-stream Mach number, M_∞ , and the angle of attack, α . The second study focuses on the prediction of near-optimal meshes for variable geometric configurations, parametrised using the control points of the NURBS that describe the geometry of an aerofoil. In all the examples, the Hessian is evaluated using the finite element recovery after splitting the quadrilateral elements into triangles, as described in Section 3.1.1 and the spacing from a given solution is computed using the smoothed pressure and the Mach number as key variables, as described in Section 3.1.3.

To perform the training and assess the accuracy, three different datasets are considered. The training set contains the cases used to optimise the weights and biases of the ANN. The validation set is used to avoid overfitting. Finally, the test set, which contains unseen cases is used, after training, to assess the accuracy of the predictions.

In all cases, the data required to train the ANN has been generated using the in-house vertex-centred finite volume solver FLITE (Sørensen et al. 2003).

4.1. Prediction of the Spacing for Variable Operating Conditions

The first study considers a fixed geometric configuration, corresponding to the RAE2822 aerofoil, and it shows the ability to predict near-optimal meshes for different flow conditions. Numerical experiments

are performed to illustrate the influence of the ANN hyperparameters and the size of the training set on the accuracy of the predictions.

The Reynolds number is fixed at $Re_\infty = 6.5 \times 10^6$ and the range of the flow parameters considered is $M_\infty \in [0.4, 0.85]$ and $\alpha \in [0^\circ, 3^\circ]$. Therefore, all the cases involve turbulent compressible flows that span over the subsonic and transonic regimes. To illustrate the variation of the solution induced by the parameters, Figure 14 shows three solutions corresponding to three different flow conditions. The first solution corresponds to a subsonic case, whereas the second and third solution display a transonic flow regime. The different solutions exhibit a completely different behaviour, one with no shocks, another with a strong shock on the upper surface of the aerofoil and another with one shock on the upper and one shock on the lower surface of the aerofoil. Clearly, the ideal mesh for each case is completely different, requiring refinement in different zones.

The data required to train the ANN has been generated using a fixed hybrid unstructured mesh with 59,397 nodes and 100,094 elements, with 18,042 elements being quadrilaterals in the inflation layer. The inflation layer has an initial element height of 5×10^{-6} and a variable growing factor, namely 1.1 for the first three layers, 1.2 for the next 17 layers and 1.45 for the last 14 layers. In addition, point sources were introduced in the leading and trailing edges with a spacing of 8×10^{-4} . A constant background spacing of 4×10^{-2} has been specified in a rectangular region of dimension $[5c, 3c]$, where c is the chord length of the aerofoil. The rationale for designing the mesh follows the current industrial practice (Michal 2019) that consists of generating an over-refined mesh to conduct parametric studies in order to avoid the time consuming process of generating a different mesh

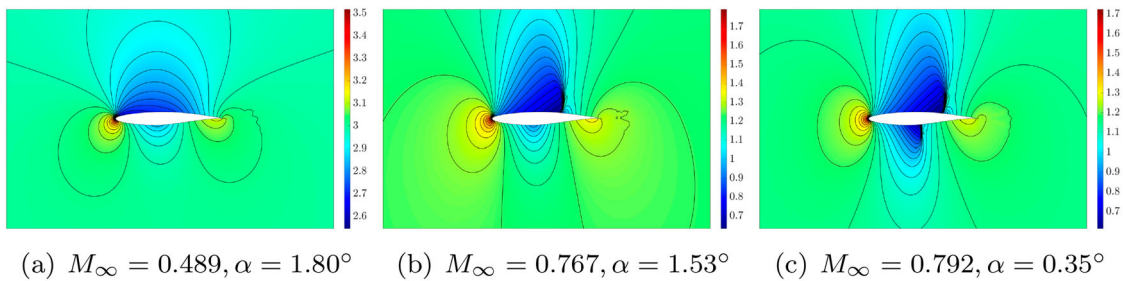


Figure 14. Pressure field and isolines for three different flow conditions with $Re_\infty = 6.5 \times 10^6$. (a) $M_\infty = 0.489, \alpha = 1.80^\circ$. (b) $M_\infty = 0.767, \alpha = 1.53^\circ$ and (c) $M_\infty = 0.792, \alpha = 0.35^\circ$.

for each simulation, which, apart from being difficult, requires a significant level of human intervention (Lock et al. 2023b).

Four training sets with $N_{tr} = 20, 40, 80, 160$ training cases have been generated using the quasi-random Halton sequencing (Cheng and Druzdzel 2000; Halton 1964). To avoid over-fitting during the training process, four validation sets with $N_{val} = 5, 10, 20, 40$ cases are also generated using Halton sequencing. Finally, to assess the accuracy of the predictions a fixed set of $N_{test} = 80$ test cases is considered, also generated using the Halton sequencing. To ensure that no extrapolation is performed, the range of the parameters is slightly reduced in the validation and tests sets, namely 3% and 6% reduction, on both ends, for the validation and test sets, respectively.

Remark 4.1: When data is already available, as expected in an industrial environment, the use of cases previously executed by engineers is expected to provide a better coverage of the design space. This is because when sampling the design space with quasi-random Halton sequencing, no attention is paid to the fact that the region of the design space leading to subsonic cases requires less sampling than the area that corresponds to transonic cases. Therefore, the results presented here are expected to be conservative and even better predictions are expected when the available data has been generated by an expert engineer.

To assess the influence of the background mesh, two different background meshes are considered in this study. These are depicted in Figure 15. The first background mesh has 976 nodes, 1780 triangular elements and 52 quadrilateral elements in the inflation layer. Local refinement has been defined by introducing two points sources, in the leading and trailing edges, and

a line source connecting the leading and trailing edge. The second background mesh has 9139 nodes, 18,076 triangular elements and 642 quadrilateral elements in the inflation layer. Local refinement has been defined by introducing a line source connecting the leading and trailing edge. It is worth noting that the anisotropic refinement near the wall in the background meshes is not designed to capture the flow physics, but to capture the expected large variation of the spacing near the aerofoil without refining isotropically.

To study the effect of the ANN hyperparameters and to compare the accuracy of the spacing prediction with the two different background meshes, a hyperparameter tuning is first performed using a simple grid search. The accuracy of the predictions is measured using the coefficient of determination, or R^2 , defined as

$$R^2 = 1 - \frac{\sum_{k=1}^{N_{tr}} \sum_{i=1}^M (y_i^k - h_i^k)^2}{\sum_{k=1}^{N_{tr}} \sum_{i=1}^M (y_i^k - \bar{y})^2}$$

where y_i^k are the target values, h_i^k the predicted values and \bar{y} is the mean of the target values.

Figure 16 shows the variation of the R^2 as a function of the number of layers and the number of neurons in each layer when using $N_{Tr} = 40$ training cases. The hyperparameter study shows that there is very little influence of the ANN architecture on the R^2 , which means that a shallow ANN with only one hidden layer and 10 neurons is enough to produce accurate predictions.

Using the best ANN architecture for each number of training cases, and for the two different background meshes considered, Figure 17 shows the best R^2 as a function of the number of training cases. The results show that, even for a very low number of training cases, i.e. $N_{tr} = 20$, the minimum R^2 is above 0.95 for both background meshes, with a value as high as 0.98 for

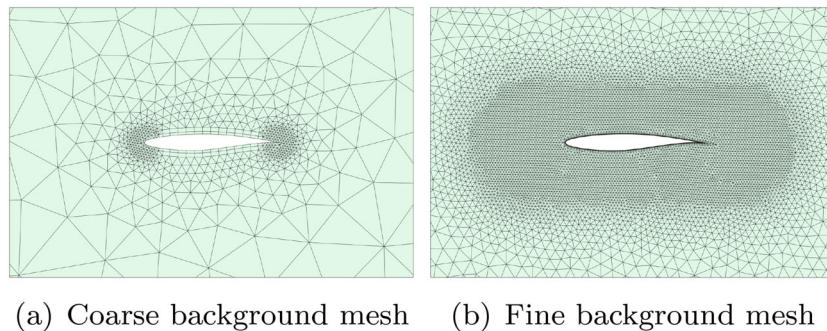


Figure 15. Two background meshes used to transfer the spacing from the computational mesh. (a) Coarse background mesh and (b) Fine background mesh.

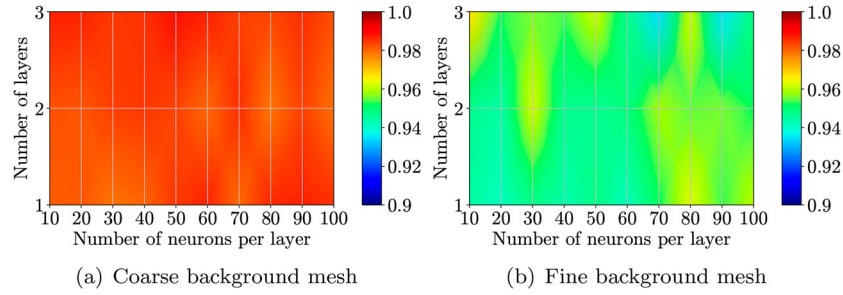


Figure 16. R^2 for the two different background meshes as a function of the number of layers and number of neurons in each layer with $N_{Tr} = 40$ training cases. (a) Coarse background mesh and (b) Fine background mesh.

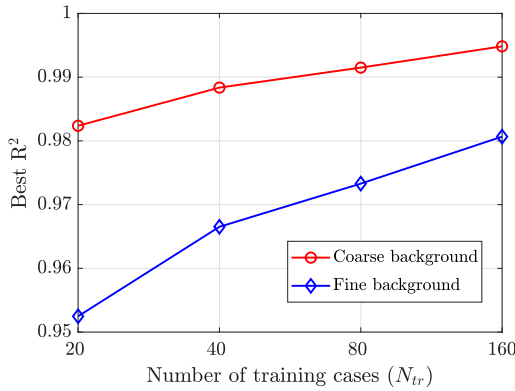


Figure 17. Best R^2 for the as a function of the number of training cases for the two different background meshes of Figure 15.

the coarse background mesh. In addition, it can be clearly observed that a better R^2 can be obtained if the coarse background mesh is used. In fact, it is worth noting that to achieve a value of R^2 of around 0.98, only 20 training cases are required when the coarse background mesh is used, compared to 160 training cases when the fine background mesh is employed. This behaviour is mainly due to the increased number of outputs in the ANN when the fine background mesh is employed. In this example, the number of outputs in the ANN is 976 when using the coarse background mesh and 9,139, almost ten times higher, when using the fine background mesh. This obviously induces a higher training time for the ANN with a fine background mesh. On average, the training time for the ANN that employs the coarse background mesh is five minutes, whereas the ANN that employs the fine background mesh requires, on average, seven minutes for training. As discussed in Section 3.2, and illustrated in Figures 12 and 13, using a finer background mesh is expected to predict near-optimal meshes with less elements and a more localised refinement in the areas of interest, however this has to be balanced with

the increased training time that induces using a finer background mesh.

To visually assess the accuracy of the predictions, Figure 18 shows the target and predicted meshes for three unseen cases during training, corresponding to solutions of Figure 14. The coarse background mesh of Figure 15(a) has been used to define the target and predicted discrete spacing function and the ANN has been trained using 40 training cases. The results clearly show the ability of the trained ANN to predict the spacing with localised refinement in the areas of interest. For the subsonic case, only a local refinement near the aerofoil is predicted, whereas for the transonic cases an extra local refinement is produced in the vicinity of the shocks.

Figure 19 shows the target and predicted meshes for the same three unseen cases, but using the finer background mesh of Figure 15(b) and 40 training cases. Similar to the previous example, it can be seen that the predictions produce local refinement in the regions of interest. However, it can be observed that the predictions seem to be more accurate when the coarse mesh is considered. This is mainly due to the fact that both ANNs have been trained with only 40 training cases. However, the number of outputs when using the fine background mesh is substantially higher, so to achieve a similar accuracy in the predictions, a larger number of training cases would be required.

The target mesh depends on the background mesh used, because the target spacing function is defined as a nodal field in the background mesh. As mentioned previously, this comparison clearly illustrates the compromise that a user needs to accept. To achieve a lower number of elements in the predicted meshes with the refinement localised on the areas of interest, a finer background mesh is preferred. However, this induces a more time consuming training process and requires

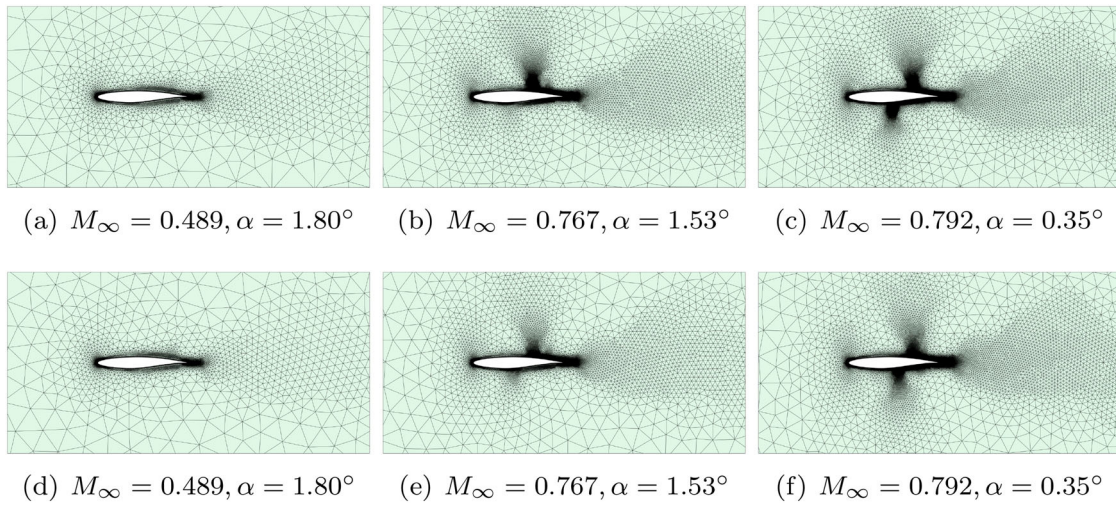


Figure 18. Target (top) and predicted (bottom) meshes for three unseen cases during training, corresponding to solutions of Figure 14 employing the coarse background mesh of Figure 15(a). (a) $M_\infty = 0.489, \alpha = 1.80^\circ$. (b) $M_\infty = 0.767, \alpha = 1.53^\circ$. (c) $M_\infty = 0.792, \alpha = 0.35^\circ$. (d) $M_\infty = 0.489, \alpha = 1.80^\circ$. (e) $M_\infty = 0.767, \alpha = 1.53^\circ$ and (f) $M_\infty = 0.792, \alpha = 0.35^\circ$.

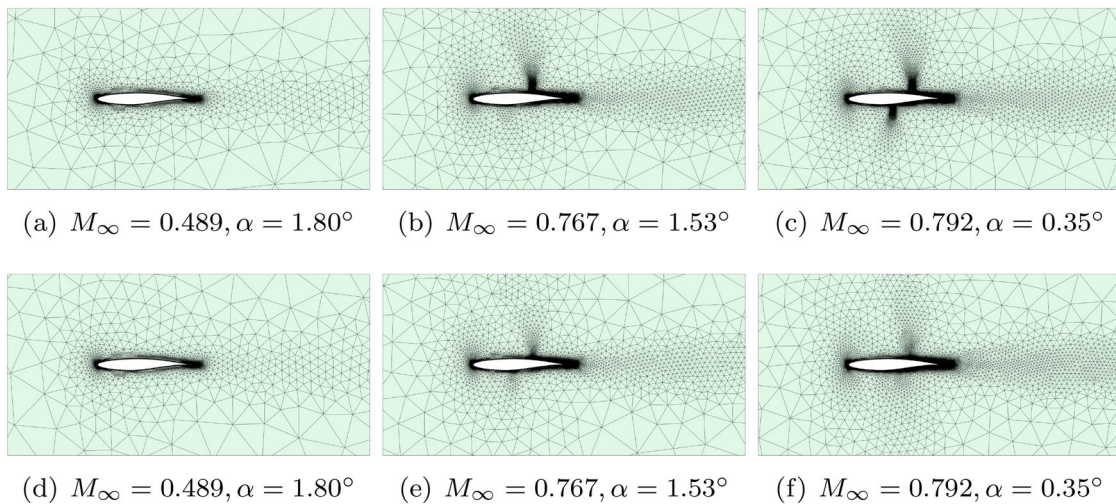


Figure 19. Target (top) and predicted (bottom) meshes for three unseen cases during training, corresponding to solutions of Figure 14 employing the fine background mesh of Figure 15(b). (a) $M_\infty = 0.489, \alpha = 1.80^\circ$. (b) $M_\infty = 0.767, \alpha = 1.53^\circ$. (c) $M_\infty = 0.792, \alpha = 0.35^\circ$. (d) $M_\infty = 0.489, \alpha = 1.80^\circ$. (e) $M_\infty = 0.767, \alpha = 1.53^\circ$ and (f) $M_\infty = 0.792, \alpha = 0.35^\circ$.

more training cases to produce accurate predictions. In contrast, a coarser background mesh produces good predictions with a limited number of training cases and the training times are low. However, the predictions produce meshes that contain a larger number of elements.

To further assess the accuracy of the ANN predictions, the predicted spacing function is compared to the target spacing function by computing the ratio between the target and predicted spacing at the centroid of each element, and for all the test cases. The results are displayed as a histogram in Figure 20 for the two ANN that employ a different background mesh. The results show again a very similar accuracy for both

background meshes. It can be observed that more than 80% of the elements have a predicted spacing between $1/1.15$ and 1.15 , which is considered accurate enough to produce a mesh capable of capturing all the relevant flow features appearing in the solution. For a value of the ratio higher than 1.15 , the prediction induces a larger spacing than desired, resulting in under-refined areas. Similarly, a value of the ratio below $1/1.15$ the prediction induces an over-refinement of the mesh. While the histograms show a slight better performance of the predictions obtained by the ANN that uses the finer background mesh, this is because the number of elements in the finer mesh is higher. When averaging the element size to produce the histograms, the very

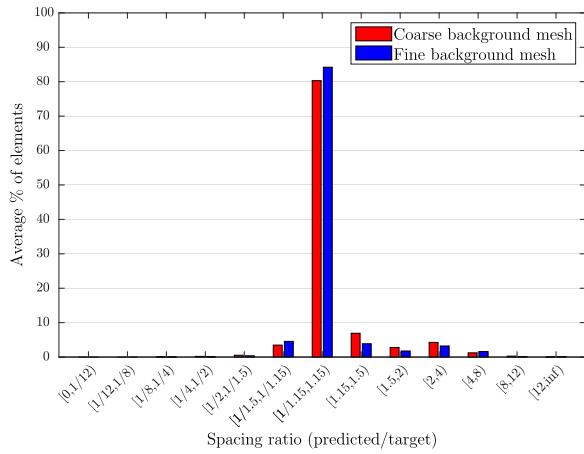


Figure 20. Histogram of the ratio between predicted and target spacing for the example with varying flow conditions and for the two different background meshes of Figure 15.

localised areas where the predicted size is not visually matching the target spacing, as shown in Figure 19, do not contribute significantly to the value shown in the histogram.

Finally, the suitability of the predicted meshes to perform simulations is studied. To this end, the three test cases of Figure 14, unseen during training of the ANN, are considered. The predicted near-optimal meshes are utilised to perform a simulation to compare the relevant quantities of interest against a reference solution. Table 1 reports the lift (C_L) and drag (C_D) coefficients obtained from a reference simulation and the ones obtained after performing a simulation with

Table 1. Comparison of the aerodynamic quantities of interest computed with the target and predicted meshes for three different flow conditions.

	$M_\infty = 0.489$ $\alpha = 1.80^\circ$		$M_\infty = 0.767$ $\alpha = 1.53^\circ$		$M_\infty = 0.792$ $\alpha = 0.35^\circ$	
	Target	Prediction	Target	Prediction	Target	Prediction
C_L	0.448	0.448	0.545	0.529	0.323	0.294
C_D	0.0069	0.0069	0.0169	0.0164	0.0177	0.0169

the predicted meshes using the coarse background mesh and 40 training cases.

The results show that for some configurations, corresponding to smooth flows, the predicted meshes lead to aerodynamic quantities of interest that perfectly match the reference results. When more complicated, transonic, cases are considered the predicted meshes lead to less accurate results as expected. From a practical point of view, the results show that the trained ANN produces good predictions of the regions that need localised refinement, even with only 40 training cases. These meshes can be considered as a very good starting point to perform an automatic adaptivity process, ensuring that no features of the solution are missed in the first computation.

To further analyse the accuracy of the predictions, Figure 21 shows the percentage error map of the predicted spacing for the three unseen cases during training, when the ANN is trained using the coarse background mesh of Figure 15(a). The results show that for the subsonic case, the error of the predicted nodal spacing is below 0.2%. For the transonic case with one shock on the upper surface of the aerofoil, the maximum error is near the shock region and is around 1.7%. Similarly, for the transonic case with two shocks on the upper and lower surfaces of the aerofoil the maximum error is near the shock region and is around 6%. This behaviour is mainly attributed to the fact that in the training dataset all cases present a refinement near the aerofoil as required by the subsonic case, but only a subset of these cases present one shock and a smaller subset present two shocks.

Finally, to compare the performance of the predictions when using a finer background mesh, Figure 22 shows the percentage error map of the predicted spacing for the three unseen cases during training, when the ANN is trained using the coarse background mesh of Figure 15(b). The maximum error of the predictions is again localised in the shock regions when

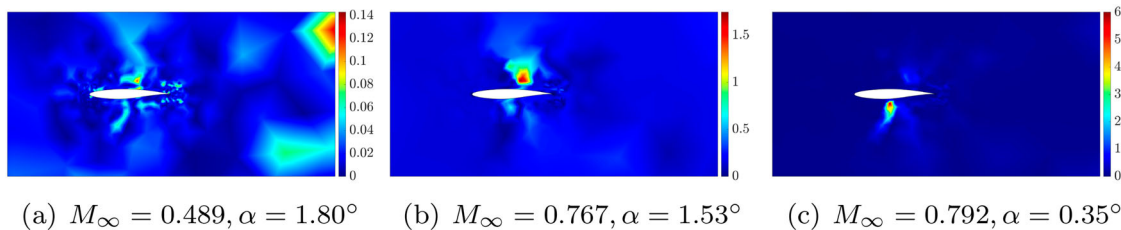


Figure 21. Percentage error of the predicted spacing for three unseen cases during training, corresponding to solutions of Figure 14 employing the coarse background mesh of Figure 15(a). (a) $M_\infty = 0.489, \alpha = 1.80^\circ$. (b) $M_\infty = 0.767, \alpha = 1.53^\circ$ and (c) $M_\infty = 0.792, \alpha = 0.35^\circ$.

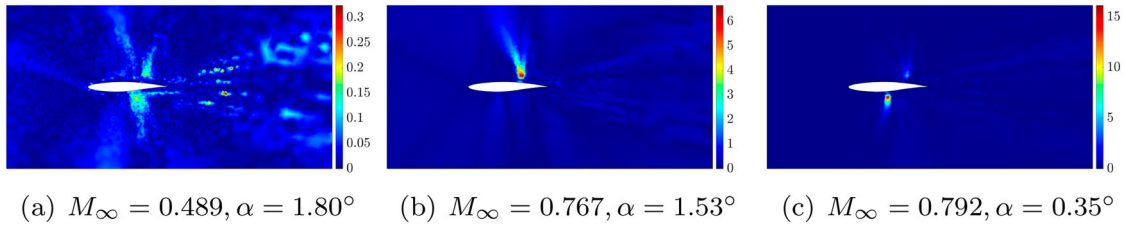


Figure 22. Percentage error of the predicted spacing for three unseen cases during training, corresponding to solutions of Figure 14 employing the coarse background mesh of Figure 15(b). (a) $M_\infty = 0.489, \alpha = 1.80^\circ$. (b) $M_\infty = 0.767, \alpha = 1.53^\circ$ and (c) $M_\infty = 0.792, \alpha = 0.35^\circ$.

transonic cases are considered. In addition, in all three unseen cases the maximum error is higher when using the finer background mesh and more localised. This behaviour is attributed to the fact that when a finer background mesh is considered, the target spacing at a node varies more abruptly in the parametric space and larger training datasets would be required to obtain accurate predictions.

4.2. Prediction of the Spacing for Variable Geometries

The next example considers the prediction of near-optimal meshes for variable geometric configurations and fixed turbulent transonic flow conditions, corresponding to $Re_\infty = 6.5 \times 10^6$, $M_\infty = 0.730$, and $\alpha = 2.79^\circ$. The geometry of the aerofoil is parametrised using two NURBS curves, for the top and bottom surface of the aerofoil, respectively. Each NURBS curve has eight control points and the original configuration, also referred to here as the *undeformed configuration*, corresponds to the control points detailed in Table 2.

Table 2. Position of control points for the NURBS curves defining the aerofoil in its original configuration.

i	1	2	3	4	5	6	7	8
X_i^+	0.000	0.000	0.105	0.391	0.594	0.799	0.933	1.000
Y_i^+	0.000	0.024	0.059	0.076	0.059	0.034	0.012	0.000
X_i^-	0.000	0.000	0.105	0.391	0.594	0.799	0.933	1.000
Y_i^-	0.000	-0.024	-0.059	-0.084	-0.046	-0.010	0.000	0.000

Notes: The superscripts $^+$ and $^-$ indicate points on the upper and lower parts of the aerofoil, respectively.

Table 3. Variation of the position of control points for the NURBS curves defining the aerofoil.

i	2	3	4	5	6	7
δX_i^+	4.3×10^{-2}	4.3×10^{-2}	3.1×10^{-2}	2.5×10^{-2}	1.5×10^{-2}	5.3×10^{-3}
δY_i^+	2.2×10^{-2}	2.2×10^{-2}	1.5×10^{-2}	1.2×10^{-2}	7.4×10^{-3}	2.7×10^{-3}
δX_i^-	–	4.3×10^{-2}	4.5×10^{-2}	3.9×10^{-5}	3.2×10^{-2}	1.9×10^{-2}
δY_i^-	–	2.2×10^{-2}	2.3×10^{-2}	1.9×10^{-5}	1.6×10^{-2}	9.6×10^{-3}

Notes: The superscripts $^+$ and $^-$ indicate the upper and lower parts of the aerofoil, respectively.

The geometric parameters are the relative position of the control points with respect to the original or undeformed configuration. Table 3 details the range of variation of the position of the control points considered in this example. The control points corresponding to leading and trailing edges are fixed and, as pointed out in Remark 2.1, the second control point in the bottom curve is not varied freely to ensure \mathcal{G}^1 continuity of the resulting aerofoil geometry. The position of this control point is defined as $\mathbf{B}_2^- = \mathbf{B}_1^- + \vartheta(\mathbf{B}_1^- - \mathbf{B}_2^+)$ where ϑ is a geometric parameter that varies between 0.5 and 1.5. The weights of all the control points are taken as one and not altered during the deformation. Therefore, the total number of geometric parameters is 23.

To illustrate the variation of the solution induced by the variation of the geometric parameters, Figure 23 shows three solutions corresponding to three different geometric configurations.

The position of the control points for the three geometric configurations of Figure 23 are detailed in Tables 4–6, respectively.

The plots in Figure 23 show the increased complexity of the pressure fields that result from varying the geometry of the aerofoil, compared to the previous example with varying operating conditions. The first example shows two shocks in the upper and lower surface of the aerofoil respectively. The second geometry induces a more complicated flow pattern with two shocks in the upper surface of the aerofoil and also a relatively strong shock in the lower surface. Finally,

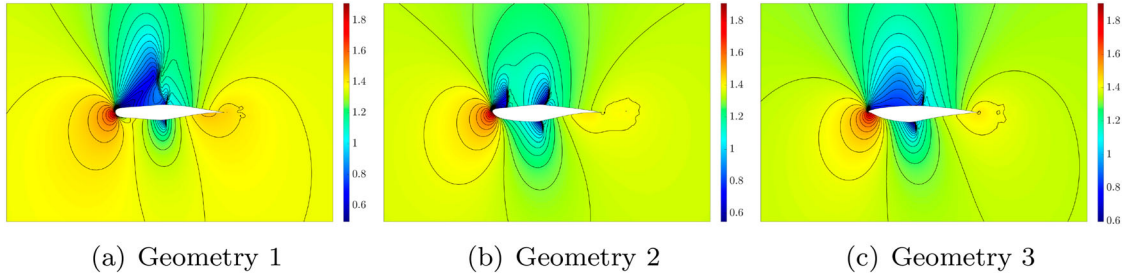


Figure 23. Pressure field and isolines for three different geometric configurations with $Re_\infty = 6.5 \times 10^6$, $M_\infty = 0.730$, and $\alpha = 2.79^\circ$. (a) Geometry 1. (b) Geometry 2 and (a) Geometry 3.

Table 4. Position of control points for the NURBS curves defining the aerofoil of Figure 23(a).

i	1	2	3	4	5	6	7	8
X_i^+	0.000	-0.007	0.139	0.412	0.598	0.794	0.931	1.000
Y_i^+	0.000	0.043	0.046	0.071	0.067	0.031	0.011	1.000
X_i^-	0.000	0.008	0.130	0.406	0.595	0.822	0.924	1.000
Y_i^-	0.000	-0.048	-0.046	-0.077	-0.046	-0.002	-0.004	1.000

Table 5. Position of control points for the NURBS curves defining the aerofoil of Figure 23(b).

i	1	2	3	4	5	6	7	8
X_i^+	0.000	-0.019	0.123	0.413	0.583	0.801	0.934	1.000
Y_i^+	0.000	0.034	0.042	0.076	0.056	0.034	0.011	1.000
X_i^-	0.000	0.021	0.108	0.393	0.595	0.809	0.948	1.000
Y_i^-	0.000	-0.037	-0.074	-0.091	-0.046	0.002	-0.000	1.000

Table 6. Position of control points for the NURBS curves defining the aerofoil of Figure 23(c).

i	1	2	3	4	5	6	7	8
X_i^+	0.000	-0.035	0.075	0.412	0.594	0.795	0.928	1.000
Y_i^+	0.000	0.009	0.047	0.065	0.054	0.035	0.010	1.000
X_i^-	0.000	0.022	0.093	0.362	0.595	0.813	0.926	1.000
Y_i^-	0.000	-0.006	-0.059	-0.098	-0.046	-0.017	-0.006	1.000

the third geometry involves smoother gradients but still with shocks in the upper and lower surfacert of the aerofoil. It can be observed that predicting near-optimal meshes in this example is even more challenging. Not only is the number of parameters substantially

higher, but the nature of the parameters induces more extreme variations in the flow features that require a completely different mesh, with refinement in different regions for each case.

To ensure that the background mesh employed for each geometric configuration contains the same number of nodes, the mesh morphing approach described in Section 2.4 is employed. A coarse background mesh of the undeformed configuration is generated and deformed for each unseen geometric configuration using an elasticity approach. Figure 24 shows three deformed configurations of the background mesh for the geometries of Figure 23.

The background meshes have 976 nodes, 1780 triangular elements and 52 quadrilateral elements in the inflation layer, which is the same as the coarse mesh used in the previous example. Local refinement has been defined by introducing two points sources, in the leading and trailing edges, and a line source connecting the leading and trailing edge. Again, it is worth pointing out that the anisotropic refinement near the wall in the background meshes is not designed to capture the flow physics, but to capture the expected large variation of the spacing near the aerofoil.

Training sets with $N_{tr} = 20, 40, 80, 160, 320, 640$ training cases have been generated using Halton

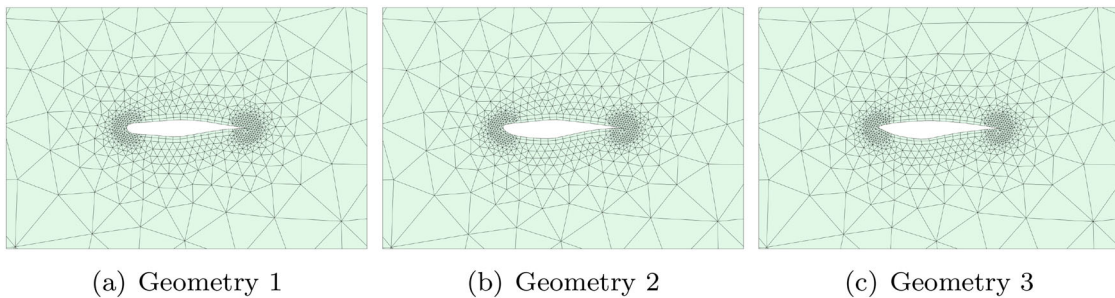


Figure 24. Background meshes obtained using an elastic deformation for the three geometric configurations of Figure 23. (a) Geometry 1. (b) Geometry 2 and (c) Geometry 3.

sequencing. Given the high dimensionality of the parameter space, the so-called scrambled Halton sequencing (Vandewoestyne and Cools 2006) is utilised to avoid the undesired correlation in the sampling that occurs with the traditional Halton sequencing in high dimensional problems. As in the previous example, validation sets with $N_{val} = 5, 10, 20, 40, 80, 160$ cases are also generated using scrambled Halton sequencing. The accuracy of the ANN predictions is assessed using a fixed set of $N_{test} = 74$ test cases. Finally, the range of the parameters is slightly reduced in the validation and tests sets, namely 3% and 6% reduction, at both ends, for the validation and test sets, respectively, to avoid extrapolation.

Next, the effect of the ANN hyperparameters is studied employing a simple grid search. Figure 25 shows the variation of the R^2 as a function of the number of layers and the number of neurons in each layer when using an increasing number of training cases. The results clearly show that high values of the R^2 can be achieved with small training sets. Comparing the results with the hyperparameter study in the previous example, it is clear that larger training datasets are required to achieve values of the R^2 close to 1, given the substantial increase in the number of inputs. In this example, a training set with 640 samples provides high

values of the R^2 , even for shallow networks with a low number of neurons.

To better observe the influence of the number of training cases in the accuracy of the predictions, Figure 26 shows the best R^2 as a function of the number of training cases. The results confirm the accuracy of the predictions in this example and they also illustrate the increased difficulty of performing accurate predictions with large number of parameters. In the previous example, a value of R^2 near 0.99 was achieved with 80 training cases, whereas in this example the

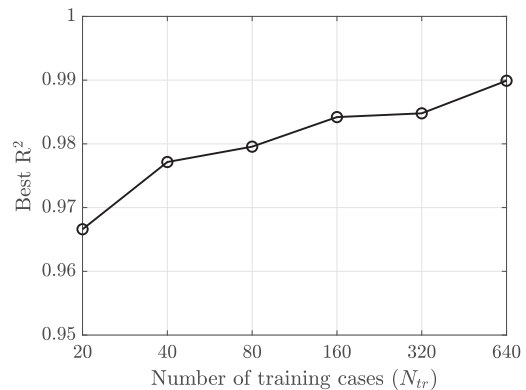


Figure 26. Best R^2 for the as a function of the number of training cases for the problem with varying geometric parameters.

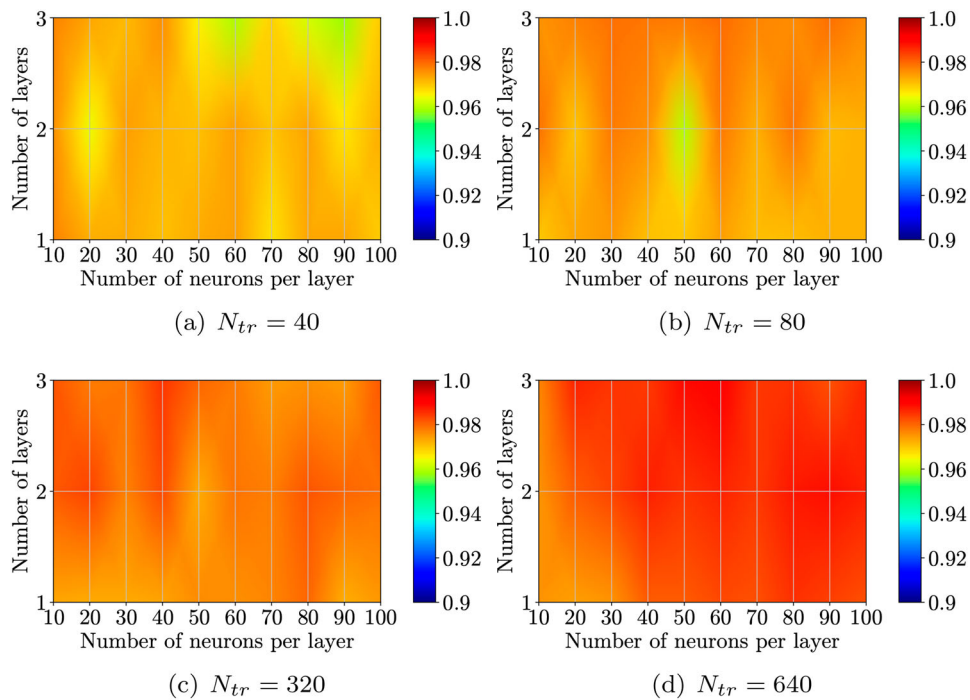


Figure 25. R^2 for different number of training cases (N_{tr}) as a function of the number of layers and number of neurons in each layer. (a) $N_{tr} = 40$. (b) $N_{tr} = 80$. (c) $N_{tr} = 320$ and (d) $N_{tr} = 640$.

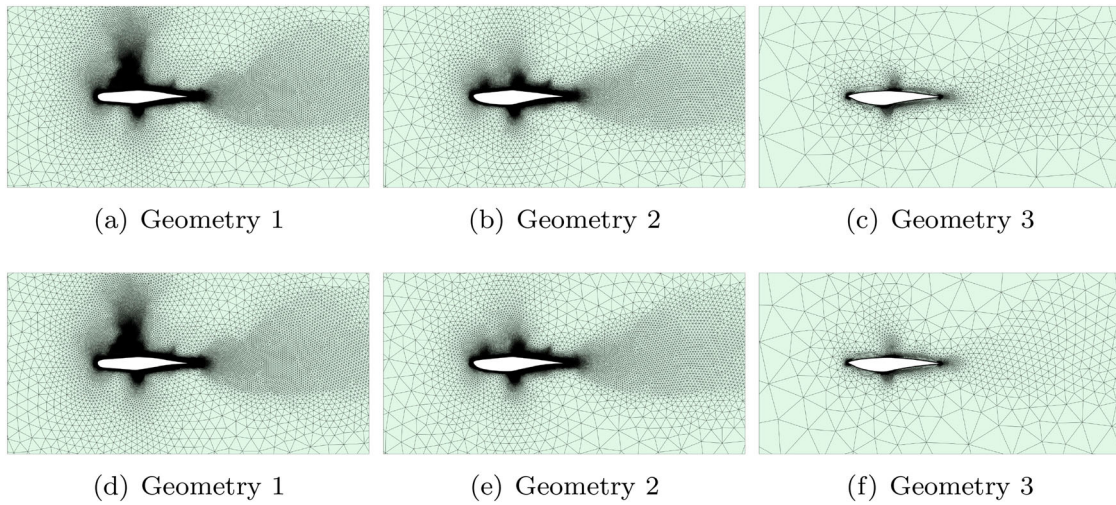


Figure 27. Target (top) and predicted (bottom) meshes for three geometric configurations unseen during training, corresponding to solutions of Figure 23 employing the coarse background meshes of Figure 24. (a) Geometry 1. (b) Geometry 2. (c) Geometry 3. (d) Geometry 1. (e) Geometry 2 and (f) Geometry 3.

number of training cases required to achieve a similar R^2 is 640. In the previous example, with only two parameters, a sampling with 80 training cases corresponds to 8.9 cases per parameter, whereas in the current example a sampling with 640 cases corresponds to only 1.3 cases per parameter. This analysis already indicates the potential of the proposed ANN approach to accurately predict the spacing function for unseen geometric configurations.

To illustrate the predictive capability of the trained ANNs, Figure 27 shows the target and predicted meshes for three unseen cases during training, corresponding to solutions of Figure 23. The predictions are obtained with the ANN trained using 640 training cases. The results show that for three geometric configurations with very different target meshes, the predicted meshes successfully provide the required local refinement near the regions where the solutions present a large gradient.

As in the previous example, to further assess the accuracy of the ANN predictions, the predicted spacing function is compared to the target spacing function by computing the ratio between the target and predicted spacing at the centroid of each element, and for all the test cases. The results are displayed as a histogram in Figure 28 for the two different training sets with 320 and 640 cases. The results show the ability of the trained ANNs to accurately predict the spacing for the large majority of the elements. Using an ANN trained with 320 cases, 76% of the elements have a predicted spacing between 1/1.15 and 1.15, which is

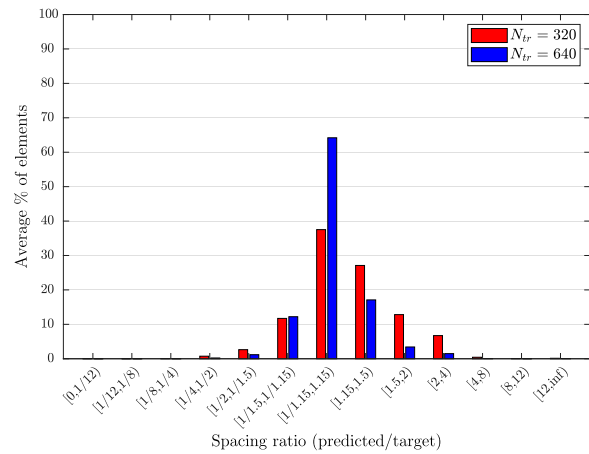


Figure 28. Histogram of the ratio between predicted and target spacing for the example with varying geometric parameters and for two ANNs trained with a different number of training cases.

considered accurate enough to produce a mesh capable of capturing all the relevant flow features appearing in the solution. Increasing the number of training cases to 640 leads to a significant benefit, as it produces meshes where almost 70% of the elements have a predicted spacing between 1/1.15 and 1.15. In addition, more than 90% of the elements have a predicted spacing between 1/1.5 and 1.5.

To conclude, the suitability of the predicted meshes to perform simulations is analysed. For the three test cases of Figure 23, unseen during training of the ANN, the predicted near-optimal meshes are utilised to perform a simulation to compare the relevant quantities of interest against a reference solution. Table 7 reports

Table 7. Comparison of the aerodynamic quantities of interest computed with the target and predicted meshes for three different geometric configurations.

	Geometry 1		Geometry 2		Geometry 3	
	Target	Prediction	Target	Prediction	Target	Prediction
C_L	0.689	0.683	0.354	0.343	0.566	0.539
C_D	0.0212	0.0202	0.0347	0.0349	0.0186	0.0219

the lift (C_L) and drag (C_D) coefficients obtained from a reference simulation and the ones obtained after performing a simulation with the predicted meshes using 640 training cases.

This study confirms the ability of the ANN to accurately predict the required spacing for unseen geometric configurations. For some configurations, the predicted meshes lead to aerodynamic quantities of interest that closely match the reference results, whereas for geometric configurations that lead to more complex flow features a larger discrepancy can be observed in the quantities of interest. Given the large number of geometric parameters considered in this example, more accurate predictions can obviously be obtained if the training set is increased. However, an alternative to avoid the generation of large data sets is to utilise the predicted meshes as a starting point for an adaptive process. The successful prediction of the regions where refinement is required will ensure that features are not missed in the first iteration of an adaptive process, and it is expected that this will lead to faster convergence of an adaptive process without human intervention for generating an initial mesh.

5. Concluding Remarks

A methodology to predict a near-optimal spacing function for turbulent compressible flow simulations has been presented. The strategy enables the use of available accurate solutions to learn the spacing required to capture the flow features of unseen cases.

The strategy proposed involves three main stages, namely computing the target spacing for a given solution, transferring the spacing to a (potentially morphed) background mesh and training an ANN. After training the ANN it can be used to predict the spacing for unseen geometric configuration or flow conditions. The work proposed focuses on three main challenges that appear when dealing with turbulent compressible viscous flows. First, it is necessary to consider several

key variables to compute the target spacing. The combination of the pressure and Mach number is selected in this work. Second, a special treatment of stretched elements is proposed to ensure that the target spacing is a true representation of the given solution. Finally, the effect of using different numerical strategies to compute the Hessian of a key variable is discussed.

The results presented consider two problems with variable flow conditions and geometric parameters. In both cases accurate predictions can be made with limited training data and numerical studies are presented to show the influence of the ANN architecture and the size of the training dataset. The problem involving geometric parameters considers as geometric parameters the positions of the control points of the NURBS describing the boundary. Therefore, it provides a tight coupling with the CAD model.

The suitability of the meshes for performing simulations is also assessed by comparing the aerodynamic quantities of interest of reference results and the results obtained with the predicted meshes. Despite the reduced number of training cases employed, a good accuracy is obtained, indicating that the proposed strategy provides a very good initial mesh that can be further enhanced using an adaptivity procedure.

Future work will involve the use of an anisotropic metric to define the spacing. This requires the prediction of a metric per point rather than a single scalar.

Acknowledgments

The authors would like to thank Dr Jason Jones for the help provided in preprocessing the results of the simulation shown in Figure 8(a).

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

The financial support of the Engineering and Physical Sciences Research Council (Grant Number: EP/T009071/1) is gratefully acknowledged.

References

- Abadi, M., P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, et al. 2016. "TensorFlow: A System for Large-Scale Machine Learning." 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), Savannah, GA, USA, 265–283.

- Alfonzetti, S. 1998. "A Finite Element Mesh Generator Based on An Adaptive Neural Network." *IEEE Transactions on Magnetics* 34 (5): 3363–3366. <https://doi.org/10.1109/20.717791>.
- Alfonzetti, S. 2003. "A Neural Network Generator for Tetrahedral Meshes." *IEEE Transactions on Magnetics* 39 (3): 1650–1653. <https://doi.org/10.1109/TMAG.2003.810325>.
- Alfonzetti, S., S. Coco, S. Cavalieri, and M. Malgeri. 1996. "Automatic Mesh Generation by the Let-it-grow Neural Network." *IEEE Transactions on Magnetics* 32 (3): 1349–1352. <https://doi.org/10.1109/20.497496>.
- Balla, Kensley, Ruben Sevilla, Oubay Hassan, and Kenneth Morgan. 2022. "Inverse Aerodynamic Design Using Neural Networks." In *Advances in Computational Methods and Technologies in Aeronautics and Industry*, 131–143. Springer.
- Bohn, J., and M. Feischl. 2021. "Recurrent Neural Networks As Optimal Mesh Refinement Strategies." *Computers & Mathematics with Applications* 97:61–76. <https://doi.org/10.1016/j.camwa.2021.05.018>.
- Chang-Hoi, A., L. Sang-Soo, L. Hyuek-Jae, and L. Soo-Young. 1991. "A Self-organizing Neural Network Approach for Automatic Mesh Generation." *IEEE Transactions on Magnetics* 27 (5): 4201–4204. <https://doi.org/10.1109/20.105028>.
- Chedid, R., and N. Najjar. 1996. "Automatic Finite-element Mesh Generation Using Artificial Neural Networks-Part I: Prediction of Mesh Density." *IEEE Transactions on Magnetics* 32 (5): 5173–5178. <https://doi.org/10.1109/20.538619>.
- Cheng, J., and M. J. Druzdzel. 2000. "Computational Investigation of Low-Discrepancy Sequences in Simulation Algorithms for Bayesian Networks." *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, Stanford, CA, USA, 72–81.
- Cook, P. H., M. A. McDonald, and M. C. P. Firmin. 1979. "Aerofoil RAE2822. Pressure Distributions, and Boundary Layer and Wake Measurements. Experimental Data Base for Computer Program Assessment." *AGARD Report AR 138*:47.
- Dawes, W. N., P. C. Dhanasekaran, A. A. J. Demargne, W. P. Kellar, and A. M. Savill. 2001. "Reducing Bottlenecks in the CAD-to-mesh-to-solution Cycle Time to Allow CFD to Participate in Design." *Journal of Turbomachinery* 123 (3): 552–557. <https://doi.org/10.1115/1.1370162>.
- Dyck, D. N., D. A. Lowther, and S. McFee. 1992. "Determining An Approximate Finite Element Mesh Density Using Neural Network Techniques." *IEEE Transactions on Magnetics* 28 (2): 1767–1770. <https://doi.org/10.1109/20.124047>.
- Fidkowski, K. J., and G. Chen. 2021. "Metric-based, Goal-oriented Mesh Adaptation Using Machine Learning." *Journal of Computational Physics* 426:109957. <https://doi.org/10.1016/j.jcp.2020.109957>.
- Foucart, C., A. Charous, and P. F. Lermusiaux. 2023. "Deep Reinforcement Learning for Adaptive Mesh Refinement." *Journal of Computational Physics* 491:112381. <https://doi.org/10.1016/j.jcp.2023.112381>.
- Fritzke, B. 1994. "A Growing Neural Gas Network Learns Topologies." *Advances in Neural Information Processing Systems* 7: 625–632.
- George, P. L., H. Borouchaki, F. Alauzet, P. Laug, A. Loseille, D. Marcum, and L. Maréchal. 2017. "Mesh Generation and Mesh Adaptivity: Theory and Techniques." In *Encyclopedia of Computational Mechanics Second Edition*, edited by Erwin Stein, Rene de Borst, and Thomas J. R. Hughes, Volume Part 1 Fundamentals, chapter 7. Chichester: John Wiley & Sons, Ltd.
- Hagan, T. M., H. B. Demuth, and M. Beale. 1997. *Neural Network Design*. Stillwater: PWS Publishing Co.
- Halton, J. H. 1964. "Algorithm 247: Radical-inverse Quasi-random Point Sequence." *Communications of the ACM* 7 (12): 701–702. <https://doi.org/10.1145/355588.365104>.
- Huang, K., M. Krügener, A. Brown, F. Menhorn, H. J. Bungartz, and D. Hartmann. 2021. "Machine Learning-Based Optimal Mesh Generation in Computational Fluid Dynamics." arXiv preprint arXiv:2102.12923.
- Kingma, D. P., and J. Ba. 2014. "ADAM: A Method for Stochastic Optimization." arXiv preprint arXiv:1412.6980.
- Lei, N., Z. Li, Z. Xu, Y. Li, and X. Gu. 2023. "What's the Situation with Intelligent Mesh Generation: A Survey and Perspectives." *IEEE Transactions on Visualization and Computer Graphics*. <https://ieeexplore.ieee.org/abstract/document/10141677>.
- Lock, C., O. Hassan, R. Sevilla, and J. W. Jones. 2023a. "Predicting the Near-Optimal Mesh Spacing for a Simulation Using Machine Learning." *Proceedings of the SIAM International Meshing Roundtable Workshop*, Amsterdam, The Netherlands.
- Lock, C., O. Hassan, R. Sevilla, and J. W. Jones. 2023b. "Meshing Using Neural Networks for Improving the Efficiency of Computer Modelling." *Engineering with Computers* 39: 3791–3820.
- Loseille, A., A. Dervieux, and F. Alauzet. 2010. "Fully Anisotropic Goal-oriented Mesh Adaptation for 3D Steady Euler Equations." *Journal of Computational Physics* 229 (8): 2866–2897. <https://doi.org/10.1016/j.jcp.2009.12.021>.
- Lowther, D. A., and D. N. Dyck. 1993. "A Density Driven Mesh Generator Guided by a Neural Network." *IEEE Transactions on Magnetics* 29 (2): 1927–1930. <https://doi.org/10.1109/20.250785>.
- Manevitz, L., A. Bitar, and D. Givoli. 2005. "Neural Network Time Series Forecasting of Finite-element Mesh Adaptation." *Neurocomputing* 63:447–463. <https://doi.org/10.1016/j.neucom.2004.06.009>.
- Manevitz, L., M. Yousef, and D. Givoli. 1997. "Finite-Element Mesh Generation Using Self-Organizing Neural Networks." *Computer-Aided Civil and Infrastructure Engineering* 12 (4): 233–250. <https://doi.org/10.1111/mice.1997.12.issue-4>.
- Martinetz, T., and K. Schulten. 1994. "Topology Representing Networks." *Neural Networks* 7 (3): 507–522. [https://doi.org/10.1016/0893-6080\(94\)90109-0](https://doi.org/10.1016/0893-6080(94)90109-0).
- Michal, T. 2019. "Development of an Anisotropic Solution Adaptive Meshing Tool for Production Aerospace Applications." In *Sixth Workshop on Grid Generation for Numerical Computations*, Paris, France.

- Peraire, Jaime, Morgan Vahdati, Ken Morgan, and Olgierd C. Zienkiewicz. 1987. "Adaptive Remeshing for Compressible Flow Computations." *Journal of Computational Physics* 72 (2): 449–466. [https://doi.org/10.1016/0021-9991\(87\)90093-3](https://doi.org/10.1016/0021-9991(87)90093-3).
- Piegl, L., and W. Tiller. 1996. *The NURBS Book*. Berlin: Springer Science & Business Media.
- Slotnick, J. P., A. Khodadoust, J. Alonso, D. Darmofal, W. Gropp, E. Lurie, and D. J. Mavriplis. 2014. *CFD Vision 2030 Study: A Path to Revolutionary Computational Aerosciences*. Technical report, NASA.
- Sørensen, K. A., O. Hassan, K. Morgan, and N. P. Weatherill. 2003. "A Multigrid Accelerated Hybrid Unstructured Mesh Method for 3D Compressible Turbulent Flow." *Computational Mechanics* 31 (1-2): 101–114. <https://doi.org/10.1007/s00466-002-0397-9>.
- Spalart, P. R., and S. R. Allmaras. 1992. "A One-Equation Turbulence Model for Aerodynamic Flows." *AIAA Paper*.
- Tlales, K., K. E. Otmani, G. Ntoukas, G. Rubio, and E. Ferrer. 2022. "Machine Learning Adaptation for Laminar and Turbulent Flows: Applications to High Order Discontinuous Galerkin Solvers." arXiv preprint arXiv:2209.02401.
- Triantafyllidis, D. G., and D. P. Labridis. 2002. "A Finite-element Mesh Generator Based on Growing Neural Networks." *IEEE Transactions on Neural Networks* 13 (6): 1482–1496. <https://doi.org/10.1109/TNN.2002.804223>.
- Vandewoestyne, B., and R. Cools. 2006. "Good Permutations for Deterministic Scrambled Halton Sequences in Terms of L2-discrepancy." *Journal of Computational and Applied Mathematics* 189 (1–2): 341–361. <https://doi.org/10.1016/j.cam.2005.05.022>.
- Wallwork, J. G., J. Lu, M. Zhang, and M. D. Piggott. 2022. "E2N: Error Estimation Networks for Goal-Oriented Mesh Adaptation." arXiv preprint arXiv:2207.11233.
- Xie, Z. Q., R. Sevilla, O. Hassan, and K. Morgan. 2013. "The Generation of Arbitrary Order Curved Meshes for 3D Finite Element Analysis." *Computational Mechanics* 51 (3): 361–374. <https://doi.org/10.1007/s00466-012-0736-4>.
- Yang, J., T. Dzanic, B. Petersen, J. Kudo, K. Mittal, V. Tomov, J.-S. Camier, et al. 2022. "Reinforcement Learning for Adaptive Mesh Refinement." In *International Conference on Learning Representations*. [Virtual].
- Zhang, Z., P. K. Jimack, and H. Wang. 2021. "MeshingNet3D: Efficient Generation of Adapted Tetrahedral Meshes for Computational Mechanics." *Advances in Engineering Software* 157:103021. <https://doi.org/10.1016/j.advengsoft.2021.103021>.
- Zhang, Z., Y. Wang, P. K. Jimack, and H. Wang. 2020. "MeshingNet: A New Mesh Generation Method Based on Deep Learning." In *International Conference on Computational Science*, 186–198. Springer.
- Zienkiewicz, O. C., and J. Z. Zhu. 1992a. "The Superconvergent Patch Recovery and a Posteriori Error Estimates. Part 1: The Recovery Technique." *International Journal for Numerical Methods in Engineering* 33 (7): 1331–1364. <https://doi.org/10.1002/nme.v33:7>.
- Zienkiewicz, O. C., and J. Z. Zhu. 1992b. "The Superconvergent Patch Recovery and a Posteriori Error Estimates. Part 2: Error Estimates and Adaptivity." *International Journal for Numerical Methods in Engineering* 33 (7): 1365–1382. <https://doi.org/10.1002/nme.v33:7>.