Deep learning-based wall models for aerodynamic simulations: A new approach inspired by classical wall laws

Michele Romanelli, Samir Beneddine, Heloise Beaugendre, Denis Sipp, Ivan Mary, Michel Bergmann

Deep learning-based wall models for aerodynamic simulations: A new approach inspired by classical wall laws

Michele Romanelli[1], Samir Beneddine[1], Héloïse Beaugendre [2], Denis Sipp [1], Ivan Mary [1], and Michel Bergmann [3]

[1]Département aérodynamique, aéroélasticité et acoustique (DAAA), ONERA, Université Paris Saclay, F-92190 Meudon, France
[2]Bordeaux INP, Université de Bordeaux, IMB, UMR 5251, F-33400 Talance, France, and Equipe-projet Cardamom, Inria Bordeaux-Sud Ouest, F-33400 Talance, France
[3]Université de Bordeaux, IMB, UMR 5251, F-33400 Talence, France, and Equipe-project Memphis, Inria Bordeaux-Sud Ouest, F-33400 Talence, France

Abstract

A new deep learning-based approach to wall models is proposed here. The wall model is inspired by classical wall laws, which rely on wall dimensionless quantities, such as the wall distance y^+, the wall velocity u^+, the wall friction velocity u_τ and the wall pressure gradient p^+. This paper describes the model formulation, the implementation in a CFD code and demonstration simulations. The numerical results are presented for a flat plate boundary layer at zero pressure gradient and a bump case. The neural network models are trained on RANS datasets (Spalart-Allmaras model) from these configurations that account for different levels of adverse pressure gradient and Reynolds numbers. Finally, the wall model is applied and tested on RANS simulations with under-resolved meshes at the wall.

1 Introduction

The availability of reliable and accurate wall laws is one of the main challenges of modern CFD. The adoption of wall models has become unavoidable for Large Eddy Simulations (LES) since any flow representative of aeronautical applications remains out of reach even with the most modern supercomputers due to the need to solve all turbulent scales in the boundary layers [7]. Modeling the near-wall region is also needed in Reynolds-Averaged Navier-Stokes simulations (RANS), especially when using immersed boundary method (IBM), one major tool in the highly competitive industrial design environment that requires fast and reliable simulations. The present work focuses on the latter by proposing new data-based wall models for RANS simulations.

In wall-bounded flows, a large number of computational cells are generally used to resolve the boundary layers to correctly represent the strong gradients established in the wall-normal direction. Usually, wall-resolved RANS requires the first computational cell placed in the viscous sublayer of the boundary layer at a wall distance near $y^+ = 1$. Wall models allow for loosening these grid requirements, thus reducing the grid size. The gain in computational cost is not only linked to this reduction but also a lower aspect ratio of near-wall cells, resulting in a reduced computational stiffness [3].

Most of the wall functions rely on the generality of the wall law valid for turbulent boundary layers, which shows that under many flow conditions, the main shape of the solution between the wall and the outer edge of the logarithmic layer is mainly invariant if appropriate scaling is used. Adaptation of the general wall-bounded flows behavior is, however, required in the presence of pressure gradients. In recent decades, the high demand for wall models has led to an evolution from analytical equilibrium models toward more sophisticated wall laws, valid even in the presence of significant pressure gradients and separated flows [7, 4]. However, even the most recent approaches still lack generality and robustness. Given the complexity of wall flows and the wide range of phenomena that can be modeled with the flexibility of neural networks (NN), innovative deep learning-based methods show undeniable potential in their application to wall flow modeling. The proposed wall model aims to integrate the flexibility of neural networks to model wall-bounded flows.

This work is in the continuity of recent articles: neural networks have already been used to model wall flows, for instance, by [13], but a clear methodology does not emerge regarding the choice of input and output variables that are exchanged with neural networks. In this work, the chosen approach complies with classical wall models. Dimensionless wall quantities are chosen to feed the neural networks, such as:

$$u^+ = \frac{u_\parallel}{u_\tau} \qquad y^+ = \frac{\rho y u_\tau}{\mu} \qquad p^+ = \frac{\nu}{\rho_\infty u_\tau^3}\frac{\partial p}{\partial S}, \tag{1}$$

where y^+ is the dimensionless wall distance, u^+ dimensionless flow velocity, p^+ is dimensionless pressure gradient along streamwise direction at the wall surface, $u_\parallel$ is the wall tangent flow velocity component, and y is the wall distance. These dimensionless quantities are defined by the shear velocity u_τ, computed using the skin friction τ_w as:

$$u_\tau = \sqrt{\frac{\tau_w}{\rho}} \quad \text{with} \quad \tau_w = \mu \left.\frac{\partial u_\parallel}{\partial y}\right|_{y=0}. \tag{2}$$

Our approach differs from [13] by choice of the adimensioning factors, model inputs, and outputs. A more conventional set of wall model inputs and outputs are here chosen to conform with the classical set of parameters of analytical equilibrium models instead of directly estimating the wall friction τ_w from flow data far from the wall. We believe that this approach is bound to provide more general results than a direct estimation of dimensional quantities.

In the present study, the capabilities of two different neural networks are compared. The first one is only wall distance informed (it estimates the relation $u^+ = f(y^+)$), and the second one is also trained with pressure gradient data (providing a relation $u^+ = f(y^+, p^+)$). The paper aims to detail the practical implementation of such an approach and provides a first demonstration of its capability for RANS simulations.

The article is organized as follows: the first section presents the flow configurations considered in the paper. Then, the numerical strategy proposed in this work is detailed in section 3. Section 4 then provides details on the neural network architectures and their training and implementation within the CFD code used for this article. The last section presents the results obtained with these data-based models before concluding.

2 Flow configurations and datasets

For this paper, the proposed wall model is designed to be implemented in RANS simulations using the Spalart-Allmaras (S-A) turbulence model. Therefore, all data and models in the following correspond to the S-A RANS equations.

2.1 CFD solver

The results of this paper are based on ONERA's finite volume structured cell-centered solver FASTs [1], used for both the data generation and the test of the wall models. For the present article, data are generated using a Roe flux scheme [8] with a third-order spatial discretization. The solution of the S-A RANS equations is then computed using a local time-stepping strategy, with a CFL value imposed to 20. The convergence of the results is monitored by ensuring a decay of the residuals by at least six orders of magnitude.

2.2 Flat plate case

As said in the introduction, this paper compares a model that only accounts for the wall distance with a model that also uses pressure gradient information. Therefore, two cases are considered, each relevant to one of these approaches. The first dataset is a collection of flow profiles obtained from a fine wall-resolved RANS simulation over a flat plate at a Mach number $Ma = 0.2$ and a Reynolds number $Re = 5 \cdot 10^6$. The free-flow temperature is $T_\infty = 300$K, and there is no imposed adverse pressure gradient. The length of the considered plate is equal to 3.5m, and the Reynolds number is calculated based on unitary flat plate length. The simulation domain extends from $x = -0.33$m to $x = 3.5$m and $y = 0$m to $y = 1$m, as shown in figure 1. Training data are only extracted for a well-established, fully turbulent boundary layer. This is achieved by limiting the extraction zone used to build the dataset between $x = 1.5$m and $x = 3.5$m.

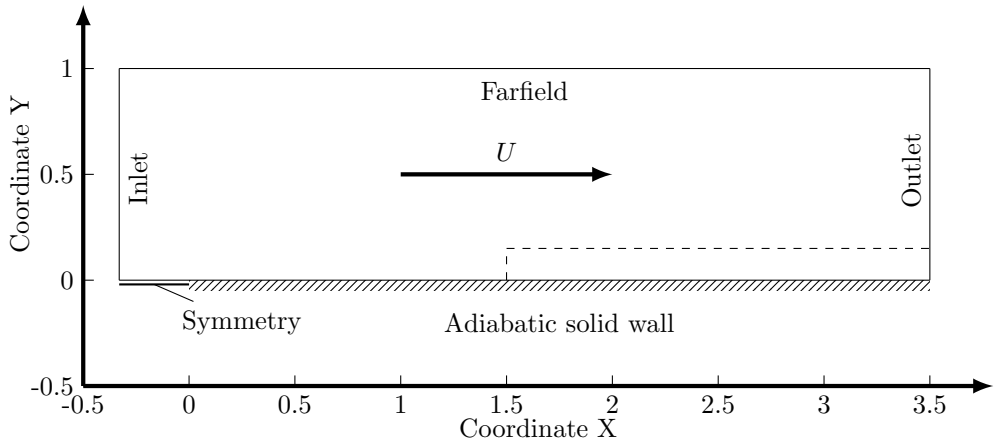


Figure 1: Flat plate case: domain and boundary conditions for turbulent flow over a flat plate. The training dataset extraction is bounded by the dashed line.

The Inlet boundary condition provides a uniform flow at the domain's entry, while a uniform pressure condition is applied for the outflow. In the upper boundary, a far-field characteristic-based condition that avoids numerical reflection is used. The leading edge of the flat plate is placed at $x = 0$m, following a symmetry boundary condition between $x = -0.33$m and $x = 0$m. An adiabatic wall condition is used for the flat plate. All boundary conditions are represented in figure 1.

Training data are obtained through a wall-resolved simulation using a fine Cartesian grid (first-cell center distance from the wall between $y^+ = 0.13$ and $y^+ = 0.2$ all along the flat plate), ensuring grid-convergence. A growing ratio of 2% is used in the wall-normal direction. In the streamwise direction, the refined grid at the leading edge gets coarser along the $X-$coordinate direction, while a uniformly spaced grid is used from $x = 1.5$m.

2.3 Bump Case

The second configuration displays both favorable and adverse pressure gradients. Data are obtained from a set of fine wall-resolved RANS simulations over a bump at $Ma = 0.2$ with Reynolds numbers $Re = 10^6$, $Re = 3 \cdot 10^6$, and $Re = 6 \cdot 10^6$ with $T_\infty = 300$K. The Reynolds number is calculated based on unitary length. The bump function is given by:

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

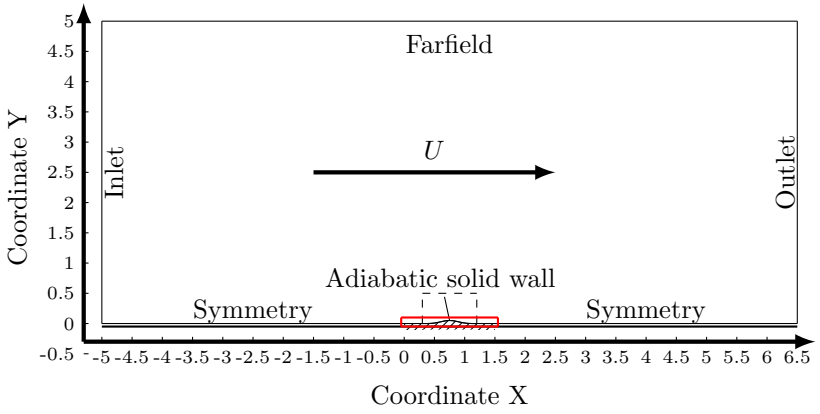
where the maximum height h of the bump is set to 0.05m, 0.06m, and 0.07m for the data extraction cases, the chosen bump height is selected to yield a weak and moderate pressure gradient along the geometry, without inducing flow separation.

Note that the case $h = 0.05$m and $Re = 3 \cdot 10^6$ corresponds to a documented computation from NASA [9], from which the present numerical setup has been validated. More details about this case setup may be found in appendix A.

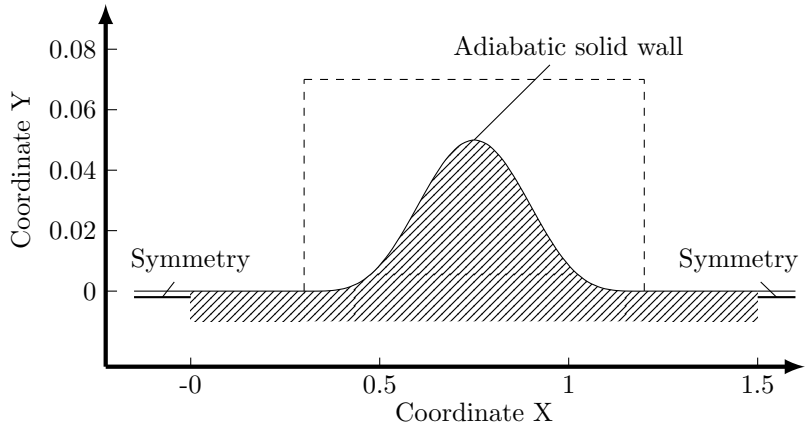
The dataset consists in nine simulations which combine all possible parameters. Eight of them are used for the training process, while the remaining one is used to test the interpolation capabilities of the neural networks. The test case corresponds to the NASA configuration (i. e. $h = 0.05$m and $Re = 3 \cdot 10^6$).

Equation 3 is applied for $x \in [0\text{m}, 1.5\text{m}]$, which corresponds to the adiabatic wall extent. The simulation domain ranges in $x \in [-5\text{m}, 6.5\text{m}]$ and $y \in [0\text{m}, 5\text{m}]$, as shown in figure 2, where the chosen boundary conditions may also be seen.

Symmetry boundary conditions precede and follow the adiabatic wall. Training data are only extracted for a well-established fully turbulent boundary layer. This is achieved by limiting the extraction zone for the dataset along the bump geometry between $x = 0.3$m and $x = 0.9$m.



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



(b) Details of the bump. The training dataset extraction is bounded by the dashed lines.

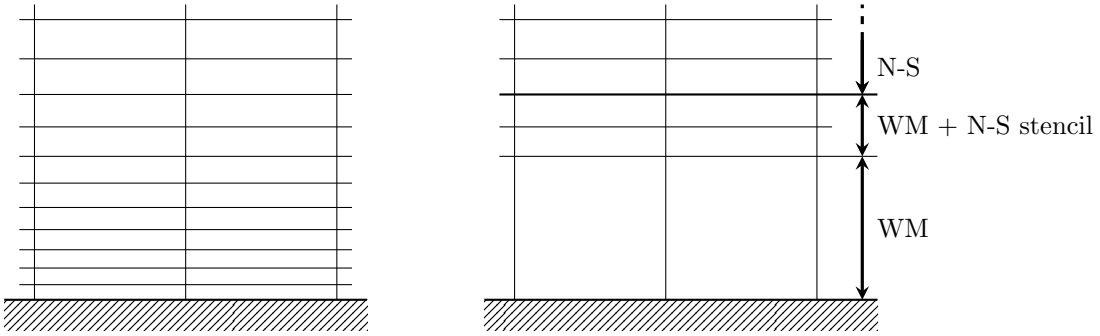
Figure 2: Bump case: Simulation domain, boundary conditions and geometry details.

A fine structured grid is used to obtain the training data. The first-cell center distance from the wall is located between $y^+ = 0.6$ and $y^+ = 0.01$ along the bump for all the different cases for the training database. A growing ratio for the mesh of 2% is used in the wall-normal direction. In the streamwise direction, a refined grid at the leading edge and the trailing edge of the wall surface is used, while a uniformly spaced grid is used from $x = 0.3$m to $x = 0.9$m.

3 Wall model strategy

Contrary to the existing approach defined in the recent literature [13], a model able to estimate the flow profile from the wall entirely is here sought. Instead of solving the RANS equations in the near-wall region, the model allows to directly impose the primitive variables such as density ρ, velocity components u^i, temperature T, and the transported variable of the S-A model $\tilde{\nu}$. The simulation domain can thus be split into two different layers. The first one lies just above the wall, where the flow solution is modeled, and the RANS computation is disabled. The second layer consists of the classical RANS-resolved simulation domain.

The enforcement of the wall law is set to a pre-defined number of cell layers in the wall's normal direction just before the model interface. Above it, the conventional integration of the RANS equations takes over. The number of needed cells in the modeled part depends on the numerical stencil of the solver. Using the wall model, we impose the value of "ghost" cells in the modeled region such that for the solver, this ends



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

(b) Grid structure used for RANS simulations coupled with wall modeling and trimmed grid

Figure 3: Example of grid structure for RANS simulations in the near-wall region. Comparison between a standard wall refined grid and a "trimmed" grid.

up as setting a specific Neumann boundary condition (since it imposes the exchanged flux) at the interface between model-driven and RANS-resolved cells. Therefore, only the cells within the numerical stencil of the cells of the RANS regions are actually needed. For this strategy to work, the modeled cells at the interface should not be too coarse to yield a correct estimation of the local gradients.

In practice, the grids are obtained by "trimming" resolved RANS grids, obtained by removing the first cells near the wall, up to the desired y^+ location where the RANS computation takes place, then restoring the n cells corresponding to the stencil size of the numerical scheme.

A schematic example of grid structure is given in figure 3. A classic grid for the wall-resolved simulation (a) is compared to the trimmed grid (b) used with the presented wall model. The trimmed grid consists mainly of the model-driven zone and the RANS domain. At the interface, two cell layers in the modeled zone contribute to the RANS-resolved computation through the numerical stencil of the cells near the interface. Below them, there is no refinement constraint on the cell size since all lower cells that would be kept are not actually used for the computation of the RANS region. A single coarse cell can be employed as in the example.

The solution of the discrete RANS equations is associated with an increasing numerical error with decreasing grid resolution [3]. Trimmed grids decrease the number of solution grid points in modeled regions without affecting grid resolution in RANS integrated domain. Consequently, a correct evaluation of tested wall functions is achieved, since RANS numerical error induced by grid coarsening is limited.

Note that the adopted approach may potentially be non-conservative since the RANS computation is disabled in the wall-modeled region. Mass, momentum, and energy transport are thus implicitly driven by the proposed modeling strategy and hypothesis. Further discussion regarding these issues is proposed in the conclusion.

The following sections provide details regarding the computation of the velocity components (section 3.1), the modeling of flow temperature and density (section 3.2) and of the S-A variable (section 3.3) by the wall model.

3.1 Velocity components

The integration of a neural network in the wall model allows for reconstructing the evolution of the dimensionless velocity u^+ from the wall to obtain wall tangent velocity components $u_{\parallel}$ for the modeled cell layers of the computational domain. The following paragraphs focus only on the pressure gradient informed neural network for brevity. However, the more straightforward formulation for only wall distance informed neural network can easily be obtained by neglecting pressure gradient related terms.

The neural network is trained to approximate the function:

$$u^+ = f\left(y^+, p^+\right) \tag{4}$$

or, in an extended form:

$$\frac{u_\parallel}{u_\tau} = f\left(\frac{\rho y u_\tau}{\mu}, \frac{\nu}{\rho_\infty u_\tau^3}\frac{\partial p}{\partial S}\right). \tag{5}$$

The neural network is thus capable of providing dimensionless wall velocity u^+ from y^+ and p^+. The skin friction velocity u_τ can consequently be computed from flow data sensed at a sampling point S far from the wall (i.e., in the computed section of the domain, above modeled layers). An iterative Newton-Raphson method is employed to compute u_τ satisfying equation 5 at the sampling point. The shear velocity u_τ is obtained iteratively through the equation:

$$u_\tau^n = u_\tau^{n-1} - \frac{g\left(u_\parallel^S,\, y^S,\, \frac{\partial p}{\partial S},\, u_\tau^{n-1},\, \rho,\, \rho_\infty,\, \mu\right)}{g'\left(u_\parallel^S,\, y^S,\, \frac{\partial p}{\partial S},\, u_\tau^{n-1},\, \rho,\, \rho_\infty,\, \mu\right)}, \tag{6}$$

where function g is defined as:

$$g = f\left(\frac{\rho y^S u_\tau}{\mu}, \frac{\nu}{\rho_\infty u_\tau^3}\frac{\partial p}{\partial S}\right) - \frac{u_\parallel^S}{u_\tau} \tag{7}$$

and its derivative g':

$$g' = \frac{y^S \rho}{\mu}\cdot\frac{\partial f}{\partial y^+} - \frac{3\nu}{\rho_\infty u_\tau^4}\frac{\partial p}{\partial S}\cdot\frac{\partial f}{\partial p^+} + \frac{u_\parallel^S}{u_\tau^2}. \tag{8}$$

Note that data from the sampling point S is insufficient to solve this equation. Additional quantities are also required to compute u_τ: the density ρ and temperature T (from which the molecular viscosity μ and kinematic viscosity ν may be deduced using Sutherland's law). The modeling of these two quantities is detailed in the next section. The pressure gradient $\frac{\partial p}{\partial S}$ is obtained through a projection along the streamwise direction at the wall surface of the pressure gradients in i directions for the first wall cell.

The partial derivatives $\frac{\partial f}{\partial p^+}$ and $\frac{\partial f}{\partial p^+}$ are obtained through algorithmic differentiation of trained neural networks (more details in section 4). The shear velocity computation then consists of finding the u_τ value that voids g function. The iterative process is thus applied until the residual value of g approaches zero within a given tolerance (set in the following to 10^{-9}).

The iterative process requires an initial guess for the skin friction velocity u_τ. To accelerate the convergence process, the initial choice is conducted through Werner and Wengle's wall law [12], which can be reformulated as follows:

$$u_\tau\left(u_\parallel^S\right) = \begin{cases} \sqrt{\dfrac{u_\parallel^S \nu}{y^S}} & u_\parallel \le \dfrac{\nu}{4y^S} A^{\frac{2}{1-B}} \\ \left[\dfrac{1+B}{A}\left(\dfrac{\nu}{2y^S}\right)^B u_\parallel^S + \dfrac{1-B}{2} A^{\frac{1+B}{1-B}}\left(\dfrac{\nu}{2y^S}\right)^{1+B}\right]^{\frac{1}{1+B}} & \text{otherwise} \end{cases}, \tag{9}$$

where $A = 8.3$ and $B = \frac{1}{7}$. The obtained shear velocity u_τ is then helpful to impose the dimensionless wall tangent velocity u^+ of modeled cells and compute the wall tangent velocity components $u_\parallel^i$. Finally, the wall-normal velocity $u_\perp$ is defined by considering a linear development from the wall to the sampling point S:

$$u_\perp = u_\perp^S \cdot \frac{y}{y^S}. \tag{10}$$

This hypothesis 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 $u_\perp$ is negligible since the main velocity component is tangential to the wall surface.

3.2 Temperature and density

Temperature profiles from the wall are defined by the Crocco-Busemann relation [6] adapted for adiabatic wall conditions, as shown in equation 11.

$$T = T_w - \frac{T_w - T_e}{U_e^2} U^2, \tag{11}$$

where T_w is the wall temperature, T_e and U_e are the flow temperature and velocity magnitude outside the boundary layer. 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. The velocity components are determined as shown in section 3.1. However, T_e, U_e, and the wall temperature T_w (needed to evaluate the relation) are unknown. This forces to use an iterative approach along with solver iterations. Wall temperature T_w and $\frac{T_w - T_e}{U_e^2}$ ratio are determined, at each solver time step, through the resolution of the linear equations system established by the Crocco-Busemann relation written for the first two cells in the RANS area, above the enforcement of the wall model. Once T_w and $\frac{T_w - T_e}{U_e^2}$ are determined, it is possible to extend the Crocco-Busemann relation to all modeled cell layers in the near-wall zone.

Near the wall in the boundary layer, the pressure gradient in the wall-normal direction can be neglected. The density profile can thus be modeled using the perfect gas law applied to Crocco-Busemann's temperature evolution from the wall combined with pressure data at the first node in the RANS domain.

Additionally, since the temperature T and density ρ evolution are known, the molecular viscosity μ and kinematic viscosity ν can be determined through the viscosity model chosen by the solver (in our case, Sutherland's law).

3.3 Spalart-Allmaras variable

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

$$\tilde{\nu}^+ = \frac{\tilde{\nu}}{\nu} \tag{12}$$

in viscous sublayer and logarithmic layer can be modeled as follows

$$\tilde{\nu}^+ = \kappa y^+, \tag{13}$$

where $\kappa = 0.41$ is the Von Kármán constant. The presented wall models set the S-A variable $\tilde{\nu}$ to respect equation 13 in each cell of the modeled wall-bounded region.

4 Neural network training and implementation

This work uses two different feedforward neural networks (FNN) with different inputs as wall models. They are presented in the following sections 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 is composed of a variable number of nodes or neurons. The number of nodes in input and output layers is respectively imposed by the numbers of entries and outputs of the neural network (NN). Every layer of the NN is a dense layer, i.e., a fully connected layer. This means that every node of a single layer receives information from all nodes belonging to the previous layer, and reciprocally, its outputs are passed to all nodes of the next layer.

Considering a node i^L of a NN dense layer L, consisting in N_L neurons. The output O_i^L of the neuron i^L is given by:

$$O_i^L = f_i^L(W_i^L + b_i^L), \tag{14}$$

where f_i^L is a so-called activation function, and b_i^L is a scalar value called a bias. Moreover, W_i^L is the weighted inputs to the node i^L, obtained from a linear combination of the inputs I_i^{L-1} of the i^{th} node in

$L-1$ layer as expressed below

$$W_i^L = \sum_{n=1}^{N_{L-1}} w_i^L I_i^{L-1}. \tag{15}$$

The activation functions f_i^L add non-linearity to the network, allowing to fit complex nonlinear data patterns than an affine relationship. 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 both neural networks, expressed as:

$$f_i^L(X_i) = \begin{cases} x & x > 0 \\ \alpha(e^x - 1) & x \leq 0 \end{cases}, \tag{16}$$

with the hyperparameter α set to 1 in our case. The choice for ELU is driven by shown improved learning characteristics compared to other activation functions (e.g., ReLu) [2].

The proposed wall model is data-driven, which means that the neural network is built upon training driven by data extracted, in this case, from wall-resolved RANS simulations. The NN training, which sets w_i^L and b_i^L, is based on the minimization of a loss function that evaluates the error between RANS data and NN predictions. For the present work, the loss ϵ is defined as:

$$\epsilon = \frac{1}{N^S} \sum_{i=1}^{N^S} \left(y_i - y_i^S\right)^2 + \frac{\lambda_0}{N^S} \left\| w_i^L \right\|_1, \tag{17}$$

where N^S is the number of data samples and λ_0 is the regularization rate, which is set to 0.01. The first term in equation 17 denotes the mean squared error (MSE) between the NN output y_i and the output y_i^S from the training dataset. The second term is an L1 regularization term that promotes sparsity of optimized weights w_i^L. The optimization is then carried out using a gradient-based algorithm, in our case, the Adam algorithm [5].

To avoid overfitting, two datasets are considered: the training dataset and the 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 this 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 of neural networks training are given in appendix B.

4.1 Wall distance informed neural network (Network 1)

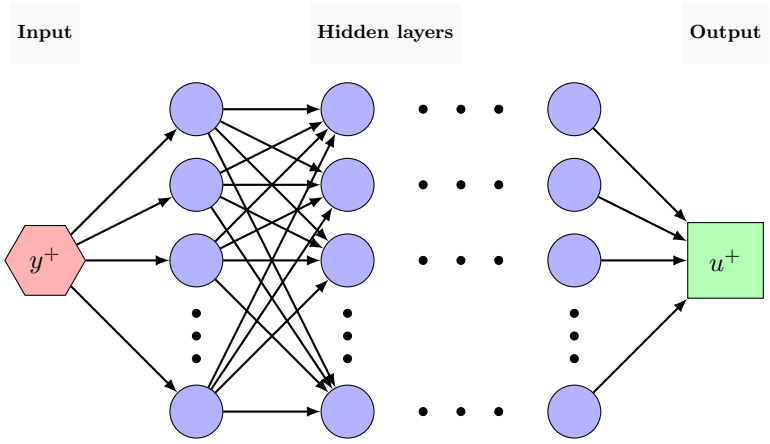


Figure 4: Network 1 (wall distance informed neural network): schematic diagram of the feedforward neural network with multiple hidden layers.

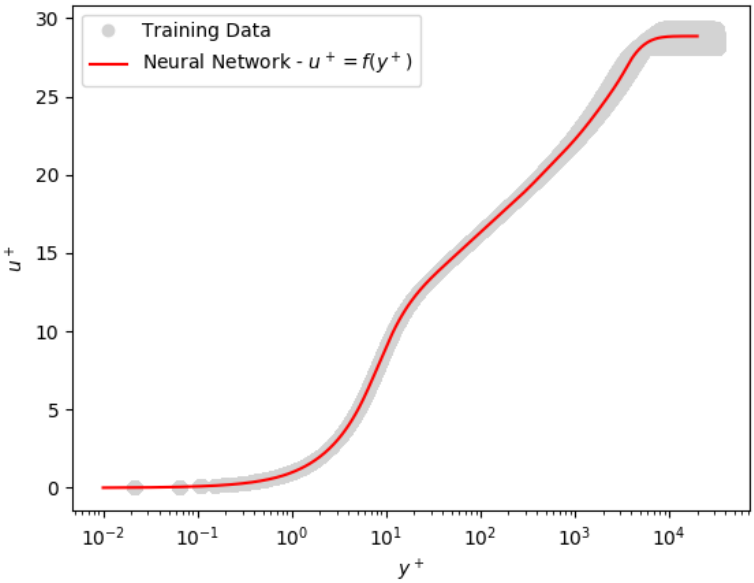


Figure 5: Network 1 (wall distance informed neural network): learned approximation of the dimensionless wall velocity $\mathbf{u}^+ = \mathbf{f}(\mathbf{y}^+)$.

The first neural network presented in this work is trained to estimate the dimensionless wall velocity u^+ from y^+ only, as in the relation below

$$u^+ = f\left(y^+\right). \tag{18}$$

The following refers to this network as "Network 1". Four hidden layers are employed in the wall distance informed neural network, composed of 16, 16, 8, and 4 units. A schematic diagram of the presented neural network is given in figure 4. The selected NN structure offers an efficient compromise between the complexity of the learned function and computational cost. Note that for a wall law, whose primary goal is the simulation cost reduction, having a lightweight model is maybe as important as the model's accuracy.

Network 1 is meant to reconstruct the profile of dimensionless tangential velocity u^+ independently of the presence of pressure gradient. Consequently, the dataset used for training is that of the flat plate case. Figure 5 shows the resulting velocity profile obtained from the NN at the end of the training, superimposed on the training data.

4.2 Wall distance and pressure gradient informed neural network (Network 2)

The second neural network is fed with the dimensionless wall distance and the dimensionless pressure gradient, such that it estimates equation 4. In the following, it will be called "Network 2". The network architecture is similar to the previous one: four main hidden layers are still employed, composed of 16, 16, 8, and 4 neurons. However, since the dynamical range of each input strongly differs (e.g. $y^+ \in [0, 10^3]$ and $p^+ \in [-0.02, 2]$), an additional hidden layer is added after the input layer to allow the NN to rescale the inputs. A schematic diagram of the NN architecture is given in figure 6

Network 2 aims to reconstruct the profile of dimensionless tangential velocity u^+ even in the presence of non-equilibrium effects induced by pressure gradients. The network is therefore trained using the database of the bump case. However, since the training dataset is composed of 8 cases that vary for Reynolds number Re and bump height h, remarkable differences are expected in the boundary layer, especially for high values of dimensionless wall distance y^+, which makes the training process problematic far from the wall. Training data are thus limited to the inner region of the boundary layer (i.e., $\frac{y}{\delta} \leq 0.15$). This selection in the training dataset equally limits the range of dimensionless wall distance, on which Network 2 can be applied, to a

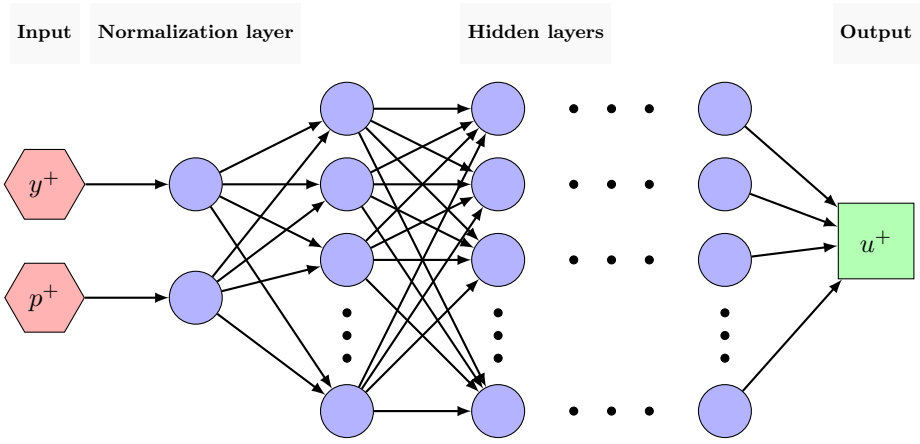
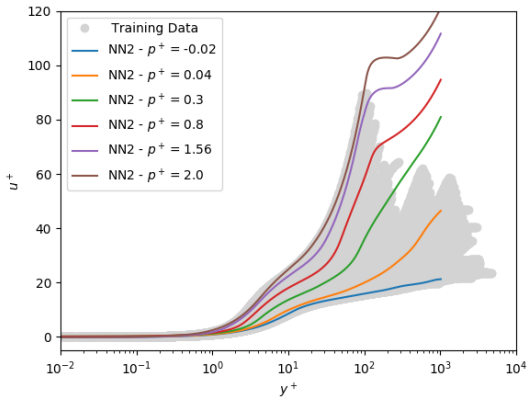
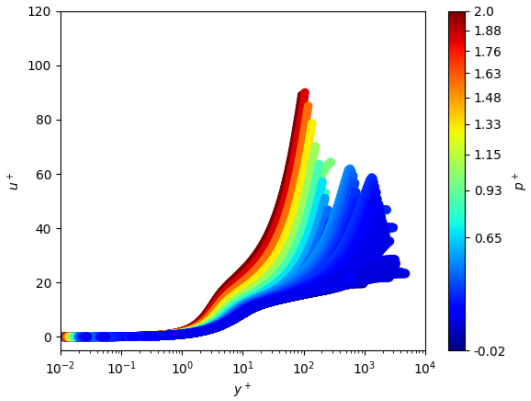


Figure 6: Network 2 (wall distance and pressure gradient informed neural network): schematic diagram of the feedforward neural network with multiple hidden layers.



(a) Learnt approximation of dimensionless wall velocity $\mathbf{u^+ = f(y^+, p^+)}$. Comparison with training dataset.



(b) Detail of selected training dataset. Limitation to inner region of boundary layer ($\frac{y}{\delta} \leq 0.15$).

Figure 7: Network 2 (wall distance and pressure gradient informed neural network): selected training data and learnt wall normal evolution of the dimensionless velocity u^+.

maximum y^+ of approximately 100. This limit is set by the smallest range of dimensionless wall distance found in the training dataset for the lowest 15% of the boundary layer thickness. Higher values of y^+ encountered during Network 2 exploitation could thus lead to non-physic results. Consequently, the grid refinement at wall should be chosen to respect the limitations of the proposed wall model.

The training process of Network 2 follows the methodology from Section 4. The resulting learned relation $u^+ = f(y^+, p^+)$ is shown in figure 7, which also displays the training RANS dataset. The validation of the learned relation $u^+ = f(y^+, p^+)$ is given in appendix C.

4.3 Implementation in the CFD solver

Neural networks are fully implemented and exploited in the FORTRAN source code of the CFD solver FASTs (see section 2.1), while the training process is performed through the deep-learning library TensorFlow [10].

As shown by equations 14 and 15, dense feedforward neural network can be considered as a chain of matrix operations followed by a nonlinear vector function performed between the inputs and outputs of each NN layer. The generic structure of this operation chain and all the standard activation functions have been

implemented in the FORTRAN code. The converged neural network weights, biases, and names of the used activation functions are provided to the CFD solver in the simulation case setup. The code thus adapts to the provided NN parameters to correctly reproduce the trained NN inside the CFD solver.

Additionally, since the derivative of the neural network is needed for the Newton algorithm (see section 3), the generated FORTRAN code that mimics the NN is algorithmically differentiated, provided a new routine that is added to the FORTRAN source code of the solver.

For this work, NN derivative codes have been generated by Tapenade [11], a source to source automatic differentiation engine developed by Inria Tropics and Ecuador teams, able to provide the algebraic derivative of a given source code automatically. Similarly to the direct exploitation of the NN, derivative algorithms are fully defined by the provided data of the trained neural network.

5 Results and discussions

This section presents the results obtained through the usage of Networks 1 and 2. Both of them are tested on the flat plate case and the bump case. The results are compared with the wall-resolved RANS simulation and the solution achieved by the utilization of the classically used Musker's wall law:

$$u^+ = 5.424 \tan^{-1}\left[\frac{2y^+ - 8.15}{16.7}\right] + \log_{10}\left[\frac{\left(y^+ + 10.6\right)^{9.6}}{\left(y^{+\,2} - 8.15y^+ + 86\right)^2}\right] - 3.52. \tag{19}$$

In equation 19, the terms taking into account the wall curvature are expressly neglected since their contribution becomes important only in the outer region of the boundary layer ($y/\delta \gg 0.15$), and this work focuses on the inner region for $y/\delta \leq 0.15$.

All the wall models are applied to a structured trimmed grid, as mentioned in section 3, which is based on the training dataset grid. Test grids are obtained by eliminating wall cells until the desired wall distance y^+ is achieved in the first RANS computed cells. For the test cases, the wall models affect the first three cell layers above the wall surface, while the sampling point is placed on the 5^{th} cell layer. Consequently, the standard RANS integration takes over from the 4^{th} cell center above the wall, and the sampling point is well defined in the RANS computed zone. The stencil of the discretization scheme used in the solver includes two cell layers below the RANS interface (e.g., 2^{nd} and 3^{rd} cell layers). Therefore, the first single coarse cell located between the wall and the first spatial discretization stencil is not used for the computation in practice. Figure 3.b illustrates the grid structure used for the test cases.

For each configuration, three values are considered for the dimensionless wall distance y^+ at the first RANS computed cell ($y^+ \approx 10$, 30, 50). The first computed RANS integrated cell is thus placed respectively in the buffer layer, between the buffer layer and the logarithmic region, and in the logarithmic region.

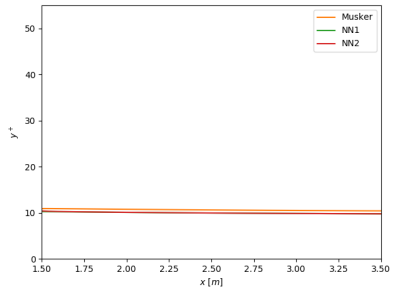
5.1 Flat plate test case

The presented wall models are tested on the flat plate case used for Network 1 training. These tests allow us to evaluate the training process of Network 1 and the interpolation capabilities of Network 2 to cope with an unseen case with no adverse pressure gradient. To limit the application of wall laws for well-established boundary layers, the simulation domain in figure 1 is limited to $x \in [1.5\text{m}, 3.5\text{m}]$.

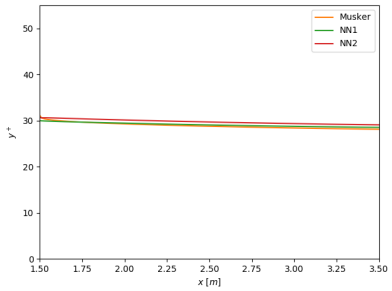
As said earlier, the wall models are tested on three meshes with different first RANS cell-center distances from the wall, which correspond to y^+ ranging from 9 to 11, from 28 to 31, and from 47 to 52 (see figure 8). For simplicity, these cases are labelled in the following as $y^+ \approx 10$, 30, 50, respectively.

Figure 9 shows the tangential velocity profile over the flat plate obtained at $x = 3\text{m}$ (85% of considered plate length). The NN wall models reproduce the tangential velocity profile accurately for all three grids. They significantly outperform Musker's wall law when the modeled zone is placed in the buffer region of the boundary layer ($y^+ \approx 30$ simulation).

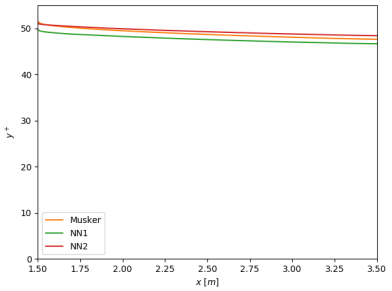
The temperature evolution from the wall at $x = 3\text{m}$ is shown in figure 10. For brevity, density profiles are omitted here since their behavior closely reproduces the performance of the temperature profile. The results strongly depend on the modeled grid point distance from the wall. For the first mesh simulation, the reference results are correctly reproduced by Musker's law and Network 2, with the former matching slightly better the reference. Network 1 shows the worse performance. The same trend also characterizes the other



(a) Wall distance at first RANS computed cell (mesh labeled as $\mathbf{y^+ \approx 10}$).

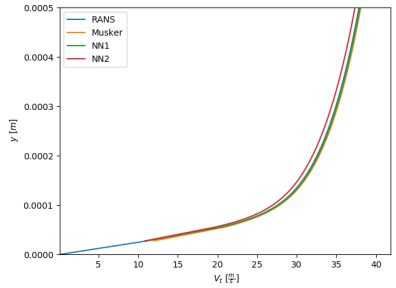


(b) Wall distance at first RANS computed cell (mesh labeled as $\mathbf{y^+ \approx 30}$).

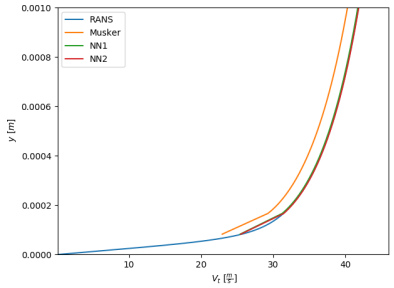


(c) Wall distance at first RANS computed cell (mesh labeled as $\mathbf{y^+ \approx 50}$).

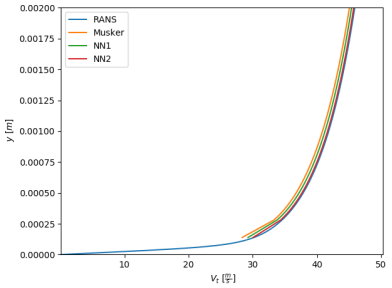
Figure 8: Flat plate case: y^+ variation of the first RANS computed grid point for the 3 different meshed considered.



(a) Wall distance at first RANS computed cell $\mathbf{y^+ \approx 10}$.



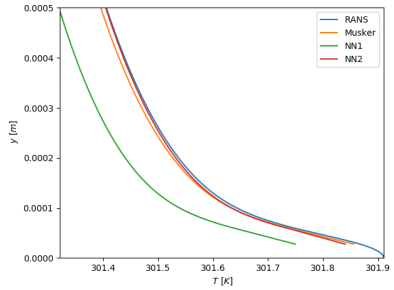
(b) Wall distance at first RANS computed cell $\mathbf{y^+ \approx 30}$.



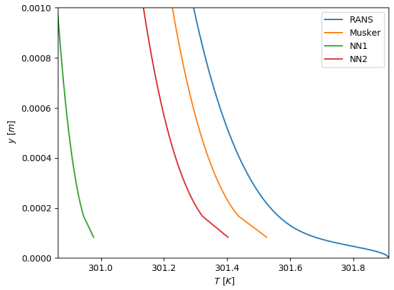
(c) Wall distance at first RANS computed cell $\mathbf{y^+ \approx 50}$.

Figure 9: Flat plate case: Wall normal evolution of tangential velocity at $\mathbf{x = 3m}$.

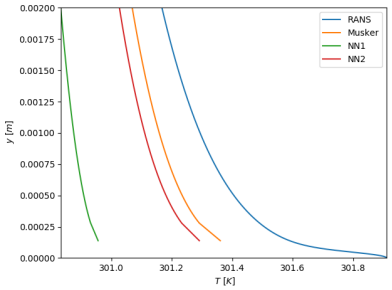
grid refinements, with an increasing error for all the compared wall models as the modeled region becomes larger. Nonetheless, for all the mesh simulation, the relative error on temperature prediction is very limited (i.e. below 1%) with respect to the wall-resolved simulation.



(a) Wall distance at first RANS computed cell $\mathbf{y^+ \approx 10}$.



(b) Wall distance at first RANS computed cell $\mathbf{y^+ \approx 30}$.



(c) Wall distance at first RANS computed cell $\mathbf{y^+ \approx 50}$.

Figure 10: Flat plate case: Wall normal evolution of temperature at $\mathbf{x = 0.75m}$ (top of the bump).

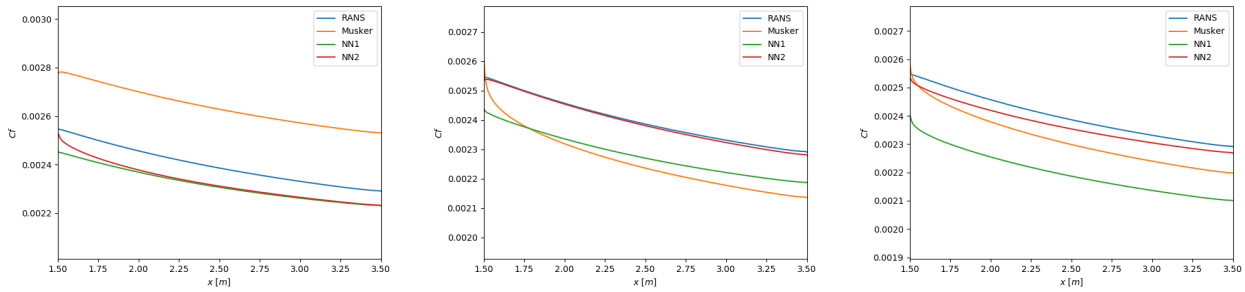
Figures 11 and 12 show for all cases the skin friction coefficient C_f along the wall, and its error with respect to the wall-resolved simulation. The error e is normalised by the mean C_f value of the reference simulation over the wall (a point-by-point normalisation using local C_f values is not used since nearly-zero

friction is found), such that

$$e(X_i) = \frac{C_f(X_i) - C_{f,ref}(X_i)}{\overline{C_{f,ref}}}, \tag{20}$$

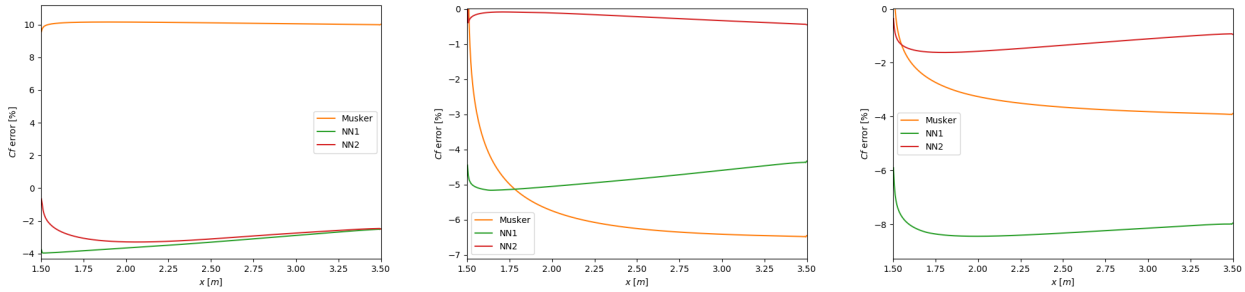
with X_i referring to the equally spaced solution points in $X-$coordinate direction along the considered portion of the wall, C_f the estimation of friction coefficient from the wall model, $C_{f,ref}$ the estimation of friction coefficient from the wall-resolved simulation and $\overline{C_{f,ref}}$ its mean value over the wall. Note that this error is purposely defined as a signed value (to evaluate potential model under/over-estimation).

The best results are achieved by Network 2 with an error that never exceeds 4 %, outperforming Musker's model. A close match with the wall-resolved solution is displayed for $y^+ \approx 30$ simulation. Network 1, on the contrary, shows good results overall, performing better on the finest and mid-refined grid than the classical Musker's law, despite a drop in performance when the modeled region is the highest. Despite being satisfactory, its performance is somewhat surprising since it is systematically less good than Network 2 despite being tested on a case seen during training. This reveals that even on a flat plate test case, accounting for p^+ as an input variable may lead to improved performances.



(a) Wall distance at first RANS computed cell $y^+ \approx 10$. **(b) Wall distance at first RANS computed cell $y^+ \approx 30$.** **(c) Wall distance at first RANS computed cell $y^+ \approx 50$.**

Figure 11: Flat plate case: Skin friction coefficient C_f along the $X-$coordinate direction.



(a) Wall distance at first RANS computed cell $y^+ \approx 10$. **(b) Wall distance at first RANS computed cell $y^+ \approx 30$.** **(c) Wall distance at first RANS computed cell $y^+ \approx 50$.**

Figure 12: Flat plate case: Error on skin friction coefficient C_f with respect to the wall-resolved RANS simulation.

Finally, table 1 shows the global 2-norm error e_2 computed for the flat plate case. The error is obtained as follows

$$e_2 = \frac{\left\| C_f(X_i) - C_{f,ref}(X_i) \right\|_2}{\left\| C_{f,ref}(X_i) \right\|_2} = \frac{\sum_i \left[C_f(X_i) - C_{f,ref}(X_i) \right]^2}{\sum_i C_{f,ref}^2(X_i)}. \tag{21}$$

The global error is in line with the previous results. Network 2 outperforms Network 1 and Musker's law in all the tested grid refinements. Network 1 displays increasing error when the grid is less refined. Nonetheless, it performs better than the classical Muskers's law for the finest and mid-refined grids.

2-Norm Error [%]	**Network 1**	**Network 2**	**Musker WM**
$\mathbf{y^+ \approx 10}$	3.30	2.92	10.08
$\mathbf{y^+ \approx 30}$	4.81	0.26	5.93
$\mathbf{y^+ \approx 50}$	8.21	1.33	3.46

Table 1: Flat plate case: 2-norm global error with respect to the wall-resolved RANS simulation. Comparison between wall distance informed neural network (Network 1), wall distance and pressure gradient informed neural network (Network 2) and Musker's wall law tested on three different wall refinements.

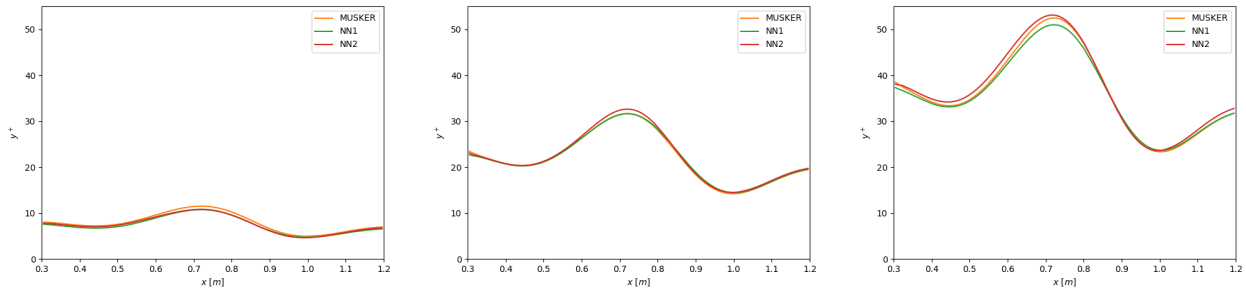
5.2 Bump test case

The second test presented in this work is the bump introduced in section 2. The test case corresponds to $h = 0.05$m and $Re = 3 \cdot 10^6$. This case represents an unseen combination of parameters during the training of Network 2. It allows for the evaluation of the interpolation capabilities of the NN. Network 1, which was not designed to handle adverse pressure gradients, is also tested and compared with Musker's law in such a situation.

Like the flat plate case, the wall models are only applied to established turbulent boundary layers. For this reason, the complete simulation domain for the wall-modeled computation is now limited to $x \in [0.3\text{m}, 6.5\text{m}]$. However, wall model performances are evaluated on the bump geometry only, from $x = 0.3$m to $x = 0.9$m, which corresponds to the extraction zone for the training data. Inflow conditions are such that a fully developed boundary layer is injected at $x = 0.3$m.

The dimensionless wall distance y^+ at the first RANS computed cell-center for each of the three grid refinements considered is shown in figure 13. The first integrated cell is placed at y^+ ranging from 4 to 12, from 33 to 14, and from 54 to 23, respectively. These three meshes are referred to in the following as $y^+ \approx 10$, 30, 50, respectively.

Figure 14 shows tangential velocity profiles obtained at $x = 0.75$m, corresponding to the top of the bump, where significant differences between the compared models are found. The behavior of the modeled velocity profiles qualitatively corresponds to the observations from the flat plate case. The proposed data-driven models perform closely and estimate the actual wall-resolved solution accurately. The difference with Musker's law is the strongest for the mesh $y^+ \approx 30$, where their tangential velocity predictions significantly outperform Musker's model. For the other meshes, the results are closer to each other.

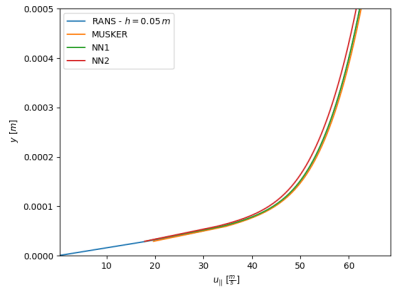


(a) Wall distance at first RANS computed cell (mesh labeled as $\mathbf{y^+ \approx 10}$).

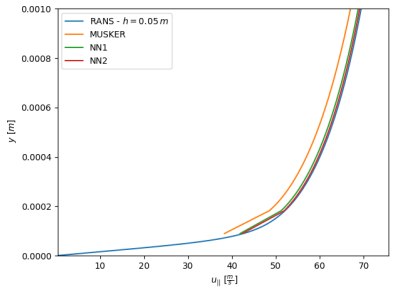
(b) Wall distance at first RANS computed cell (mesh labeled as $\mathbf{y^+ \approx 30}$).

(c) Wall distance at first RANS computed cell (mesh labeled as $\mathbf{y^+ \approx 50}$).

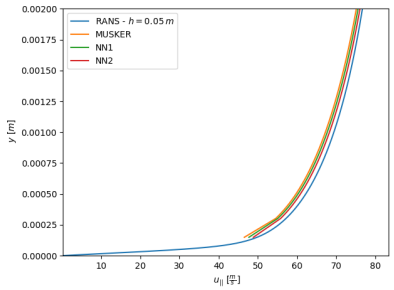
Figure 13: Bump test case: y^+ variation of the first RANS computed grid point for the 3 different meshed considered.



(a) Wall distance at first RANS computed cell $y^+ \approx 10$.



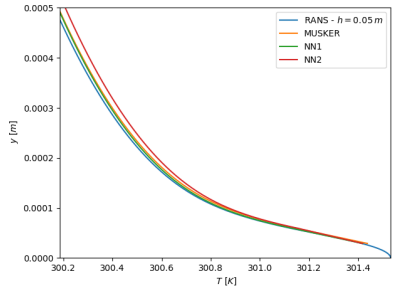
(b) Wall distance at first RANS computed cell $y^+ \approx 30$.



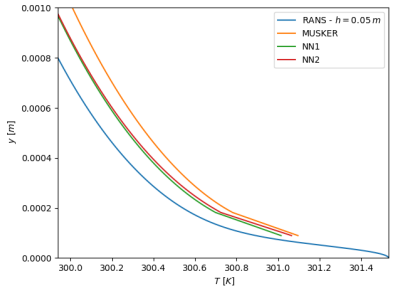
(c) Wall distance at first RANS computed cell $y^+ \approx 50$.

Figure 14: Bump test case: Wall normal evolution of tangential velocity at x = 0.75m (top of the bump).

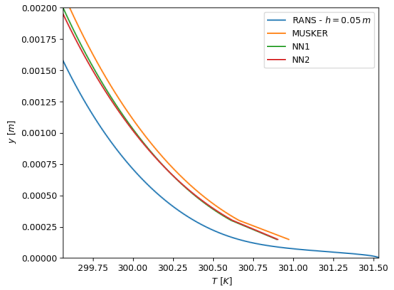
Figure 15 shows the temperature evolution from the wall at $x = 0.75$m. All wall models seem to perform similarly, especially for the first mesh refinement. Both NN models outperform Musker's law for the two coarser grids. The observed error increases as the grid get coarser, but it stays well below 1%.



(a) Wall distance at first RANS computed cell $y^+ \approx 10$.



(b) Wall distance at first RANS computed cell $y^+ \approx 30$.



(c) Wall distance at first RANS computed cell $y^+ \approx 50$.

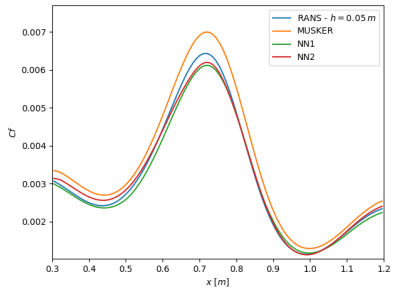
Figure 15: Bump test case: Wall normal evolution of temperature at x = 0.75m (top of the bump).

Figures 16 and 17 show respectively the skin friction coefficient C_f over the bump in the $X-$coordinate direction and its error with respect to the wall-resolved simulation (defined in the previous section). The best performances are achieved, as expected, by Network 2, which was specifically designed and trained to deal with pressure gradients. It significantly outperforms the other wall models. On the contrary, Network 1, which was not trained on pressure gradient data, performs better than Musker's model with the most refined grid (i.e., $y^+ \approx 10$), but for coarser grids, the classical wall model gives better results. Overall, this shows, as expected, that none of these two models are suited to be used as wall models for RANS simulations with significant adverse pressure gradients.

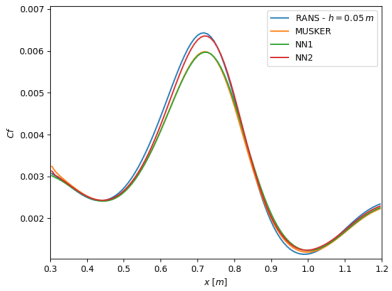
Table 2 shows the 2-norm error (defined in the previous section) computed for the bump case. As expected, the global error is in line with previous conclusions. Network 2 outperforms the other wall models, with an error that stays nearly constant as the grid refinement changes.

6 Conclusion and perspectives

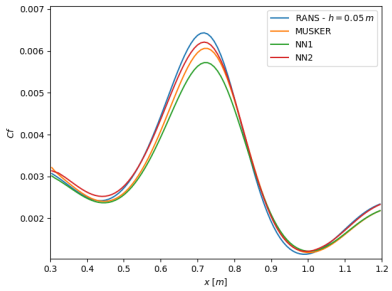
This work presents a new deep learning-based approach to wall models for RANS simulations inspired by classical wall laws. The proposed wall models rely on wall dimensionless quantities, such as the wall distance y^+, the wall friction velocity u_τ and the wall pressure gradient p^+ to reconstruct the dimensionless wall velocity u^+ profiles in wall-bounded region. The model provides embedded neural networks to the



(a) Wall distance at first RANS computed cell $y^+ \approx 10$.

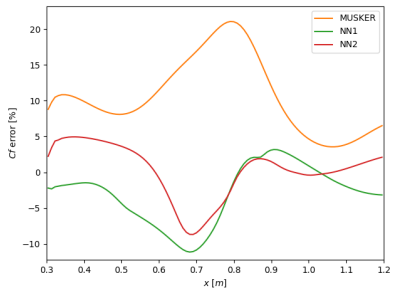


(b) Wall distance at first RANS computed cell $y^+ \approx 30$.

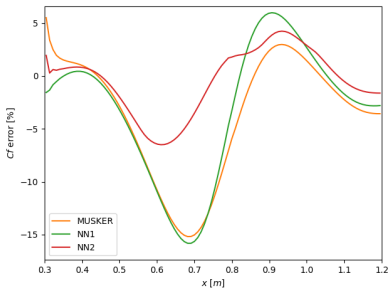


(c) Wall distance at first RANS computed cell $y^+ \approx 50$.

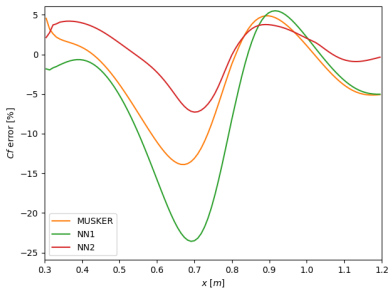
Figure 16: Bump test case: Skin friction coefficient C_f along the $X-$coordinate direction.



(a) Wall distance at first RANS computed cell $y^+ \approx 10$.



(b) Wall distance at first RANS computed cell $y^+ \approx 30$.



(c) Wall distance at first RANS computed cell $y^+ \approx 50$.

Figure 17: Bump test case: Error on skin friction coefficient C_f with respect to wall-resolved RANS simulation along $X-$coordinate direction.

2-Norm Error [%]	**Network 1**	**Network 2**	**Musker WM**
$\mathbf{y^+ \approx 10}$	4.47	3.29	10.59
$\mathbf{y^+ \approx 30}$	6.19	2.74	6.02
$\mathbf{y^+ \approx 50}$	9.17	2.97	5.7

Table 2: Bump case: 2-norm global error with respect to wall-resolved RANS simulation. Comparison between wall distance informed neural network (Network 1), wall distance and pressure gradient informed neural network (Network 2) and Musker's wall law on the three trimmed grid refinements.

CFD solver code which force the primitive variables in modeled cells near the wall, where Navier-Stokes computation is disabled. A Neumann boundary condition between the wall modeled and conventional RANS computed regions is thus established. This new approach may be interesting for future use for RANS simulations with immersed boundary method simulations (IBM). One identified shortcoming is that the proposed methodology could lead to conservativity problems. However, in this first work, conservativity issues are found to be negligible since the main velocity component of the flow in the modeled region is tangential to the wall surface. However, further considerations of the problem are required to extend the validity domain of the wall model, which may be an interesting research direction for the future.

Two deep learning-based approaches are presented. The first one consist of a wall distance informed neural network trained on a dataset extracted from a fine wall-resolved RANS simulation of the flow over a flat plate. The second network is wall distance and pressure gradient informed, and is trained on data extracted from a set of fully resolved RANS simulations of the flow over a bump. The training process is performed with different Reynolds number conditions and pressure gradient levels based on the bump height.

Both neural networks are tested and compared with fully resolved RANS simulation and with the classical Musker's wall model. Two test cases are selected. The first one coincides with the training simulation over the flat plate, and the second one reproduces the flow over a bump in a configuration of Reynolds number and bump height that is not included in the training dataset of any of the neural networks. The benchmark cases are run over different grid refinements to test the proposed wall model with various wall distances at the first RANS computed cell center.

The wall distance informed neural network showed acceptable results in all the test conditions with better skin friction predictions than the classical Musker's wall model if the modeled zone is below the logarithmic region of the boundary layer where pressure gradients effects become not negligible.

The wall distance and pressure gradient informed network displayed the best results, consistently outperforming the classic analytical law and the other neural network in the skin friction prediction for all the test cases and grids. Only minor inaccuracies were found in the wall tangential velocity and temperature (and density) evolution.

Even though the test cases are characterized by a fairly simple geometry with moderate adverse pressure gradients, this work highlights the potential of neural networks for wall-bounded region modeling. The present results are the starting point for further studies and investigations required to overcome the minor issues encountered during this work and to 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.

References

[1] Ivan Mary et al. *FAST: A compressible flow solver python package*. URL: https://w3.onera.fr/FAST/.

[2] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. *Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)*. 2015. DOI: 10.48550/ARXIV.1511.07289. URL: https://arxiv.org/abs/1511.07289.

[3] Georgi Kalitzin et al. "Near-wall behavior of RANS turbulence models and implications for wall functions". In: *Journal of Computational Physics* 204.1 (2005), pp. 265–291. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2004.10.018. URL: https://www.sciencedirect.com/science/article/pii/S0021999104004164.

[4] Soshi Kawai and Kengo Asada. "Wall-modeled large-eddy simulation of high Reynolds number flow around an airfoil near stall condition". In: *Computers and Fluids* 85 (2013), pp. 105–113. ISSN: 00457930. DOI: 10.1016/j.compfluid.2012.11.005. URL: http://dx.doi.org/10.1016/j.compfluid.2012.11.005.

[5] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. DOI: 10.48550/ARXIV.1412.6980. URL: https://arxiv.org/abs/1412.6980.

[6] B. W. van Oudheusden. "Some Classic Thermal Boundary Layer Concepts Reconsidered (and their Relation to Compressible Couette Flow)". In: *IUTAM Symposium on One Hundred Years of Boundary Layer Research*. Ed. by G. E. A. Meier, K. R. Sreenivasan, and H.-J. Heinemann. Dordrecht: Springer Netherlands, 2006, pp. 425–434. ISBN: 978-1-4020-4150-1.

[7] Ugo Piomelli. "Wall-layer models for large-eddy simulations". In: *Progress in Aerospace Sciences* 44.6 (2008), pp. 437–446. ISSN: 03760421. DOI: 10.1016/j.paerosci.2008.06.001.

[8] P.L Roe. "Approximate Riemann solvers, parameter vectors, and difference schemes". In: *Journal of Computational Physics* 43.2 (1981), pp. 357–372. ISSN: 0021-9991. DOI: https://doi.org/10.1016/0021-9991(81)90128-5. URL: https://www.sciencedirect.com/science/article/pii/0021999181901285.

[9] Christopher Rumsey. *2D Bump-in-channel Verification Case*. URL: https://turbmodels.larc.nasa.gov/bump.html.

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

[11] INRIA Tropics and Ecuador teams. *TAPENADE: An Automatic Differentiation Engine*. URL: https://team.inria.fr/ecuador/en/tapenade/.

[12] H. Werner and H. Wengle. "Large-Eddy Simulation of Turbulent Flow Over and Around a Cube in a Plate Channel". In: *Turbulent Shear Flows 8*. Ed. by Franz Durst et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, pp. 155–168. ISBN: 978-3-642-77674-8.

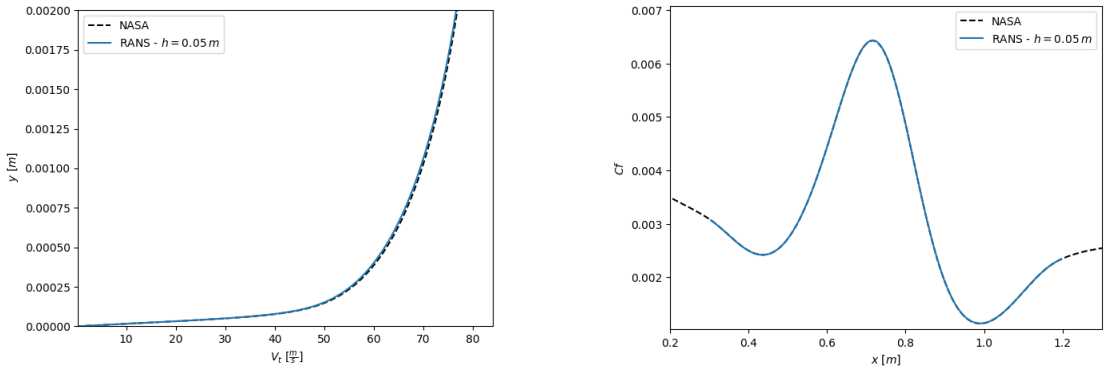
[13] Zhideng Zhou, Guowei He, and Xiaolei Yang. "Wall model based on neural networks for les of turbulent flows over periodic hills". In: *Physical Review Fluids* 6.5 (2021), pp. 1–30. ISSN: 2469990X. DOI: 10.1103/PhysRevFluids.6.054610. arXiv: 2011.04157.

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 [9]. 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 0.05m. A Reynolds number $Re = 3 \cdot 10^6$ and a Mach number $Ma = 0.2$ 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 18, the tangential velocity at the top of the bump ($x = 0.75$m) 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)**

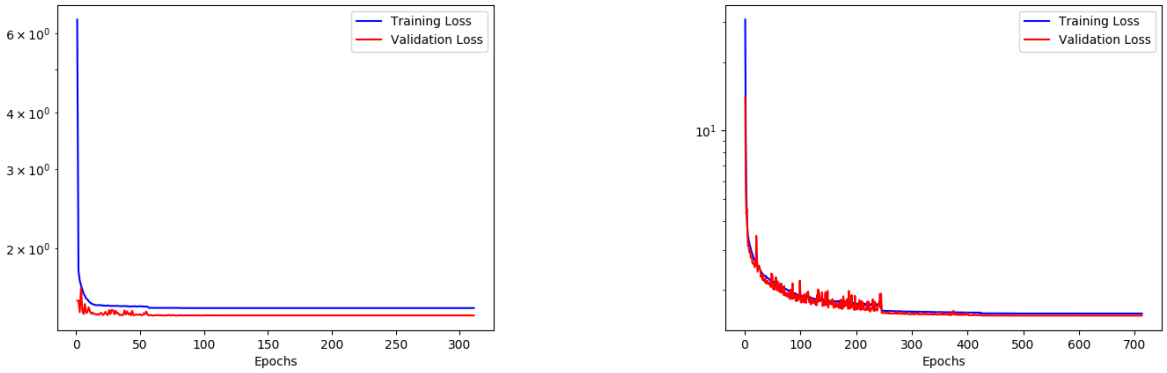
Figure 18: Validation of bump case setup. Comparison of NASA results and wall-resolved RANS simulation of the flow over a bump with an height $\mathbf{h = 0.05\,m}$. Tangential velocity to surface at $\mathbf{x = 0.75}$m (top of the bump) (a) and skin friction coefficient $\mathbf{C_f}$ over the bump (b).

B Training process of neural networks

This section details the training process performed on Network 1 and 2 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 17, based on the mean squared error (MSE) with an L1 regularization term to promote sparsity of 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 64 samples. Using a standard validation stopping criterion as mentioned above, the training of both Network 1 and 2 required respectively 312 and 717 epochs. The stopping criteria, based on the validation loss, stops the training if the loss value does not decrease of 10^{-4} during the last 250 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 19 for both NN.

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 19. Loss oscillations visibly reduce as the learning rate is decreased. Like for the stopping criteria, the learning rate reduction is driven by the



(a) Network 1 **(b) Network 2**

Figure 19: Training and validation loss evolution during training process.

validation loss. The learning rate is reduced by steps of 10% until a minimum value of 10^{-8} is achieved. Each reduction step is performed if the mean loss value over the last 20 epochs does not vary.

C Validation of wall distance and pressure gradient informed neural network

The validation of the training process performed on the wall distance and pressure gradient informed neural network (Network 2) is obtained by comparing the learned relation $u^+ = f\left(y^+, p^+\right)$ with the training data.

In figure 20, the training dataset is split into different subsets based on the dimensionless pressure gradient; then for each of the subsets, the learned u^+ evolution is traced for extreme values of p^+ found in the subset. As expected, the learned evolution of the dimensionless velocity u^+ sets the upper and lower limits of the training subsets in the validity range of the proposed wall model (i.e. $y^+ \leq 100$). This validates the training process of Network 2.

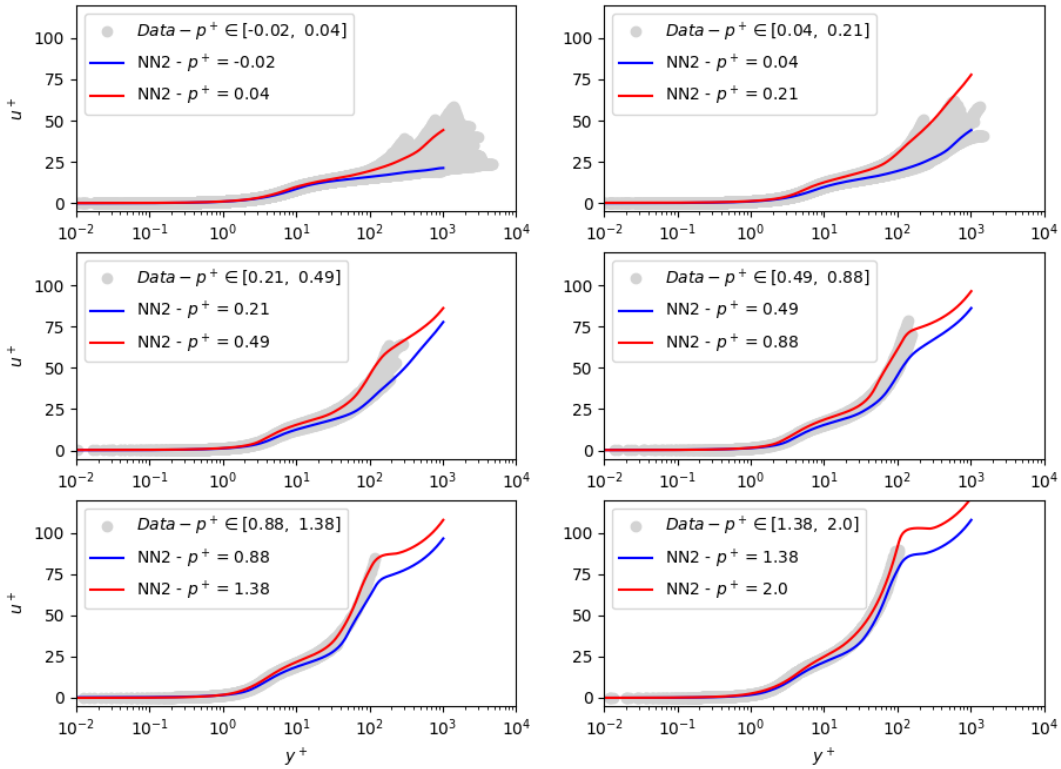


Figure 20: Learnt $\mathbf{u}^+ = \mathbf{f}(\mathbf{y}^+, \mathbf{p}^+)$ relation by wall distance and pressure gradient informed neural network (Network 2).