Data-driven wall models for Reynolds Averaged Navier-Stokes simulations

Michele Romanelli[a], Samir Beneddine[a], Ivan Mary[a], Héloïse Beaugendre[b,c], Michel Bergmann[b,d], Denis Sipp[a]

[a]*Aerodynamics, Aeroelasticity and Acoustics Department (DAAA), ONERA, University of Paris Saclay, Meudon, F-92190, France*
[b]*Bordeaux INP, University of Bordeaux, IMB, UMR 5251, Talence, F-33400, France*
[c]*Project team Cardamom, Inria Bordeaux-Sud Ouest, Talence, F-33400, France*
[d]*Project team Memphis, Inria Bordeaux-Sud Ouest, Talence, F-33400, France*

Abstract

This article presents a data-based methodology to build Reynolds-Averaged Navier-Stokes (RANS) wall models for aerodynamic simulations at low Mach numbers. Like classical approaches, the model is based on nondimensional local quantities derived from the wall friction velocity u_τ, the wall viscosity μ_w, and the wall density ρ_w. A fully-connected neural network approximates the relation $u^+ = f(y^+, p^+)$. We consider reference data (obtained with RANS simulations based on fine meshes up to the wall) of attached turbulent flows at various Reynolds numbers over different geometries of bumps, covering a range of wall pressure gradients. After training the neural networks on a subset of the reference data, the paper assesses their ability to accurately recover data for unseen conditions on meshes that have been trimmed from the wall up to an interface height where the learned wall law is applied. The network's interpolation and extrapolation capabilities are quantified and carefully examined. Overall, when tested within its interpolation and extrapolation capabilities, the neural network model shows good robustness and accuracy. The global error on the skin friction coefficient is a few percent and behaves consistently over all the considered test cases.

Keywords: wall model, machine learning, RANS, neural network

1. Introduction

The availability of reliable and accurate wall laws is one of the main challenges of modern CFD. Wall models are mandatory for applying Large Eddy Simulations (LES) in representative aeronautical applications, which remain out of reach if all turbulent scales are solved in the boundary layers [1]. Modeling the near-wall region is also needed for Reynolds-Averaged Navier-Stokes simulations (RANS) when using immersed boundary methods (IBM) [2] to represent physical boundaries, one major tool in the highly competitive industrial design environment that requires fast and reliable simulations. Additionally, even though RANS simulations are now affordable for complex configurations, lightweight RANS computations are crucial for parametric studies, flow control, or shape optimization, where numerous simulations are often needed. Therefore, even with the current progress in available computational power, there is still the need to reduce the cost of such simulations.

In wall-bounded flows, a large number of computational cells are required to resolve the boundary layer profiles to correctly represent the strong gradients established in the wall-normal direction. Thus, wall-resolved RANS simulations require the first computational cell to be in the viscous sublayer at a wall distance below $y^+ = 1$. Wall models allow for loosening these grid requirements, thus reducing the grid size. The computational cost gain is linked to this reduction as well as a lower aspect ratio of near-wall cells, resulting in a reduced computational stiffness [3].

The existence of universal wall laws for turbulent boundary layers relies on boundary layer theory, scaling arguments, and dimensional analysis. When the flow quantities vary slowly in the streamwise direction (as compared to the wall-normal direction), the shape of the streamwise velocity profile $u_{\parallel}(y)$ becomes invariant in the inner region of the boundary layer when scaled with appropriate quantities, here the wall viscosity μ_w, the density ρ_w and the friction velocity u_τ, computed using the skin friction τ_w:

$$u_\tau = \sqrt{\frac{\tau_w}{\rho_w}} \quad \text{with} \quad \tau_w = \mu_w \left.\frac{\partial u_{\parallel}}{\partial y}\right|_{y=0}. \tag{1}$$

In the absence of streamwise pressure gradient, it is straightforward to show that:

$$u^+ = f(y^+), \qquad u^+ = \frac{u_{\parallel}}{u_\tau} \qquad y^+ = \frac{\rho_w u_\tau}{\mu_w} y, \tag{2}$$

where y^+ is the dimensionless wall distance and u^+ the dimensionless streamwise velocity. From dimensional analysis, it is then possible to show that $u^+ = y^+$ close to the wall and that u^+ follows a logarithmic profile for large y^+ (in the inner region). In the presence of streamwise pressure gradients or separation, these wall laws need to be adapted. A new non-dimensional parameter steps in, the streamwise wall pressure gradient:

$$u^+ = f(y^+, p^+), \qquad p^+ = \frac{\mu_w}{\rho_w^2 u_\tau^3} \frac{\partial p}{\partial x}, \tag{3}$$

where x is the streamwise coordinate. This additional parameter has been introduced by Afzal [4], who gave an analytical expression for the evolution of the nondimensional streamwise velocity

$$u^+ = \kappa^{-1} \left[log(y^+) - 2log\left(\frac{\sqrt{1 + p^+ y^+}}{2} \right) + 2\left(\sqrt{1 + p^+ y^+} - 1 \right) + \kappa B \right]. \tag{4}$$

Even if Afzal's wall law includes the effects of the pressure gradient on the boundary layer evolution, equation (4) is valid only in the logarithmic region of the boundary layer. Additionally, it can only be applied for adverse (i.e., positive) pressure gradients due to the square root in the expression. In practice, this may cause implementation issues and possible convergence problems when trying to impose the wall model.

More recently, more sophisticated wall laws have been introduced [1, 5]. These new non-equilibrium zonal models involve the resolution of the boundary layer equations on a sub-grid placed between the wall and the first grid point. But these approaches significantly increase the computational cost compared to analytical models and may still lack generality in some flow configurations. Additionally, accounting for the pressure gradient with such approaches may lead to numerical difficulties when using IBM, as reported by [6] for instance.

In the present article, contrary to Zhou et al. [7], we choose the dimensionless quantities (y^+, p^+) and u^+ for the input and output of our data-driven model, respectively. Neural networks are then used as universal interpolators on physically-scaled data to find a relation $u^+ = f(y^+, p^+)$. This straightforward approach is bound to provide more general results than a direct estimation of dimensional quantities. Given that there would exist a universal law $u^+ = f(y^+, p^+)$ for the near-wall region, once learned with enough data, this law could be used for any unseen geometry or unseen flow conditions.

The scope of the present work is restricted to RANS simulations. The paper describes in detail the practical implementation of such an approach and quantifies its performance in several test cases. To our knowledge, such work is absent from the existing literature. It may serve as a baseline for future articles that would follow a similar strategy using larger, higher-fidelity training datasets. Additionally, the resulting wall model is shared on a GitHub repository[1] so that it may be straightforwardly integrated in any CFD code following the implementation details given in the paper.

The paper is organized as follows. First, the flow configurations considered in the article are presented (§2). Then, the wall modeling approach is detailed (§3), as well as the architecture, training strategy, and CFD implementation of the neural network (§4). The last section presents the results obtained with the data-based model on training and unseen configurations (interpolation and extrapolation conditions) before concluding.

2. Flow configurations and dataset

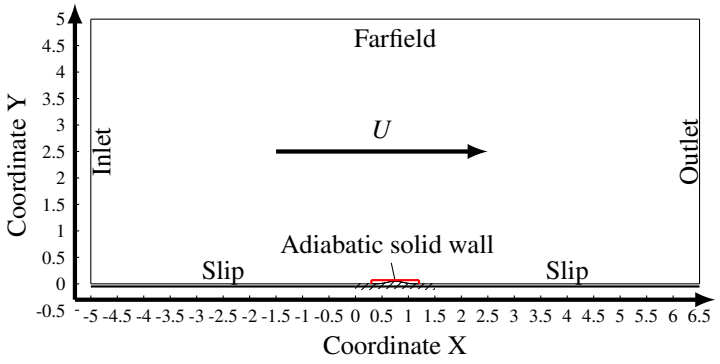
2.1. CFD solver

The reference datasets are based on wall-resolved RANS simulations with the Spalart Allmaras (S-A) turbulence model. We use ONERA's finite volume structured cell-centered compressible solver FASTs [8] for both the wall-resolved computations and the test of the wall models. All convective fluxes are discretized with the Roe flux scheme [9] extended to third-order with a MUSCL strategy. The steady solution of the S-A RANS equations is then computed using a local time-stepping technique, with a CFL value set to 20. All flow computations in this paper are initialized with a constant field (free-stream conditions), then converged with a decay of the residual norms of six orders of magnitude at least.

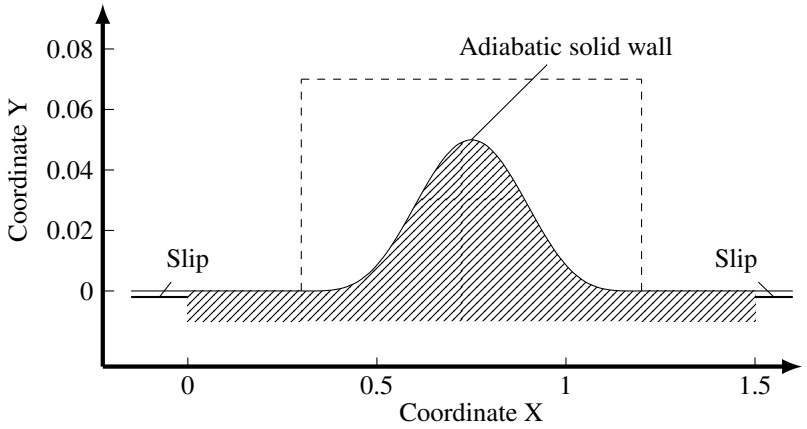
2.2. Flow configurations

The chosen configuration is a flow over a bump. It displays both favorable and adverse pressure gradients. Reference and training data are obtained from a set of fine wall-resolved RANS simulations over different bump geometries, inspired by a documented test case from NASA [10]. In the following, (X, Y) designates the Cartesian coordinate system associated with the bump configuration, while (x, y) refers to the local wall-tangent and wall-normal system. The Mach number (based on free-stream quantities) is set to $M = 0.2$, the reference temperature for the Sutherland law is the free-stream temperature $T_\infty = 300\text{K}$. The Reynolds number Re is based on free-stream quantities and a reference length $L = 1\,m$ (consistently with Rumsey [10]), which roughly corresponds to the length of the bump. Unless stated otherwise, all quantities

[1] Link: https://github.com/RomMic/Data-driven-wall-models-for-Reynolds-Averaged-Navier-Stokes-simulations.git



(a) Domain and boundary conditions for turbulent flow over the bump. The training dataset extraction is bounded by the red lines.



(b) Details of the bump. The extraction region to build the datasets is bounded by the dashed lines.

Figure 1: Bump case: Simulation domain, boundary conditions and geometry details.

are made non-dimensional using the length L, the free-stream velocity, temperature, and density. The bump geometry is then defined as:

$$Y(X) = \begin{cases} h \cdot \sin\left(\frac{\pi}{0.9}X - \frac{\pi}{3}\right)^4 & 0.3 \leq X \leq 1.2 \\ 0 & 0 < X < 0.3 \text{ and } 1.2 < X < 1.5 \end{cases}, \tag{5}$$

where h is the height of the bump. Equation 5 is used for $X \in [0,\ 1.5]$, which corresponds to the adiabatic wall extent. The simulation domain spans $X \in [-5,\ 6.5]$ and $Y \in [0,\ 5]$, as shown in figure 1(a), where the chosen boundary conditions are also depicted (these boundary conditions are the same than the NASA case [10]). Slip boundary conditions are applied upstream and downstream from the adiabatic wall.

Training and evaluation data are extracted for a well-established, fully turbulent boundary layer. Thus, the extraction zone for the datasets is restricted to the range $X = 0.3$ to $X = 1.2$ (corresponding to the bump region, see figure 1(b)). Various Reynolds numbers and bump heights h are considered to build the training and test datasets. The considered Reynolds numbers are $Re = 10^6$, $Re = 3{\cdot}10^6$, $Re = 6{\cdot}10^6$ and $Re = 10^7$. Considered bump heights h are 0.05, 0.06, 0.07,

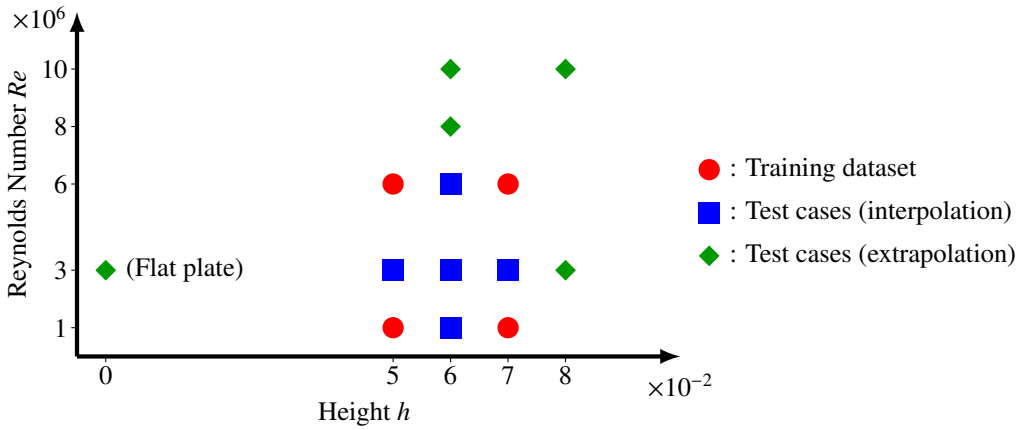


Figure 2: Training and test datasets obtained from different combinations of Reynolds number and bump height h.

and 0.08. They are selected to yield a weak to moderate pressure gradient along the geometry without inducing flow separation. A flow configuration with $h = 0$ (flat plate geometry) is also used as a test case.

Figure 2 shows the (Re, h)-combinations considered for the present article. The complete dataset (training + test) consists in 14 simulations. The training dataset (red dots) includes cases with moderate adverse pressure gradient p^+ (for instance, $h = 0.05$ and $Re = 6 \cdot 10^6$), but also a case with higher dimensionless pressure gradients approaching flow separation ($h = 0.07$ and $Re = 10^6$). This may be seen in figures 3 and 4, which show respectively the skin friction coefficient C_f (almost reaching zero for one of the cases) and the non-dimensional pressure gradient p^+ (exhibiting a very strong value for vanishing C_f) of the training cases.

The other configurations are used for testing. The test dataset contains five configurations that combine h and Re values within the range used for training (testing interpolation capabilities of the model in the (h, Re)-space), and five configurations that go beyond this range (testing extrapolation capabilities).

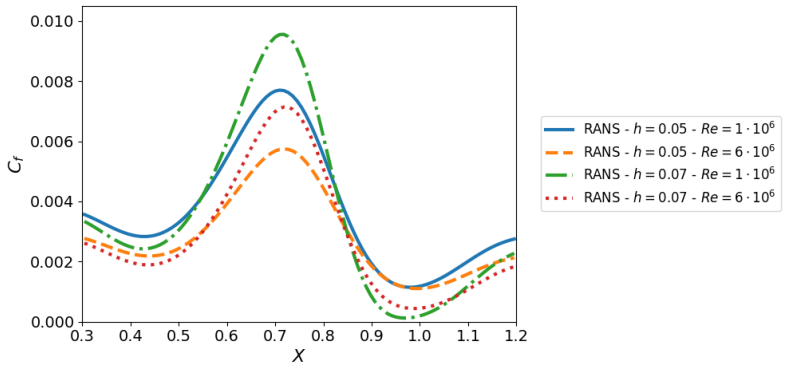


Figure 3: Streamwise evolution of the skin friction coefficient C_f for the four cases used for training.

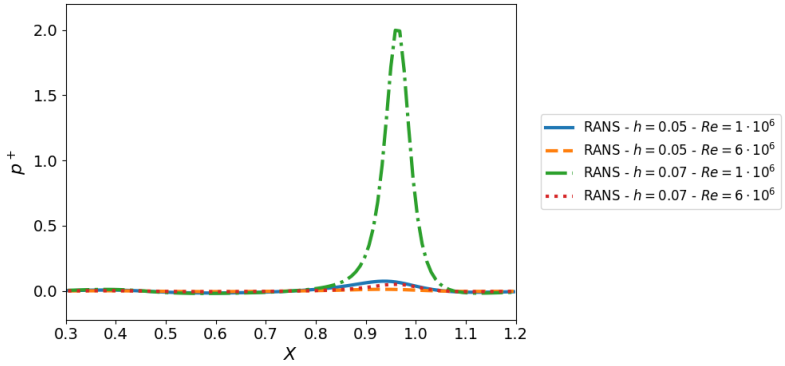


Figure 4: Streamwise evolution of the dimensionless pressure gradient p^+ for the four cases used for training.

The ratio between the bump height h and the boundary layer thickness δ at $X = 0.3$ for all the flow configurations is shown in table 1. Due to the presence of strong pressure gradients which affects the velocity profiles, the standard δ_{99} definition commonly used for flat plates cannot be applied here. Instead, the boundary layer edge and the associated boundary layer thickness δ are estimated through the vanishing of the shear stress and flow vorticity following the work of [11]. More details are given in Appendix D. Table 2 shows the strongest non-dimensional pressure gradient for all flow configurations. It reveals that the case ($h = 0.08, Re = 3 \cdot 10^6$) involves p^+ values that exceed by two orders of magnitude the training values (thus explaining some convergence problems discussed later for this case).

h/δ at $X = 0.3$					
	$h = 0$	$h = 0.05$	$h = 0.06$	$h = 0.07$	$h = 0.08$
$Re = 1 \cdot 10^7$			11.76		15.69
$Re = 8 \cdot 10^6$			11.54		
$Re = 6 \cdot 10^6$		9.26	11.11	12.96	
$Re = 3 \cdot 10^6$	0.0	8.62	10.16	11.86	13.56
$Re = 1 \cdot 10^6$		7.04	8.57	9.85	

Table 1: Ratio between bump height h and boundary layer thickness δ at $X = 0.3$ for all flow configurations

Strongest non-dimensional adverse pressure gradient p^+					
	$h = 0$	$h = 0.05$	$h = 0.06$	$h = 0.07$	$h = 0.08$
$Re = 1 \cdot 10^7$			0.014		0.099
$Re = 8 \cdot 10^6$			0.017		
$Re = 6 \cdot 10^6$		0.013	0.023	0.051	
$Re = 3 \cdot 10^6$	0.0	0.025	0.048	0.137	258.2
$Re = 1 \cdot 10^6$		0.076	0.187	1.997	

Table 2: Strongest non-dimensional adverse pressure gradient for all flow configurations

2.3. Computational grid for reference data

A fine-structured grid is used to obtain reference data for both training and testing cases. From $X = 0.3$ to $X = 1.2$, the grid is uniform along the curvilinear coordinate x. Over the bump, 100 computational points are placed, giving a $\Delta x \approx 0.01$. Upstream of the bump, in the zone ranging between $X = 0$ and $X = 0.3$, the mesh is refined following an exponential distribution in the streamwise direction: the smallest cells located at the leading edge start with a spacing $\Delta x \approx 0.001$, which progressively increases to conform to the bump region at $X = 0.3$. The downstream part of the mesh (i.e., in the zone from $X = 1.2$ and $X = 1.5$) mirrors the upstream one such that the full grid is symmetrical with respect to the bump geometry. Finally, the streamwise structure of the mesh is completed from the bump geometry to the edges of the simulation domain with a growing mesh spacing characterized by a growth ratio of 10%. In the wall-normal direction, the first computational point is placed at the same distance from the wall for all streamwise locations, giving a dimensionless wall distance between $y^+ = 0.01$ and $y^+ = 0.6$ along the bump for all the different cases for the training database. A 2% growing ratio for the cell size is used in the wall-normal direction.

Note that the case $h = 0.05$ and $Re = 3 \cdot 10^6$ corresponds to the documented test-case from NASA [10], on which the current numerical setup has been validated (see Appendix A).

3. Wall model strategy

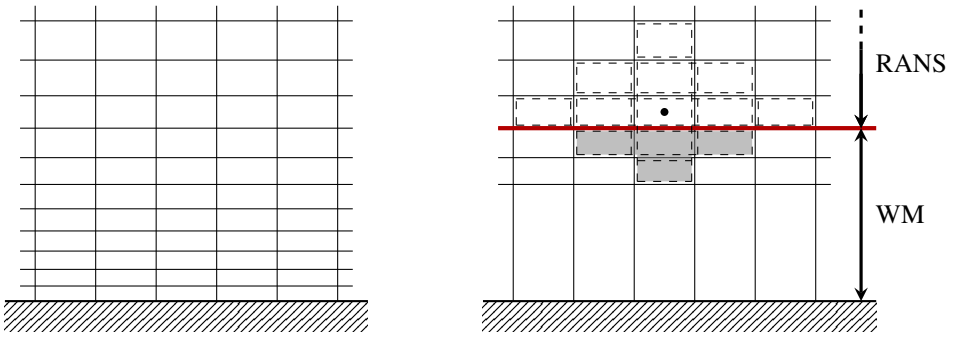
3.1. General strategy

When using wall laws, the flow is split into two regions: the near-wall region where the flow is modeled and a second region where the conventional integration scheme of the RANS equations takes over. The present strategy uses the wall model to impose a Neumann boundary condition at the interface compatible with the wall law. The wall law can therefore be seen as a Dirichlet-To-Neumann map. Similarly to a Robin boundary condition, it transforms the u velocity (Dirichlet) at the RANS interface into a normal velocity derivative $\partial_y u$ (Neumann) at that same location. The Neumann condition is enforced by imposing the exchanged numerical flux to the cells of the RANS region located near the interface. To do so, n layers of ghost cells (n being the size of the numerical stencil of the RANS spatial scheme) are added to the inner region, and their state is imposed by the wall model. Everything below these ghost cells does not affect the RANS computation (because it is outside of the numerical stencil of the RANS region's cells).

For this strategy to work, the height of the cells close to this interface should be small enough to resolve the local wall normal gradients of the flow variables. For example, if this interface is located in the log-layer, we may impose $\Delta y^+ = \kappa y^+ \Delta u^+$ with $\Delta u^+ \approx 1$. Hence, Δy^+ can be significantly larger than the value $\Delta y+ \approx 1$ that is required close to the wall, typically $\Delta y^+ \approx 41$ if the interface is located close to $y^+ \approx 100$.

In practice, the grids that include the ghost cells are obtained by "trimming" resolved RANS grids, obtained by removing the first cells near the wall, up to the desired y^+ location where Neumann condition is applied. A sufficient number of cells to build the numerical stencil of the first RANS resolved cell layer is kept below the interface (ghost cells) to enforce the Neumann boundary condition.

A schematic example of the grid structure is given in figure 5. A classic grid for the wall-resolved simulation (a) is compared to the trimmed grid (b) used with the presented wall model. The numerical stencil for the RANS integration of the first cell above the wall model interface is



(a) Grid structure used for wall-resolved RANS simulations.

(b) Grid structure used for RANS simulations coupled with wall modeling and trimmed grid. The numerical stencil of the first RANS integrated cell (dotted) is bounded by dashed lines. Wall modeled cells of the numerical stencil (ghost cells) are highlighted in gray.

Figure 5: Example of grid structure for RANS simulations in the near-wall region. Comparison between a standard wall refined grid and a "trimmed" grid.

also shown in figure 5 (b). In the present case, the adopted numerical scheme involves two wall-modeled cell layers below the RANS interface. Below them, all cells that would be kept are not actually used for the computation of the RANS region; consequently, no refinement constraint affects the cell size. A single coarse cell is thus employed as in figure 5 (b).

The advantage of this strategy is that it decreases the global number of solution grid points in the near-wall region without affecting grid resolution in the RANS region above wall model enforcement. Consequently, a correct evaluation of tested wall functions is achieved since the numerical error induced by grid coarsening (see, for instance, [3]) is limited.

The following sections provide specific details regarding the computation of the needed flow quantities imposed at the ghost cells: the modeling of flow temperature and density (section 3.2), the velocity components (section 3.3), and of the S-A variable (section 3.4). Note that in this section, we assume that a wall model $u^+ = f(y^+, p^+)$ is known. Its derivation is explained later on in section 4.

3.2. Temperature, density, and pressure gradient

The wall-normal temperature profiles are assumed to follow the Crocco-Busemann relation [12] adapted for adiabatic wall conditions:

$$T(y) = T_w - A\left(u_{\parallel}(y)^2 + u_{\perp}(y)^2\right), \text{ with } A = \frac{T_w - T_e}{U_e^2}, \tag{6}$$

where T_w is the wall temperature, T_e and $U_e = u_{\parallel}(\delta)^2 + u_{\perp}(\delta)^2$ are the flow temperature and velocity magnitude outside the boundary layer of thickness δ ($u_{\parallel}$ and $u_{\perp}$ being respectively the tangential and normal velocity components with respect to the wall). Crocco-Busemann's equation sets the relation between the flow temperature and the velocity magnitude for the modeled cell layers in the near-wall region. Since A and T_w are unknown, they have to be determined for applying the relation to the ghost cells. Assuming that the Crocco-Busemann temperature profile extends along the boundary layer, equation 6 continuity can be enforced at the interface between wall-modeled cells and RANS ones.

The wall temperature T_w and the ratio A are determined, at each solver time step, by fitting the Crocco-Busemann's relation to the first two cells in the resolved RANS region, above the enforcement of the wall model. Once T_w and A are determined, it is possible to extend Crocco-Busemann's temperature profile to all modeled cell layers in the near-wall zone.

In the boundary layer, the wall-normal pressure gradient $\partial p/\partial y$ is zero. The density profile can thus be obtained using the perfect gas law with the temperature profile obtained from Crocco-Busemann's law combined with pressure data at the first cell of the resolved RANS region.

Additionally, since the temperature T and density ρ evolution are known, the wall density ρ_w and wall molecular viscosity μ_w can be determined through the chosen viscosity model (in our case, Sutherland's law).

Finally, the pressure p being constant in the wall-normal direction, the pressure gradient $\partial p/\partial x$ in the modeled region may be obtained by projecting the pressure gradient evaluated at the RANS interface along the tangent direction x.

3.3. Velocity components

The wall model provides a functional relation between u^+, p^+, and y^+ (equation (3)). In developed form, it reads:

$$\frac{u_\parallel(y)}{u_\tau} = f\left(\frac{\rho_w u_\tau}{\mu_w} y,\ \frac{\mu_w}{\rho_w^2 u_\tau^3}\frac{\partial p}{\partial x}\right). \tag{7}$$

The previous section detailed how ρ_w, μ_w, and $\partial p/\partial x$ can be estimated. Therefore, the only values missing to determine the tangential velocity $u_\parallel(y)$ is the wall quantity u_τ. Its computation is explained in the following.

The skin-friction velocity u_τ is computed by using flow data sampled from a point S far from the wall but within the inner region of the boundary layer. Specifically, the tangential velocity $u_\parallel^S$ and corresponding wall distance y^S at S are needed. The friction velocity u_τ may be obtained by solving the non-linear equation

$$g(u_\tau) = 0, \tag{8}$$

where

$$g(u_\tau) = f\left(\frac{\rho_w u_\tau}{\mu_w} y^S,\ \frac{\mu_w}{\rho_w^2 u_\tau^3}\frac{\partial p}{\partial x}\right) - \frac{u_\parallel^S}{u_\tau}, \tag{9}$$

has been obtained by feeding equation 7 with sampling point and wall data. The solution to this nonlinear equation is obtained by an iterative Newton-Raphson method that reads

$$u_\tau^n = u_\tau^{n-1} - \frac{g(u_\tau^{n-1})}{g'(u_\tau^{n-1})}. \tag{10}$$

The derivative g' is obtained analytically:

$$g'(u_\tau) = \frac{\rho_w}{\mu_w} y^S \cdot \frac{\partial f}{\partial y^+} - \frac{3\mu_w}{\rho_w^2 u_\tau^4}\frac{\partial p}{\partial x} \cdot \frac{\partial f}{\partial p^+} + \frac{u_\parallel^S}{u_\tau^2}. \tag{11}$$

A guess for the skin friction velocity u_τ is required to initialize the Newton-Raphson procedure. For this, we use the simpler Werner and Wengle's wall law [13], which can be reformulated

as follows:

$$u_\tau\left(u_\parallel^S\right) = \begin{cases} \sqrt{\dfrac{\mu_w u_\parallel^S}{\rho_w y^S}} & \text{if } u_\parallel \leq \dfrac{\mu_w}{4\rho_w y^S} A^{\frac{2}{1-B}} \\ \left[\dfrac{1+B}{A}\left(\dfrac{\mu_w}{2\rho_w y^S}\right)^B u_\parallel^S + \dfrac{1-B}{2} A^{\frac{1+B}{1-B}}\left(\dfrac{\mu_w}{2\rho_w y^S}\right)^{1+B}\right]^{\frac{1}{1+B}} & \text{otherwise} \end{cases}, \tag{12}$$

with $A = 8.3$ and $B = \frac{1}{7}$.

The iterative process is applied until the residual value of g approaches zero within a given tolerance (set in the following to 10^{-9}). Once the skin friction u_τ is obtained, equation 7 can be used to determine the wall tangent velocity $u_\parallel(y)$ at any height y in the modeled region:

$$u_\parallel(y) = u_\tau \cdot f\left(\frac{\rho_w u_\tau}{\mu_w} y,\ \frac{\mu_w}{\rho_w^2 u_\tau^3}\frac{\partial p}{\partial x}\right). \tag{13}$$

The Newton-Raphson procedure to compute u_τ is summarized in algorithm 1 and figure 6 illustrates the general methodology to compute $u_\parallel(y)$ within the modeled region.

Algorithm 1: Computation of u_τ

Inputs: $u_\parallel^S, y^S, \rho_w, \mu_w, \partial p/\partial x$

Initialize u_τ with equation (12)

Compute $g \leftarrow f\left(\frac{\rho_w u_\tau}{\mu_w} y^S,\ \frac{\mu_w}{\rho_w^2 u_\tau^3}\frac{\partial p}{\partial x}\right) - \frac{u_\parallel^S}{u_\tau}$ (equation(8))

while $|g| > 10^{-9}$ **do**

  Compute $g' \leftarrow \frac{\rho_w}{\mu_w} y^S \cdot \frac{\partial f}{\partial y^+} - \frac{3\mu_w}{\rho_w^2 u_\tau^4}\frac{\partial p}{\partial x} \cdot \frac{\partial f}{\partial p^+} + \frac{u_\parallel^S}{u_\tau^2}$ (equation(11))

  Update $u_\tau \leftarrow u_\tau - g/g'$

  Compute $g \leftarrow f\left(\frac{\rho_w u_\tau}{\mu_w} y^S,\ \frac{\mu_w}{\rho_w^2 u_\tau^3}\frac{\partial p}{\partial x}\right) - \frac{u_\parallel^S}{u_\tau}$ (equation(8))

end

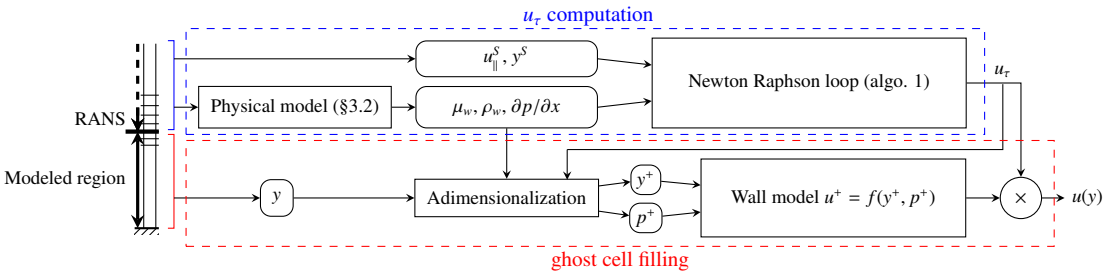


Figure 6: Schematic representation of wall model methodology to determine the tangential velocity profile within the modeled region.

Finally, the wall normal velocity $u_\perp$ is assumed to behave linearly in the inner-wall region, so that:

$$u_\perp(y) = u_\perp^S \cdot \frac{y}{y^S}. \tag{14}$$

This law is used to fill in the ghost cells below the interface. This approximation for $u_\perp$ is non-conservative, and more elaborated handling of this component may be considered in future works. However, it does not represent an issue for the quasi-equilibrium boundary layers that are involved in the present work. The error introduced in the wall-normal direction $u_\perp$ has been found to be negligible since the main velocity component is tangential to the wall surface (see section 5.5 for the study of mass conservation issues).

3.4. Spalart-Allmaras variable

Kalitzin et al. [3] studied the Spalart-Allmaras variable $\tilde{\nu}$ in the near wall region of a quasi-equilibrium boundary layer. They showed that the behavior of the dimensionless S-A variable was defined as:

$$\tilde{\nu}^+ = \frac{\rho\tilde{\nu}}{\mu} \tag{15}$$

can be modeled as

$$\tilde{\nu}^+ = \kappa y^+, \tag{16}$$

in the inner region of the boundary layer (viscous sublayer and logarithmic layer). Here, $\kappa = 0.41$ is the Von Kármán constant. The presented wall models set the S-A variable $\tilde{\nu}$ to respect equation 16 in the ghost cells below the interface. The accuracy of this model and its impact on the results are discussed in section 5.3.

4. Neural network training and implementation

4.1. Feedforward neural network

In this work, a Feedforward Neural Network (FNN) with different inputs learns the functional $f(\cdot)$. It is presented in the following section after a brief general introduction to FNN.

An FNN consists of an input layer, multiple hidden layers, and an output layer. Each of these hidden layers contains a variable number of nodes or neurons. The number of nodes in input and output layers is respectively imposed by the number of inputs and outputs of the neural network (NN). Every layer of the NN is dense, i.e., fully connected with the previous layer: every node of a single layer receives information from all nodes belonging to the previous layer and passes information to all nodes of the next layer.

Considering a layer L containing N_L neurons. In the following, the superscript $(\cdot)^{(L)}$ denotes quantities associated with this layer. The output $O_i^{(L)}$ of the i-th neuron is given by:

$$O_i^{(L)} = f_i^{(L)}(W_i^{(L)} + b_i^{(L)}), \tag{17}$$

where $f_i^{(L)}$ is the so-called activation function of the layer, $b_i^{(L)}$ is a scalar value (called a bias), and $W_i^{(L)}$ is the weighted inputs to the node i of layer L, obtained from a linear combination of the outputs of the previous layers:

$$W_i^{(L)} = \sum_{j=1}^{N_{L-1}} w_{i,j}^{(L)} O_j^{(L-1)}, \tag{18}$$

where $w_{i,j}^{(L)}$ is the weight coefficient linking the j-th neuron from layer $L-1$ to the i-th neuron of layer L. The weights coefficients and biases are the trainable scalar parameters that are optimized during the training phase of a neural network.

The activation functions $f_i^{(L)}$ add non-linearity to the network, allowing it to fit complex nonlinear patterns. This enables the NN to be trained on more complex tasks and perform better than a simple linear regression on data. In the present work, the Exponential Linear Unit (ELU) activation function is chosen for all neurons:

$$f_i^L(x) = \begin{cases} x & x > 0 \\ \alpha(e^x - 1) & x \leq 0 \end{cases}, \tag{19}$$

with the hyperparameter α set to 1 in our case. The choice for ELU is driven by improved learning characteristics compared to other commonly used activation functions (e.g., ReLu) [14].

4.2. Loss function definition and optimization strategy

The proposed wall model is data-driven, meaning the neural network is built upon training by data extracted from wall-resolved RANS simulations. The NN training optimizes $w_{i,j}^{(L)}$ and $b_i^{(L)}$ for all neurons by minimizing a loss function ϵ that evaluates the error between RANS data and NN predictions.

A first approach would be to consider the mean square error (MSE) for the loss function:

$$\text{MSE} = \frac{1}{N^S} \sum_{i=1}^{N^S} (f_{NN}(y_i^+, p_i^+) - u_{i,ref}^+)^2, \tag{20}$$

where N^S is the number of samples. Yet, it puts more emphasis on the high u^+ values since reducing the relative error of a given percentage becomes more advantageous for high u^+ values. Using the mean square relative error (MSRE):

$$\text{MSRE} = \frac{1}{N^S} \sum_{i=1}^{N^S} \left(\frac{f_{NN}(y_i^+, p_i^+) - u_{i,ref}^+}{u_{i,ref}^+} \right)^2, \tag{21}$$

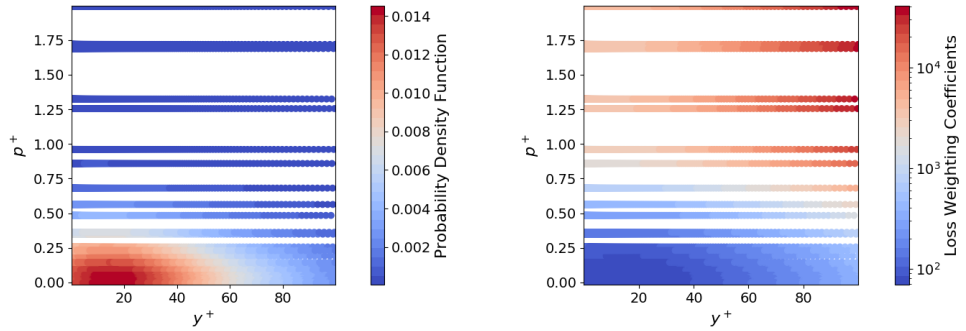
yields also difficulties for u^+ values close to zero since it may diverge even for very small absolute errors. Additionally, the relation $u^+ = f(y^+, p^+)$ learned with the MSRE loss function was found to be wavy, which was problematic for the determination of the friction velocity within the Newton-Raphson algorithm.

The present work followed the work of [15] to address the issues mentioned above, with a loss function based on a logarithmic expression of the error:

$$\epsilon = \frac{1}{N^S} \sum_{i=1}^{N^S} w_{\rho i} \left| \log \left(\frac{f_{NN}(y_i^+, p_i^+) + 1}{u_{i,ref}^+ + 1} \right) \right| + \frac{\lambda_2}{N^S} \left\| w^L \right\|_2 . \tag{22}$$

The first term in (22) is the mean absolute logarithmic error between the NN output $f_{NN}(y_i^+, p_i^+)$ and the corresponding reference value $u_{i,ref}^+$ from the training dataset. This error function does not present the problems mentioned above. The coefficient $w_{\rho i}$ is a weighting scalar that accounts for the sample distribution in the training dataset. A similar coefficient was used in the work of [7], and following their approach, we chose the weight to be inversely proportional to the sample density:

$$w_{\rho i} = \frac{1}{P_i}. \tag{23}$$



(a) (b)

Figure 7: Training samples colored by (a): the probability density function P_i of the sample density, (b): the loss weighting coefficient $w_{\rho i} = 1/P_i$ used during neural network training.

This term corrects learning issues that appear when the training samples are unevenly distributed: without the scaling w_ρ, regions in the (y^+, p^+) plane where the sample distribution is dense are artificially favored since the network attempts to minimize an average error over all samples. In our case, using w_ρ has shown improved results for high p^+ values, where there is a limited number of samples in the training dataset. Without this regularization, the network focuses on reducing the error for low p^+ values since it corresponds to most training samples.

A smooth probability density function P is obtained through a kernel density estimation (KDE) technique. It is estimated as the superposition of kernels, each of them being centered at the location (y_j^+, p_j^+) of a sample from the dataset:

$$P(y_i^+, p_i^+) = P_i = \frac{1}{N^S} \sum_{j=1}^{N^S} \frac{1}{\sigma_{y^+}\sigma_{p^+}} K\left(\frac{y_j^+ - y_i^+}{\sigma_{y^+}}, \frac{p_j^+ - p_i^+}{\sigma_{p^+}}\right). \tag{24}$$

Here σ_{y^+} and σ_{p^+} are respectively the standard deviation of the dataset with reference to y^+ and p^+ values and the Kernels are bivariate normal distributions:

$$K(\Delta y^+, \Delta p^+) = \frac{1}{2\pi\sqrt{1-r^2}} \exp\left[-\frac{1}{2(1-r^2)}\left(\Delta y^{+2} - 2r\Delta y^+\Delta p^+ + \Delta p^{+2}\right)\right], \tag{25}$$

with r such that $|r| < 1$ being the Pearson correlation coefficient between y^+ et p^+ distributions. Figure 7 shows both the sample density P_i of the dataset and the resulting coefficients $w_{\rho i}$.

The second term in the loss function ϵ is an L_2 regularization that avoids dominance of certain weights $w_{i,n}^L$ by penalising high valued ones. It is a common strategy in deep learning to help train and avoid over-fitting by forcing a homogeneous weights distribution. It is weighted by the parameter λ_2, which has been empirically set to $\lambda_2 = 0.001$ through trial-and-error. But the sensitivity to λ_2 is rather weak: different values have been tested, and when λ_2 is in the range [0.01,0.001], the results are nearly identical to those presented in the paper).

The training of the neural network is then carried out using a gradient-based algorithm, in our case, the Adam algorithm [16]. We use the deep-learning library TensorFlow [17].

Overfitting is monitored by splitting data into a training and a validation dataset. The training dataset is used to update the parameters of the network during the optimization. The validation

dataset is only used to evaluate the so-called validation loss. Overfitting can then be diagnosed if the validation loss significantly departs from the training loss. The training dataset is obtained through a random selection of 85% of data, while the validation data consist of the remaining 15%. More details on the neural network training are given in Appendix C.

4.3. Neural network architecture and training results

The neural network is trained to estimate, from the dimensionless wall distance y^+ and the dimensionless pressure gradient p^+, the dimensionless velocity u^+ (see eq. (2)). This fixes the structure of input and output layers in the neural network, which are respectively composed by two and one neurons. Since the dynamical range of each input strongly differs (e.g., $y^+ \in [0, 10^2]$ and $p^+ \in [-0.02, 2]$), an additional hidden layer is added after the input layer to allow the NN to simply rescale the input values (called "Normalization layer" in figure 8).

In the context of wall-modeled RANS simulations, the neural network architecture (number of nodes and hidden layers) strongly impacts the CFD solver's CPU cost. For this reason, the neural network structure has been optimized to minimize the number of operations without compromising the accuracy of the prediction (see Appendix B). The resulting optimized structure of the neural network comprises four main hidden layers, made of ten, ten, ten, and seven neurons, respectively. A schematic diagram of the NN architecture is given in figure 8.

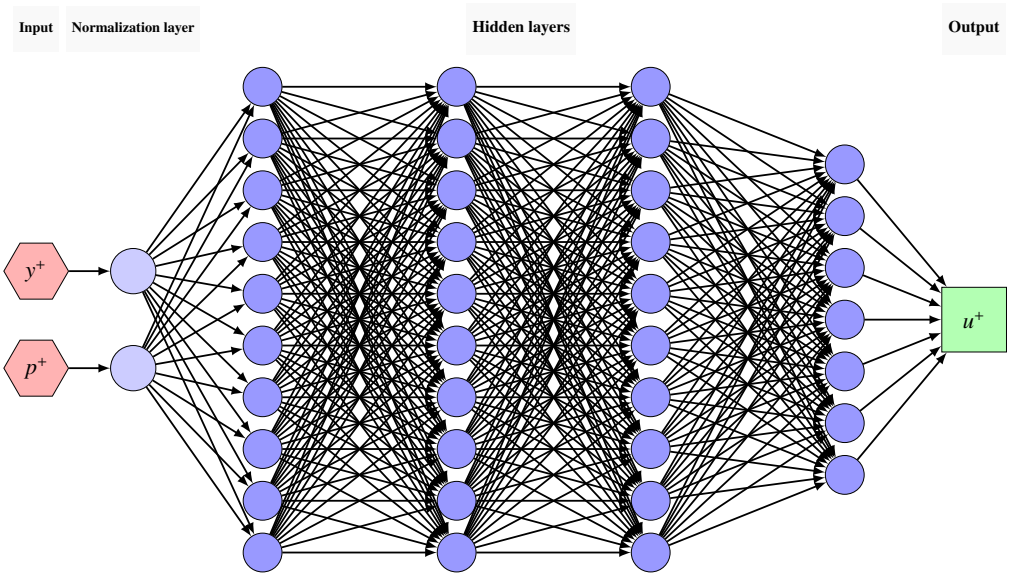
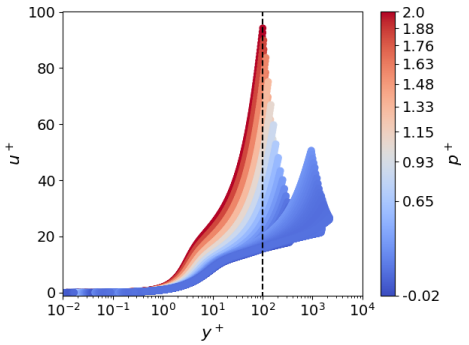


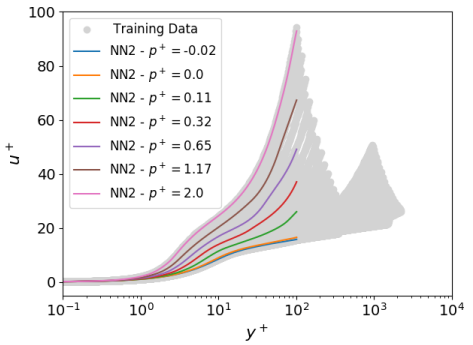
Figure 8: Schematic diagram of the resulting optimized feedforward neural network.

The inner region of the boundary layer, which is modeled by the present wall law, extends approximately up to $\frac{y}{\delta} \leq 0.15$. Training data, as well as the location of the interface for the wall model during testing, need to be located within this inner region. We have focused the learning process on samples satisfying $y^+ \leq 100$. This limit corresponds to the largest range of y^+ that is entirely contained within the 15% of the boundary layer thickness at each streamwise station.

The resulting learned relation $u^+ = f(y^+, p^+)$ is shown in figure 9, which also displays the training RANS dataset (only values below $y^+ \leq 100$ have been considered). An additional representation in the (y^+, p^+)-space may be seen in figure 10. The learned evolution of the dimen-



(a) Representation of samples with the inner region of the boundary layer ($\frac{y}{\delta} \leq 0.15$). The vertical dashed-line indicates the limit of the learning dataset $y^+ \leq 100$.



(b) Learned approximation of dimensionless wall velocity $u^+ = f(y^+, p^+)$. Comparison with training dataset.

Figure 9: Wall distance and pressure gradient informed neural network. Selected training data and learned wall normal evolution of the dimensionless velocity u^+.

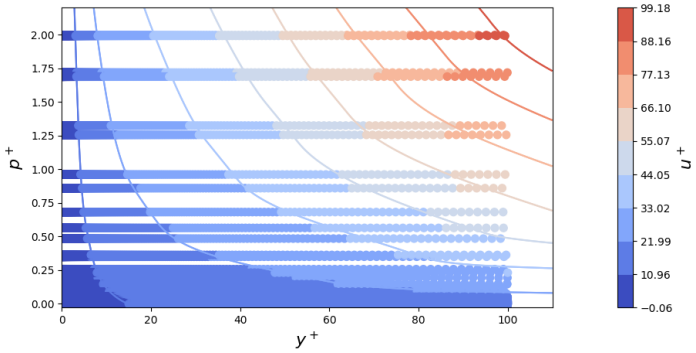


Figure 10: Contours of learnt relation $u^+ = f(y^+, p^+)$ (colored iso-lines) and training dataset points (filled colored circles). The relation beyond $y^+ \approx 100$ and $p^+ \approx 2$ is obtained through linear extrapolation.

sionless velocity u^+ expressed by the NN well reproduces the training dataset for the considered pressure gradients. Note that due to the geometric symmetry of the bump and its low height, the value range of the (dimensional) pressure gradient $\partial p/\partial x$ obtained in a given simulation is approximately symmetric (same maximal amplitude for positive and negative values). However, since the boundary layer flow is not symmetric with respect to the geometry (the friction velocity u_τ in particular), the range of dimensionless values p^+ is not symmetric, explaining why training data contains larger positive p^+ values than negative ones.

As seen in figure 10, the learning samples roughly cover $p^+ \in [-0.02, 2]$ and $y^+ \in [0, 100]$. To extend the capabilities of the model, if values beyond this range are encountered, then the neural network is replaced by a simple linear extrapolation based on $\partial f_{NN}/\partial y^+$ and $\partial f_{NN}/\partial p^+$ evaluated on the borders of the training domain, i.e., $y^+ \approx 100$ and $p^+ \approx 2$. This extrapolation is visible in figure 10.

4.4. Implementation in the CFD solver

The resulting neural network is implemented in the FORTRAN source code of the CFD solver FASTs (described in section 2.1). As shown by equations 17 and 18, dense feedforward neural networks are a chain of matrix operations (plus biases) followed by a nonlinear vector function performed between the inputs and outputs of each NN layer. Therefore, generating a code that reproduces the learned relation $u^+ = f(y^+, p^+)$ is straightforward for given weights of the network. The CFD code has been modified to read the NN parameters (architecture, weights, biases, and activation functions) and to evaluate the NN. Additionally, the derivatives of the neural network (with respect to the inputs y^+ and p^+) are needed for the Newton algorithm and the implementation of the Neumann boundary condition (see section 3). For this, the FORTRAN code emulating the NN is algorithmically differentiated to get $\partial f/\partial y^+$ and $\partial f/\partial p^+$. This has been done using TAPENADE [18], a source-to-source automatic differentiation engine.TAPENADE generates source codes corresponding to the two derivatives, which are integrated within the solver.

4.5. Summary of the general wall model strategy

Section 3 explains how the wall model is used within the solver, while section 4 details how the wall model is obtained using a neural network trained with reference data. The complete procedure may therefore be summed up as follows:

- Offline phase:
    1. Wall resolved reference RANS simulations are used to compute a training database containing (u^+, y^+, p^+) tuples.
    2. A neural network is trained to learn the general relation $u^+ = f(y^+, p^+)$ from the training database. Adequate choice of the loss function and parameters of NN (architecture, weights, biases) are presented in section 4.
    3. The NN representing function $u^+ = f(y^+, p^+)$ is translated into a FORTRAN code. This code is differentiated using TAPENADE to get FORTRAN routines that evaluate $\partial f/\partial y^+$ and $\partial f/\partial p^+$.
    4. The three FORTRAN routines (f, $\partial f/\partial y^+$, $\partial f/\partial p^+$) are implemented in the CFD software and compiled.

- Online phase:
    1. The CFD solver reads the NN parameters (architecture, weights, biases) defining the wall-law f.
    2. The wall-law f and its derivatives are used to determine and apply boundary conditions at the RANS interface following the numerical strategy described in section 3. In particular, this requires solving (at each streamwise location) for the skin-friction velocity u_τ from the knowledge of $u_\parallel^S$ at the RANS interface y^S (Newton-Raphson algorithm 1) and imposing the velocities predicted by the wall-law within the ghost cells of the modeled region (to mimic the Neumann boundary condition, see figure 6)

5. Results and discussions

5.1. Test procedure

This section presents the results obtained using the neural network as a wall model, tested on various flow configurations and geometries defined in section 2. The results are compared with the wall-resolved RANS simulation for the same parameters. The wall model is applied to structured trimmed grids, as explained in section 3. The numerical stencil of the spatial scheme in the solver includes two cell layers below the RANS interface, placed to a chosen value of y^+. Therefore, two layers of cells below the interface are kept from the reference mesh; all other cells below in the wall-normal direction are merged into a single coarse cell. Figure 5.b illustrates this grid structure. The coarse cell is beyond the numerical stencil of the RANS region and is consequently unused during the RANS computation. The wall model is applied to the first three layers of cells, and the standard RANS integration takes over from the fourth cell center above the wall. The sampling point is taken at the fifth cell from the wall in the RANS integrated zone.

The wall model is only applied to established turbulent boundary layers. For this reason, the complete simulation domain for the wall-modeled computation is limited to $X \in [0.3,\ 1.5]$, with an inflow condition that injects the fully developed boundary layer computed from the reference simulation at $X = 0.3$. Wall model performances are evaluated on the bump geometry only, from $X = 0.3$ to $X = 1.2$, which corresponds to the extraction zone for the training data.

For each configuration, three y^+ values are considered for the RANS interface: $y^+ \approx 10,\ 30,\ 50$. The first RANS-integrated cell is thus placed in the buffer layer, between the buffer layer and the logarithmic region, and in the logarithmic region, respectively.

The following subsections show first the global error for all the covered configurations, then more detailed results are presented and discussed for some selected cases.

5.2. Global errors

To evaluate the capabilities of the model, the wall model is tested on the ten test configurations proposed in 2, but also on the four training flows to obtain reference errors. The evaluation is thus performed both on seen and unseen configurations. Evaluating the model on all configurations allows comparing the error due to the approximate learned relation for u^+ and the impact of the assumptions made on other variables (temperature, density, and eddy viscosity, see section 3).

The evaluation of the global performances of the wall model is based on the estimation of the skin friction coefficient C_f on under-resolved grids compared to fully-resolved RANS simulations. A 2-norm error is computed between the reference RANS results and the simulation with the wall model. The global 2-norm error e_2 is obtained as follows

$$e_2 = \frac{\left\| C_f(X_i) - C_{f,ref}(X_i) \right\|_2}{\left\| C_{f,ref}(X_i) \right\|_2} = \frac{\sqrt{\sum_i [C_f(X_i) - C_{f,ref}(X_i)]^2}}{\sqrt{\sum_i [C_{f,ref}(X_i)]^2}} \tag{26}$$

where X_i are all the streamwise locations for X between 0.3 and 1.2.

Table 3 shows the global error e_2 computed for all the flow configurations and the three interface positions considered. The model shows good performances overall, with an error of 6.84% at most.

The global error for the training configurations (i.e. $h = 0.05 - Re = 10^6$, $h = 0.05 - Re = 6 \cdot 10^6$, $h = 0.07 - Re = 10^6$ and $h = 0.07 - Re = 6 \cdot 10^6$) is close to the error for unseen configurations. This shows that the learned relation does not suffer from interpolation

RANS interface at $y^+ \approx 10$					
	$h = 0$	$h = 0.05$	$h = 0.06$	$h = 0.07$	$h = 0.08$
$Re = 1 \cdot 10^7$			0.72%		0.88%
$Re = 8 \cdot 10^6$			0.75%		
$Re = 6 \cdot 10^6$		0.77%	0.8%	0.85%	
$Re = 3 \cdot 10^6$	0.61%	0.84%	0.93%	0.92%	*
$Re = 1 \cdot 10^6$		1.26%	1.35%	1.54%	
RANS interface at $y^+ \approx 30$					
	$h = 0$	$h = 0.05$	$h = 0.06$	$h = 0.07$	$h = 0.08$
$Re = 1 \cdot 10^7$			1.68%		2.09%
$Re = 8 \cdot 10^6$			1.92%		
$Re = 6 \cdot 10^6$		1.94%	2.27%	2.53%	
$Re = 3 \cdot 10^6$	1.21%	2.94%	3.41%	3.76%	3.98%
$Re = 1 \cdot 10^6$		5.54%	6.28%	6.84%	
RANS interface at $y^+ \approx 50$					
	$h = 0$	$h = 0.05$	$h = 0.06$	$h = 0.07$	$h = 0.08$
$Re = 1 \cdot 10^7$			2.9%		2.63%
$Re = 8 \cdot 10^6$			3.06%		
$Re = 6 \cdot 10^6$		3.05%	3.36%	3.54%	
$Re = 3 \cdot 10^6$	1.91%	3.57%	4.0%	4.34%	4.58%
$Re = 1 \cdot 10^6$		4.59%	5.1%	5.63%	

Table 3: 2-norm global error of the wall-modeled simulation on the skin friction coefficient C_f with respect to the reference wall-resolved RANS simulation for three different RANS interface positions and different combinations of bump height h and Reynolds number Re. The symbol * indicates that the computation failed to converge (non-convergence of the Newton-Raphson loop from algorithm 1)

or extrapolation issues in the range of flow conditions considered. However, the error increases as the RANS interface is further away from the wall, and it appears to reduce when lower p^+ values are encountered, both for a lower bump height and an increase in the Reynolds number. This observation provides information on the main source of error in the present model, and it is further developed in section 5.3.

The simulation case ($h = 0.08, Re = 3 \cdot 10^6$) fails to converge when the RANS interface is close to the wall (i.e. $y^+ \approx 10$). This configuration requires very high values of p^+, well beyond the extrapolation capabilities of the proposed neural network (two orders of magnitude higher than the highest value encountered during training). Yet, the problem does not persist for cases with a higher interface (i.e., $y^+ = 30$ and $y^+ = 50$). This is due to an overestimate of the skin friction coefficient C_f and skin friction velocity u_τ, drastically reducing the sensed value of p^+ fed to the neural network. The cause of this overestimation of friction in high p^+ valued areas is addressed in section 5.4.

Nonetheless, this unconverged case is interesting because it bounds the extrapolation capabilities of the model. It appears that it may be unable to treat (quasi)separated boundary layers due to the attached nature of the flows considered for training. This limitation, as well as possible solutions, are further discussed in conclusion.

5.3. Interpolation test results

Interpolation capabilities of the model (evaluated on the interpolation test configurations defined in section 2) are presented in more detail in this section. The following results compare the obtained profiles for different flow variables from the wall models. It provides a more detailed view of the modeling errors. For conciseness, let us focus on the case $h = 0.06$ and $Re = 3 \cdot 10^6$ (results on the other interpolation cases are similar). It is referred to as an interpolation case because it involves a combination of Reynolds number and bump height inside the training range values.

Figure 11 shows the skin friction coefficient C_f along the wall and its error with respect to the wall-resolved simulation. The error e is normalized by the mean C_f value of the reference simulation over the wall, such that

$$e(X_i) = \frac{C_f(X_i) - C_{f,ref}(X_i)}{\overline{C_{f,ref}}}, \tag{27}$$

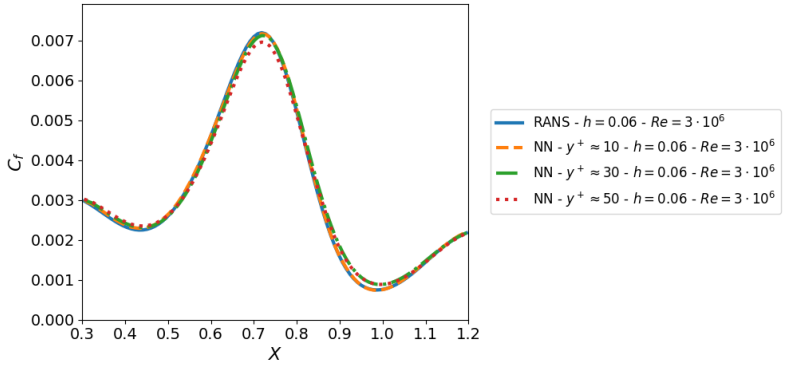
with X_i referring to the equally spaced solution points in $X-$coordinate direction along the considered portion of the wall, C_f the estimation of friction coefficient from the wall model, $C_{f,ref}$ the estimation of friction coefficient from the wall-resolved simulation and $\overline{C_{f,ref}}$ its mean value over the wall (the local value is not used for normalization to avoid artificial divergence of the error when C_f becomes too close to zero). Note that this error is purposely defined as a signed value (to evaluate potential model under/overestimation).

Globally, the wall model accurately estimates the skin friction evolution along the bump geometry, with the error increasing as the height of the interface increases. A maximum local error of 2 %, 8.5 % and 8.6 % is found for respective y^+ values of 10, 30 and 50. The maximum error is found near the top of the bump or near its downstream bottom area (where high p^+ values are expected). Note that since the average value of C_f is used to normalize the error, it is expected to find larger errors where C_f reaches its maximal value.

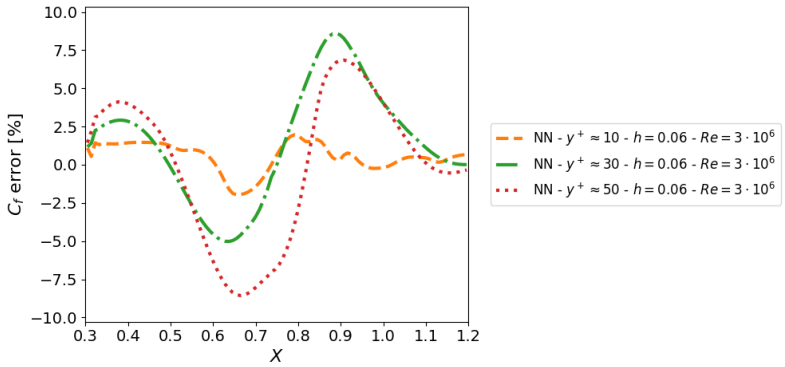
Figure 12 shows tangential velocity profiles obtained at $X = 0.75$ (top of the bump) and at $X = 0.95$ (downstream bottom area). The curves present the results of the wall model for the three considered RANS interface positions to the reference wall-resolved RANS simulation. At the top of the bump, the model closely matches fully resolved RANS results. In the downstream bottom area, the model still fits the RANS simulation when the model interface is located at $y^+ \approx 10$. The error increases for simulations with higher interfaces with a tendency to overestimate the velocity. Simulation with an interface at $y^+ \approx 30$ and $y^+ \approx 50$ show a very similar error and behavior.

The temperature evolution from the wall at $X = 0.75$ and $X = 0.95$ are shown in figure 13. For brevity, density profiles are omitted here since their behavior closely reproduces those of the temperature. Again, the profiles depart further from the reference as the interface height increases. Nonetheless, for all simulations, the relative error on the temperature is very limited (below 1%) compared to the wall-resolved simulation.

Figure 14 shows the evolution of the dimensionless velocity u^+ with the wall distance y^+ obtained from the wall model at $X = 0.75$ and $X = 0.95$. The lower end of the curve (near-wall region) is fixed by the wall model, while the upper behavior is driven by the S-A RANS integration based on the values of the modeled region. The near-wall area shows a close match with respect to the reference RANS simulation. However, the integration in the RANS region yields a more significant error that increases with the wall distance. Overall, the profiles display a good agreement with the reference solution at $X = 0.75$, while only the wall model with an



(a) Skin friction coefficient C_f



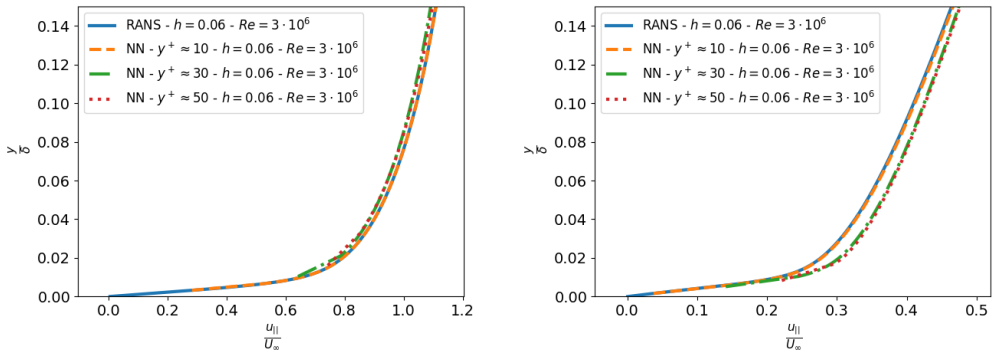
(b) Normalized error e of skin friction coefficient C_f

Figure 11: Bump interpolation case ($h = 0.06$ and $Re = 3 \cdot 10^6$). Skin friction coefficient C_f along the $X-$coordinate direction and its normalized error with respect to the wall-resolved RANS simulation. Wall distance at first RANS computed cell: $y^+ \approx 10$, $y^+ \approx 30$, $y^+ \approx 50$.

interface at $y^+ \approx 10$ well agrees with the RANS results in the downstream bottom area ($X = 0.95$). Again models with $y^+ \approx 30$ and $y^+ \approx 50$ interfaces show similar behavior.

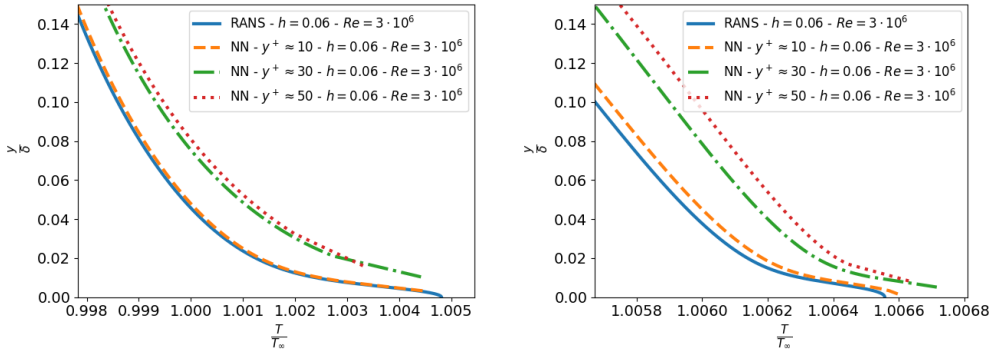
Figure 15 shows the dimensionless S-A variable $\tilde{\nu}^+$ at at $X = 0.75$ and $X = 0.95$. Again, the errors increase with higher wall distances at the first cell. The overall discrepancy appears limited at the top of the bump, while differences are found more relevant in the downstream bottom area of the bump. These differences explain the error behavior linked to the wall model strategy. The downstream bottom part of the bump displays a high p^+ value, which appears to strongly influence the dimensionless S-A variable $\tilde{\nu}^+$ compared to the modeled one (dashed line). The S-A variable in the presence of strong adverse pressure gradients tends to depart from a linear behavior for lower y^+ values, which explains the growing error observed for higher RANS interface (other modeling assumptions or approximations do not have an increasing error for higher y^+). This also explains why the global error from table 3 tends to increase for lower Re and higher h: higher p^+ values are expected with lower Reynolds numbers and higher bump heights, leading to more significant errors on $\tilde{\nu}^+$.

The error on the S-A variable impacts the RANS integration above the model interface. It



(a) (b)

Figure 12: Bump interpolation case ($h = 0.06$ and $Re = 3 \cdot 10^6$). Wall normal evolution of tangential velocity at $X = 0.75$ (top of the bump) (a) and at $X = 0.95$ (b). Wall distance at first RANS computed cell: $y^+ \approx 10$, $y^+ \approx 30$, $y^+ \approx 50$.



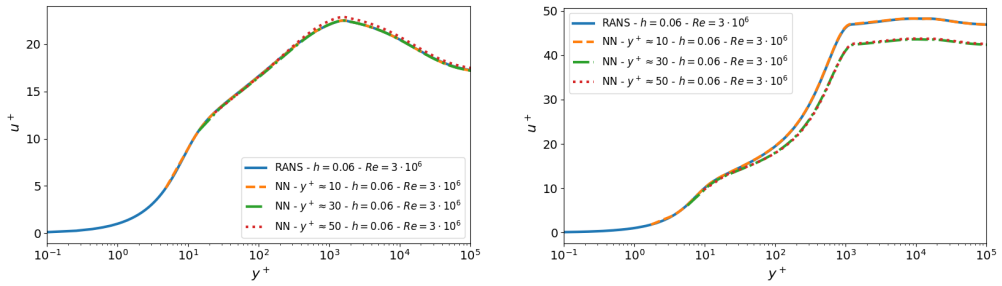
(a) (b)

Figure 13: Bump interpolation case ($h = 0.06$ and $Re = 3 \cdot 10^6$). Wall normal evolution of temperature at $X = 0.75$ (top of the bump) (a) and at $X = 0.95$ (b). Wall distance at first RANS computed cell: $y^+ \approx 10$, $y^+ \approx 30$, $y^+ \approx 50$.

strongly affects flow quantities sensed at the sample point. For this reason, a high error on $\tilde{\nu}^+$ leads to underestimated p^+ values at the sample point. Thus, the wall model converges toward overestimated solutions for the skin friction coefficient. The opposite situation happens at the top of the bump, where the poor modeling of $\tilde{\nu}^+$ for higher interface height yields underestimated C_f. Additionally, for lower Reynolds number configurations, the dimensionless S-A $\tilde{\nu}^+$ variable departs more rapidly from a linear behavior, which explains the observed error differences for different values of Re in Table 3 as the interface is moved upward.

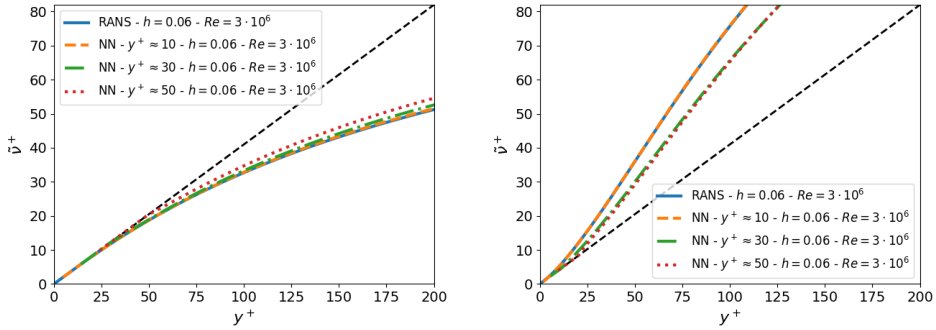
5.4. Extrapolation cases

Extrapolation capabilities of the model have been tested on unseen configurations during the training process with Reynolds number Re and a bump height h combinations beyond the range of values met in the training dataset. For almost all tested configurations, the error and conclusions are similar to those from the interpolation cases: the error on the C_f evolution, velocity, temperature, density, and eddy viscosity profiles is the highest near the top of the bump and



(a) (b)

Figure 14: Bump interpolation case ($h = 0.06$ and $Re = 3 \cdot 10^6$). Wall normal evolution of dimensionless velocity u^+ at $X = 0.75$ (top of the bump) (a) and at $X = 0.95$ (b). Wall distance at first RANS computed cell: $y^+ \approx 10$, $y^+ \approx 30$, $y^+ \approx 50$.



(a) (b)

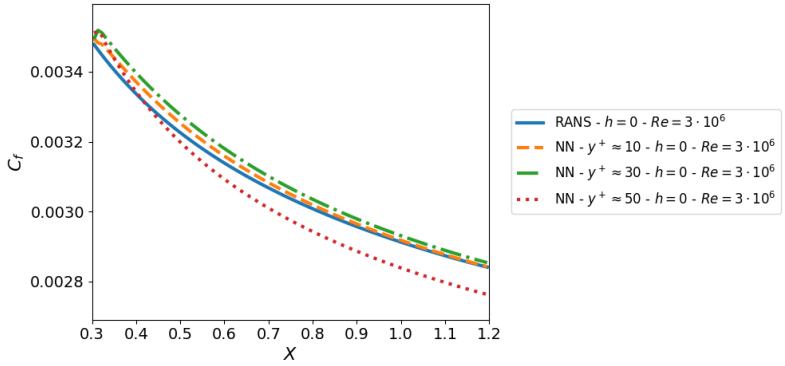
Figure 15: Bump interpolation case ($h = 0.06$ and $Re = 3 \cdot 10^6$). Wall normal evolution of dimensionless dimensionless S-A variable $\tilde{\nu}^+$ at $X = 0.75$ (top of the bump) (a) and at $X = 0.95$ (b). Wall distance at first RANS computed cell: $y^+ \approx 10$, $y^+ \approx 30$, $y^+ \approx 50$.

its downstream bottom area, with the same tendencies to over/underestimate the flow variables as previously. The performance assessment on the particular case of a zero-pressure flat plate ($h = 0$) is covered in section 5.4.1. For $h \neq 0$, the error becomes significant and may lead to convergence problems when the flow is close to separation. This is discussed in more detail in section 5.4.2.

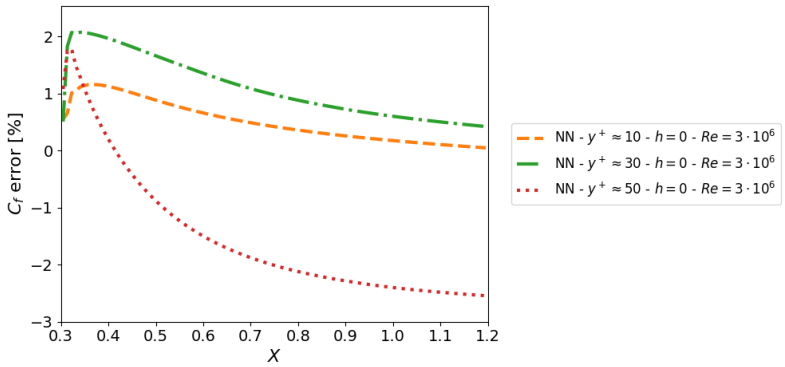
5.4.1. Flat plate case

The extrapolation capabilities of the model have been tested on the flat plate flow at $Re = 3 \cdot 10^6$. Figure 16 shows respectively the predicted skin friction coefficient C_f and its relative error e with respect to wall-resolved RANS (the normalizing value is the averaged value of C_f)

The model appears to be well adapted to quasi-equilibrium boundary layer since the skin friction coefficient is well reproduced by the neural network. The local normalized error does not exceed 3% even when the model interface is located at $y^+ \approx 50$; yet, better performances are achieved when the interface is located closer to the wall.



(a) Skin friction coefficient C_f

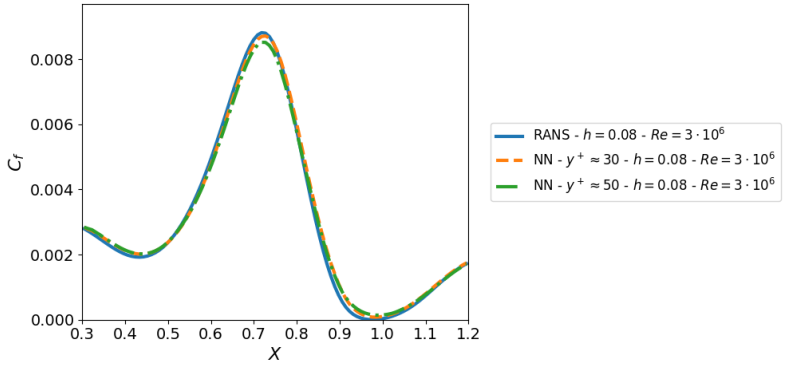


(b) Normalized error e of skin friction coefficient C_f

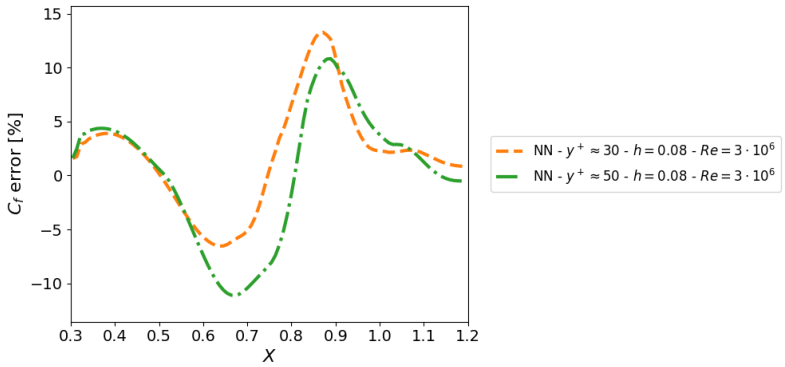
Figure 16: Flat plate case ($Re = 3 \cdot 10^6$). Skin friction coefficient C_f along the $X-$coordinate direction and its normalized error with reference to wall-resolved RANS simulation. Wall distance at first RANS computed cell: $y^+ \approx 10$, $y^+ \approx 30$, $y^+ \approx 50$.

5.4.2. Near separation case

This section focuses on the case $h = 0.08$ and $Re = 3 \cdot 10^6$. This is the test case with the strongest adverse pressure gradient, the flow being on the verge of separation. Thus, it is the most challenging case for our wall model, which was not designed to handle separated regions. Convergence problems appeared when the RANS interface was close to the wall, as reported in table 3. Figure 17 shows the results obtained for the cases that were able to converge (interface at $y^+ \approx 30$ and $y^+ \approx 50$). One may see that the evolution of C_f is qualitatively good. However, the error curve shows that the behavior is slightly erratic near the point where C_f approaches zero. Additionally, the figure shows the model's tendency to overestimate C_f in this region. As in section 5.3, this overestimation becomes less significant when the interface is closer to the wall. That explains why only the case with the interface at $y^+ \approx 10$ failed to converge: the modeling error on $\tilde{\nu}^+$ is smaller. Thus the encountered values of p^+ become closer to the reference, i.e., too high to be handled properly by the model. These results show that the wall model may handle reasonably large values of p^+ (such as those encountered when the interface is higher, underestimated due to the S-A variable modeling), but it reaches its limit for the case $y^+ = 10$.



(a) Skin friction coefficient C_f



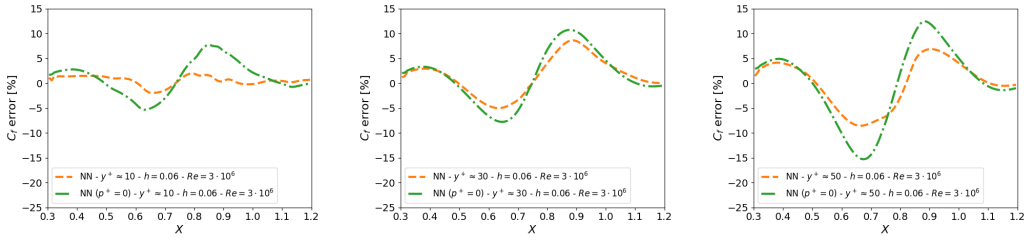
(b) Normalized error e of skin friction coefficient C_f

Figure 17: Near separation case ($h = 0.08$ and $Re = 3 \cdot 10^6$). Skin friction coefficient C_f along $X-$coordinate direction and its normalized error with reference to wall-resolved RANS simulation. Wall distance at first RANS computed cell: $y^+ \approx 50$.

5.4.3. Influence of dimensionless pressure gradient

To assess the importance of including or not the dimensionless pressure gradient p^+ as an input parameter to the wall law $u^+ = f(y^+, p^+)$, the selected interpolation case of the bump flow with $h = 0.06$ and $Re = 3 \cdot 10^6$ has been evaluated by imposing $p^+ = 0$ in the previously learned wall law during the CFD computation, i.e., $u^+ = f(y^+, p^+ = 0)$. Figure 18 shows the normalized error on skin friction coefficient C_f comparing the neural network fed with the wall distance y^+ and the dimensionless pressure gradient p^+ and the same neural network solely fed with the wall distance y^+. Overall, the pressure gradient-informed wall model manages to reproduce wall-resolved RANS simulation better. Significant differences between neural networks are detected at the top of the bump (i.e., $X = 0.75$) and downstream of the bump geometry (i.e., $X = 0.95$), where the highest pressure gradients are expected.

Table 4 shows 2-norm global errors on the skin friction coefficient C_f computed with wall distance and pressure gradient informed neural network $u^+ = f(y^+, p^+)$, and solely wall distance-informed neural network $u^+ = f(y^+, p^+ = 0)$. The 2-norm error of wall distance and pressure gradient informed neural network is extracted from table 3. The capabilities of the pressure



(a) $y^+ \approx 10$ (b) $y^+ \approx 30$ (c) $y^+ \approx 50$

Figure 18: Bump interpolation case ($h = 0.06$ and $Re = 3 \cdot 10^6$). Normalized error e of skin friction coefficient C_f obtained by solely wall distance informed neural network ($u^+ = f(y^+,\ p^+ = 0)$). Wall distance at first RANS computed cell: $y^+ \approx 10$, $y^+ \approx 30$, $y^+ \approx 50$.

gradient-fed neural network to estimate better local skin friction coefficient C_f is also confirmed on global 2-norm errors.

$h = 0.06$ - $Re = 3 \cdot 10^6$		
	$u^+ = f(y^+, p^+)$	$u^+ = f(y^+,\ p^+ = 0)$
$y^+ \approx 10$	2.4%	3.04%
$y^+ \approx 30$	3.69%	4.57%
$y^+ \approx 50$	4.0%	6.41%

Table 4: Relative 2-norm global error on the skin friction coefficient C_f of the wall modeled simulation with respect to the reference wall-resolved RANS simulation. Bump interpolation case ($h = 0.06$ and $Re = 3 \cdot 10^6$) for three different RANS interface positions. Comparison between the wall distance and pressure gradient informed neural network $u^+ = f(y^+, p^+)$ and solely wall distance informed neural network $u^+ = f(y^+)$. 2-norm error of wall distance and pressure gradient informed neural network is extracted from table 3

5.5. Mass conservation

As mentioned earlier, the present wall model strategy does not enforce the conservation of mass, which may be problematic for internal flow simulations. The mass loss in our configurations is evaluated by integrating the mass flux over the limits of the computational domain Ω (i.e., $X = 0.3$, $X = 1.2$ and $Y = 5$) by excluding the lower wall where the wall law is applied (since no mass flux exists there):

$$\oint_{\Omega} \rho \boldsymbol{u} \cdot \boldsymbol{n} d\Omega, \tag{28}$$

with $\boldsymbol{n}$ the exterior normal to the contour. This quantity is supposed to be null, and mass non-conservativity is characterized by the ratio

$$\frac{\oint_{\Omega} \rho \boldsymbol{u} \cdot \boldsymbol{n} d\Omega}{\rho_\infty U_\infty \delta}. \tag{29}$$

The lost mass rate is normalized with respect to the free-stream flow rate entering a section of height δ, which corresponds to the boundary layer thickness at the beginning of the evaluation zone (i.e., $X = 0.3$). The maximum loss equals 0.44% for the case $h = 0.07$ and $Re = 10^7$ with the RANS interface at $y^+ \approx 50$. For external aerodynamics, such an error may be acceptable; for internal flows, where the mass-flow rate may be an important quantity, such an error might become problematic.

6. Conclusion

This work presents a new deep learning-based approach to wall models for RANS simulations inspired by classical wall laws. The proposed wall models rely on wall dimensionless quantities, here the wall distance y^+ and the wall pressure gradient p^+, to reconstruct the dimensionless wall velocity u^+ profiles in wall-bounded region. The model provides embedded neural networks to the CFD solver code, which forces the primitive variables in modeled cells at a given interface near the wall, below which the RANS computation is disabled. It is equivalent to a Neumann boundary condition (whose exact value depends on the RANS velocity computed at the interface) applied to the conventional RANS region. This new approach may be interesting, particularly for RANS simulations with immersed boundary method simulations (IBM). One known shortcoming is that the proposed methodology could lead to conservativity problems, even though it was found to be negligible in the present work. However, further considerations of the problem are required to extend the validity domain of the wall model, which may be an exciting research direction for the future. For instance, more advanced treatments to impose the near wall behaviour of the turbulence model. The modeling of the Spalart-Allmaras variable in the present case or the wall normal velocity component could be explored.

The deep learning-based approach consists of a wall distance and pressure gradient-informed neural network trained on a dataset extracted from a fine wall-resolved RANS simulation of the flow over a bump. The training process has been performed with different Reynolds number conditions and pressure gradient levels based on the bump height. The neural network has been tested and compared with a fully resolved RANS simulation. The test cases were selected both from the training dataset and unseen configurations of the bump flow, characterized by a different combination of Reynolds number and bump height. The particular case of a flat plate was also included for testing. The benchmark cases have been run with varying interface heights to test the proposed wall model with various wall distances of the RANS interface. The wall distance and pressure gradient informed network yields accurate results for almost all the test cases and grids. However, the model underestimates the skin friction coefficient, with an error that increases as the interface height becomes higher.

One test case was particularly challenging. Convergence issues were found for nearly-separated cases with a strong adverse pressure gradient. This was expected since no particular treatment has been designed to enable the network to handle such cases. This may be addressed in the future by including a significant number of nearly-separated and separated cases in the training database. But this raises questions about the general strategy to follow: one may attempt to train one general network to handle all the cases or several specialized networks coupled with an automatic way of selecting which one to use. The latter approach may require exploring clustering techniques.

Even though relatively simple geometries characterize the test cases, this work highlights the potential of neural networks for wall-bounded region modeling. In particular, searching a relation between non-dimensional quantities mechanically gives the network extrapolation capabilities, enabling, for instance, simulation for Reynolds number beyond the range considered for training. This would not be possible if one tries to estimate dimensional quantities.

The present results may be the starting point for further studies and investigations required to overcome the issues encountered during this work and extend the validity domain of these deep-learning-based wall models to more complex problems, even in the presence of significant pressure gradients and flow separation. A methodology has been proposed to optimize the computational cost of the network. But this question may require more extensive attention and future

work since it is a critical point in the context of wall models for CFD.

Acknowledgment

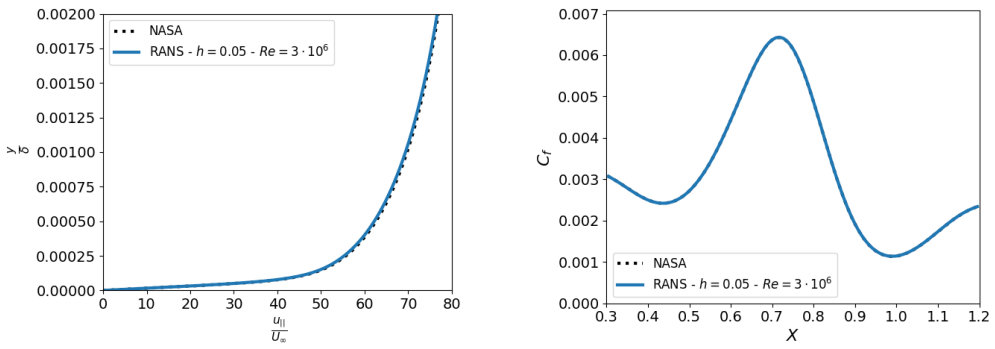
This work has been done as part of a PhD co-funded by ONERA and Region Nouvelle Aquitaine. The authors would like to thank Kevin Doria for his useful preliminary work on the subject, and Lucas Manueco and Marc Terracol for the insightful discussions related to this paper.

Appendix A. Validation of bump case setup

The bump case is inspired by a RANS simulation well documented by NASA in the context of a study about turbulent modeling techniques for RANS simulation [10]. Validation of the case is achieved by a solution comparison with NASA simulation of the flow over a bump characterized by an height h of 0.05. A Reynolds number $Re = 3 \cdot 10^6$ and a Mach number $M = 0.2$ are considered to match the NASA case. The free-flow temperature is $T = 300\,K$.

The original NASA solution was obtained on a 1409x461 structured grid, through the compressible cell-centered code CFL3D with the Spalart-Allmaras turbulence model. Roe's Flux Difference Splitting and a UMUSCL upwind approach are used for the computation, while a first-order upwinding is adopted for the advective terms of the turbulence model.

In figure A.19, the tangential velocity at the top of the bump ($X = 0.75$) and skin friction coefficient C_f are compared for the NASA computation and for the dataset extraction case. The results show that both simulations match closely, validating the presented case setup.



(a) (b)

Figure A.19: Validation of bump case setup. Comparison of NASA results and wall-resolved RANS simulation of the flow over a bump with an height $h = 0.05$. Tangential velocity to surface at $X = 0.75$ (top of the bump) (a) and skin friction coefficient C_f over the bump (b).

Appendix B. Optimization of neural network structure

Since the structure of the neural network involved in the wall model strongly determines its computational cost, an optimization process has been carried out to find optimal network architectures with respect to the accuracy/cost trade-off. This appendix presents the followed

methodology and gives the main insights that emerged from the optimization results. They may serve as guidelines for future network design. In particular, it aims at answering two questions: how deep should the network be, and what would be the proper width of each layer? To do so, we start with a given network architecture with L_h hidden layers containing each a rather large number of neurons N_L, and we define an optimization process that deactivates the least useful neurons. This yields a network architecture where the layer width is not constant anymore and has been reduced with minimal impact on the accuracy. The process has been repeated for several numbers of layers L_h, and is described below.

An optimization network is created by adding a gate directly after every hidden layer neuron. These additional nodes multiplies their input by a scalar (no activation function and no bias) which is a trainable parameter of the network. These gates are then densely connected to the downstream hidden layer. Therefore, suppressing a neuron from the network is equivalent to setting its gate's weight to zero. A schematic diagram of such a gated neural network is given in figure B.20.

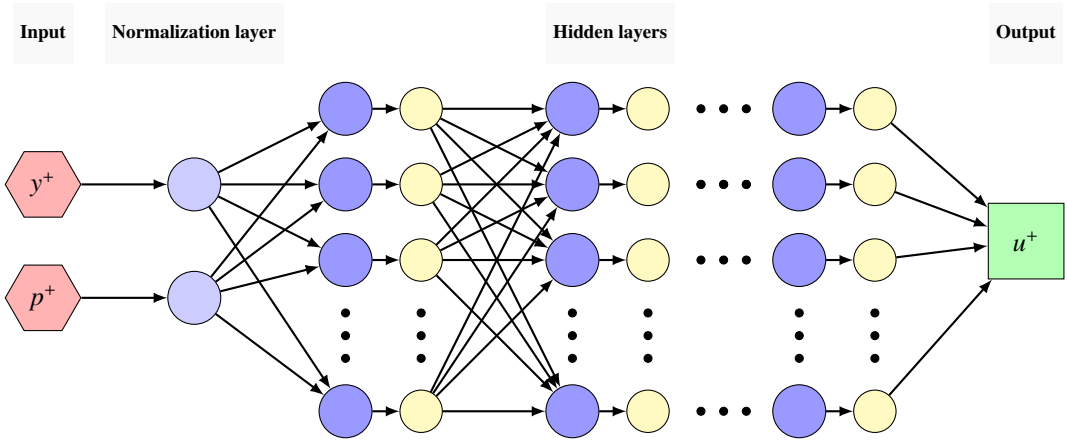


Figure B.20: Schematic diagram of the feedforward neural network with multiple hidden layers used for structure optimization process.

The initial network, composed of L_h layers and 16 neurons each, is first trained without gates following the procedure described in Appendix C to minimize the loss ϵ defined in equation (22). Then, gates are added after all hidden-layer neurons and their weight is initialized to 1. The training is launched again (by keeping the same optimization parameters) with the following modified loss ϵ':

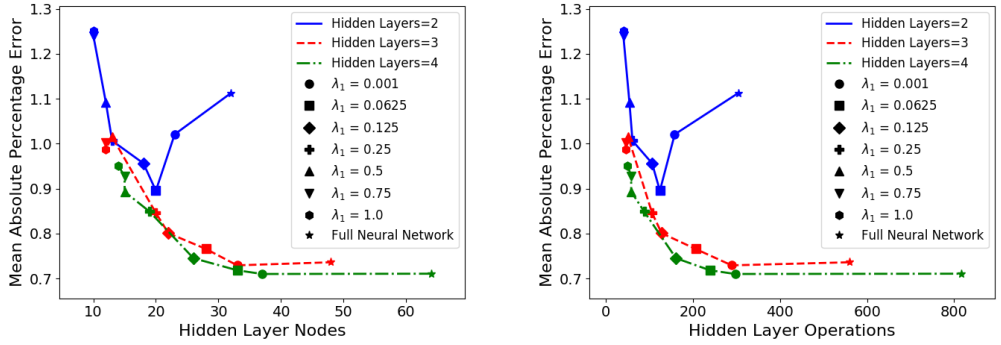
$$\epsilon' = \epsilon + \frac{\lambda_1}{N^S} \left\| w_o^L \right\|_1 , \tag{B.1}$$

where w_o^L is a vector containing all the weights of the gates and N^S the number of training samples. This extra L1 regularization term promotes sparsity and pushes toward setting some gates to zero. During the optimization, when a gate's weight drops below 0.01, it is permanently set to zero. Other threshold values, ranging from 0.01 to 0.1 have been tested. The highest values have been found to significantly affect the NN's accuracy. The penalization coefficient λ_1 controls the cost/accuracy trade-off by balancing the sparsity penalization with the rest of the loss (that promotes accuracy). Different values between 0.001 and 1 have been tested (the

higher λ_1, the more deactivated neurons) for different network depths L_h (2, 3, and 4). Once the optimization is converged, one gets the width of each layer. The resulting architecture is then evaluated through the mean absolute percentage error (MAPE) defined as

$$\text{MAPE} = \frac{1}{N^S} \sum_{i=1}^{N^S} \left| \frac{f_{NN}(y_i^+, p_i^+) - u_{i,ref}^+}{u_{i,ref}^+} \right|. \tag{B.2}$$

The error is thus monitored against the number of remaining hidden layer neurons and the number of operations in hidden layers (an operation here is considered as a link between two neurons, which corresponds to computing a quantity of the form $w_{i,j}^{(L)} O_j^{(L-1)} + b_i^{(L)}$, see equation (17). This provides Pareto fronts which are shown in figure B.21.



(a) (b)

Figure B.21: Results of structure optimization process. Mean Absolute Percentage Error (MAPE) given by neural networks with reference to the whole dataset. MAPE as function of hidden layers nodes (a) and MAPE as function of hidden layers operations (b).

Overall, the three and four hidden layer neural networks showed similar levels of accuracy, while two hidden layer networks displayed significantly lower performances. As expected, a too low number of remaining nodes results in a steep drop of accuracy.

The four hidden layer neural network trained with an L1 penalization coefficient of $\lambda_1 = 0.001$ (green circle) is retained as final architecture, motivated by a willing to favor accuracy over cost for this first methodological paper. This neural network, composed by 10, 10, 10 and 7 nodes in the hidden layers, allows to more than halve the number of operations performed by the starting network given by four hidden layers of 16 nodes each while maintaining the same level of accuracy (all lighter architectures yield a decreased accuracy). Note that the other obtained architectures (not shown here) systematically have more neurons in the first layers than in the final ones. This may therefore be considered as a general guideline for network design when one wants to reduce the network size without carrying out a similar optimization task.

Appendix C. Training process of neural networks

This section details the training process performed on the neural network and gives all information needed to reproduce the results from the paper.

Training is performed using the Adam algorithm to minimise the NN loss given in equation 22, based on the mean absolute logarithmic error with an L2 regularization term to penalize high valued weights and biases. Details on the learning rate are provided below. The training dataset consists of the 85% of the available data, while the validation dataset is composed of the remaining 15%. Evaluating the training vs. validation loss functions is a standard way in machine learning to introduce a stopping criterion for training to prevent the overfitting of the model. During each epoch, the NN and the descent algorithm are fed with the complete set of training data. Training data are given using mini-batches of 16 samples. Using a standard validation stopping criterion as mentioned above, the training of the neural network required 2004 epochs. The stopping criteria, based on the validation loss, stops the training if the loss value does not decrease of 10^{-2} during the last 400 iterations. At the end of the training process, weights and biases which gave the lowest value of validation loss during the training process are selected to build the neural network. The evolution of training and validation loss during the training process is given in figure C.22.

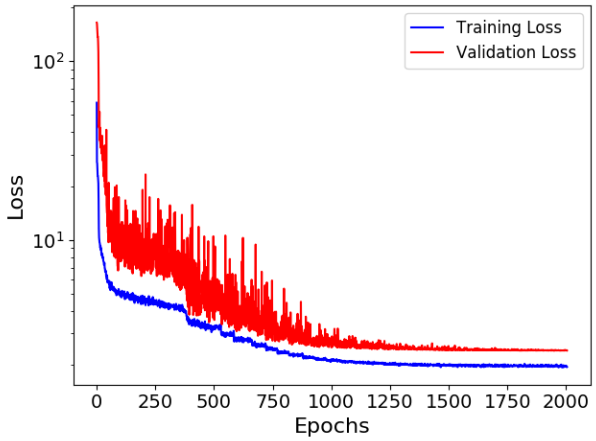


Figure C.22: Training and validation loss evolution during training process.

During the NN training, the convergence of the optimization algorithm is helped by a reduction of the learning rate over the advancement of epochs, as perceptible from figure C.22. The starting learning rate for the Adam algorithm is set to 0.001. Loss oscillations visibly reduce as the learning rate is decreased. The learning rate reduction is driven by the loss value. The learning rate is reduced by steps of 20% until a minimum value of 10^{-8} is achieved. Each reduction step is performed if the mean loss value over the last 40 epochs does not vary.

Appendix D. Boundary layer thickness definition

One common way to define the boundary layer thickness δ is as the distance to the wall where the flow velocity inside the boundary layer reaches 99% of the external velocity magnitude or asymptotic velocity, i.e. $U(\delta) = 0.99\,U_e$. However, when pressure gradients in the streamwise direction are present, external boundary layer velocity varies along the considered flow, consequently the previous definition becomes not adapted. Therefore, along the bump, another better-suited definition of δ is needed.

The boundary layer edge can thus be estimated through an another estimation method which relies on the boundary layer definition. The following details the approach from [11]. At boundary layer edge, the shear stress τ and the flow vorticity Ω must become small. The total shear stress can be defined as

$$\tau = \tau_l + \tau_t = \mu \left|\Omega\right| + \mu_t \left|\Omega\right|, \tag{D.1}$$

where the laminar component of shear stress τ_l and the turbulent shear stress τ_l are computed through the laminar and turbulent viscosity, respectively μ and μ_t.

The boundary layer thickness δ is defined such as

$$\delta = \min\left(\delta_\Omega,\, \delta_\tau\right), \tag{D.2}$$

where δ_Ω and δ_τ are the wall normal distances where ϵ_Ω and ϵ_τ reach respectively small empirical values as 0.001 and 0.015. The two parameters ϵ_Ω and ϵ_τ are defined as it follows

$$\epsilon_\Omega = \frac{|\Omega|}{|\Omega|_{max}} \quad \epsilon_\tau = \frac{|\tau|}{|\tau|_{max}}. \tag{D.3}$$

As an example, the estimated boundary layer thickness for the wall-resolved RANS case with $h = 0.06$ and $Re = 3 \cdot 10^6$ is represented over velocity magnitude contours in figure D.23.

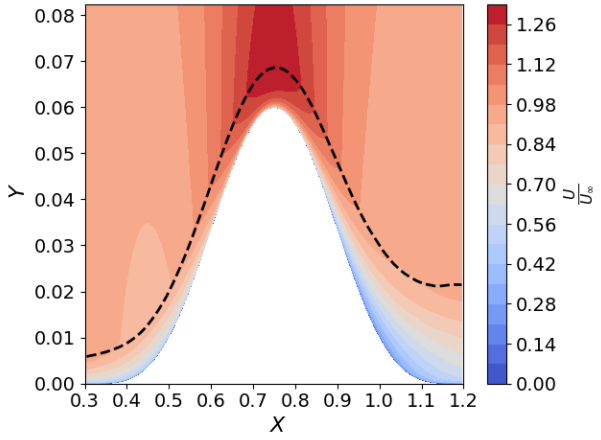


Figure D.23: Velocity magnitude contours for the wall-resolved RANS case with $h = 0.06$ and $Re = 3 \cdot 10^6$. Boundary layer thickness thickness represented with dashed line.

References

[1] U. Piomelli, Wall-layer models for large-eddy simulations, Progress in Aerospace Sciences 44 (2008) 437–446. doi:10.1016/j.paerosci.2008.06.001.

[2] R. Mittal, G. Iaccarino, Immersed boundary methods, Annu. Rev. Fluid Mech. 37 (2005) 239–261.

[3] G. Kalitzin, G. Medic, G. Iaccarino, P. Durbin, Near-wall behavior of rans turbulence models and implications for wall functions, Journal of Computational Physics 204 (2005) 265–291. URL: https://www.sciencedirect.com/science/article/pii/S0021999104004164. doi:https://doi.org/10.1016/j.jcp.2004.10.018.

[4] N. Afzal, Power law and log law velocity profiles in fully developed turbulent pipe flow: Equivalent relations at large Reynolds numbers, Acta Mechanica 151 (2001) 171–183. doi:10.1007/BF01246916.

[5] S. Kawai, K. Asada, Wall-modeled large-eddy simulation of high Reynolds number flow around an airfoil near stall condition, Computers and Fluids 85 (2013) 105–113. URL: http://dx.doi.org/10.1016/j.compfluid.2012.11.005. doi:10.1016/j.compfluid.2012.11.005.

[6] F. Capizzano, Coupling a wall diffusion model with an immersed boundary technique, Aiaa Journal 54 (2016) 728–734.

[7] Z. Zhou, G. He, X. Yang, Wall model based on neural networks for les of turbulent flows over periodic hills, Physical Review Fluids 6 (2021) 1–30. doi:10.1103/PhysRevFluids.6.054610. arXiv:2011.04157.

[8] M. Ivan, L. G. Jean-Marie, S. P. Christophe, Benoit, L. Sam, R. Thomas, B. Didier, J. Antoine, S. Alexandre, Fast: A compressible flow solver python package, 2021. URL: https://w3.onera.fr/FAST/.

[9] P. Roe, Approximate riemann solvers, parameter vectors, and difference schemes, Journal of Computational Physics 43 (1981) 357–372. URL: https://www.sciencedirect.com/science/article/pii/0021999181901285. doi:https://doi.org/10.1016/0021-9991(81)90128-5.

[10] C. Rumsey, 2d bump-in-channel verification case, 2021. URL: https://turbmodels.larc.nasa.gov/bump.html.

[11] J. Cliquet, R. Houdeville, D. Arnal, Application of laminar-turbulent transition criteria in navier-stokes computations, AIAA Journal 46 (2008) 1182–1190. URL: https://doi.org/10.2514/1.30215. doi:10.2514/1.30215. arXiv:https://doi.org/10.2514/1.30215.

[12] B. W. van Oudheusden, Some classic thermal boundary layer concepts reconsidered (and their relation to compressible couette flow), in: G. E. A. Meier, K. R. Sreenivasan, H.-J. Heinemann (Eds.), IUTAM Symposium on One Hundred Years of Boundary Layer Research, Springer Netherlands, Dordrecht, 2006, pp. 425–434.

[13] H. Werner, H. Wengle, Large-eddy simulation of turbulent flow over and around a cube in a plate channel, in: F. Durst, R. Friedrich, B. E. Launder, F. W. Schmidt, U. Schumann, J. H. Whitelaw (Eds.), Turbulent Shear Flows 8, Springer Berlin Heidelberg, Berlin, Heidelberg, 1993, pp. 155–168.

[14] D.-A. Clevert, T. Unterthiner, S. Hochreiter, Fast and accurate deep network learning by exponential linear units (elus), 2015. URL: https://arxiv.org/abs/1511.07289. doi:10.48550/ARXIV.1511.07289.

[15] Y. Park, Concise logarithmic loss function for robust training of anomaly detection model (2022). URL: https://arxiv.org/abs/2201.05748. doi:10.48550/ARXIV.2201.05748.

[16] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014. URL: https://arxiv.org/abs/1412.6980. doi:10.48550/ARXIV.1412.6980.

[17] Tensorflow: An end-to-end open source platform for machine learning, 2022. URL: https://www.tensorflow.org/.

[18] INRIA Tropics team and INRIA Ecuador team, Tapenade: An automatic differentiation engine, 2021. URL: https://team.inria.fr/ecuador/en/tapenade/.