ERCOFTAC symposium on Engineering, Turbulence, Modelling and Measurements (ETMM14)
in Mini-Symposium: Machine learning for turbulence, Barcelona, Spain 6th - 8th September 2023

USING MACHINE LEARNING FOR FORMULATING NEW WALL FUNCTIONS FOR DETACHED EDDY SIMULATION

L. Davidson

Div. of Fluid Dynamics, Dept. of Mechanics and Maritime Sciences
Chalmers University of Technology, SE-412 96 Gothenburg, Sweden
lada@chalmers.se

Abstract

Machine Learning (ML) is used for developing wall functions for Detached Eddy Simulations. I use Improved Delayed Detached Eddy Simulations (IDDES, Shur et al. (2008)) in fully-developed channel flow at a frictional Reynolds number of 5 200 to create the database. This database is used as a training set for the ML method (support vector regression, SVR, in Python is used). The input (i.e. the influence parameters) is y^+. The ML method is trained to predict U^+.

The trained ML model is saved to disk and it is subsequently uploaded into the Python CFD code **pyCALC-LES** (Davidson, 2021). IDDES is carried out on coarse wall-function meshes. The wall-shear stress (using the local y^+ and $\bar{u}$) is predicted using the developed ML model. The test cases are channel flow at $Re_\tau = 16\,000$ and flat-plate boundary layer at $Re_\theta = 2\,550$.

1 Introduction

Machine Learning (ML) is a method where known data are used for teaching the algorithm to classify a set of data. The data may be photographs where the ML algorithm should recognize, for example, traffic lights or traffic signs (Rao and Desai, 2021). Another example may be ECG signals where the ML algorithm should recognize certain unhealthy conditions of the heart (Lindholm et al., 2022). A third example is detecting fraud for credit card payments (Rachana et al., 2021). ML methods such as Support Vector Machines (SVM) and neural networks are often used for solving this type of problems.

The examples above are classification problems using supervised learning (i.e. learning to recognise a traffic light, an unhealthy heart, learn what a customers usual credit card payment looks like). However, in the present work input and output are numerical values. In this case, ML in the form of regression methods should be used (Lindholm et al., 2022); I will use support vector regression (SVR) methods available in Python.

In SVR a regression multi-dimensional "surface" is created which has as many dimensions as number of influence parameters (in the present work I use one influence parameter). Let's make a simple example. In Fig. 1 there is one influence parameter, X, and one parameter to predict, y. Two control parameters are given to the SVR methods. The first is ϵ which determines the width of the tube around the hyperplane [1]. Points that lie inside this tube are considered as correct predictions and are not penalized by the algorithm. The support vectors are the points that lie outside the tube. The second parameter given to SVR models is the C value. It controls the "slack" (ξ), see Fig. 1, which is the distance to points outside the tube. If C is increased the size of the tube is increased so that some or all of the data points are located inside the tube.

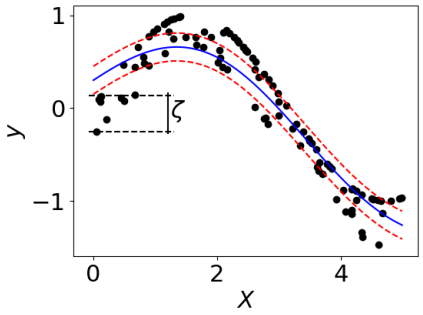


Figure 1: ——: hyperplane; – – –: $\pm\varepsilon$; •: SVR predictions with $C = 0.15$ and $\varepsilon = 0.1$. ζ: slack from hyperplane$-\varepsilon$ at $X \simeq 0$.

There are not many studies in the literature on ML for improving wall functions. In Tieghi et al. (2020) they use a time-averaged high-fidelity IDDES simulation to train a neural network for improving the predicted modeled turbulent kinetic used in wall functions (RANS). In Ling et al. (2017) they use neural network to improve the predicted wall pressure to be used in fluid-structure interactions. Their target is the wall pressure spectrum and the input parameters are the pressure power spectra above the wall. In Dominique et al. (2022) they use neural network to predict the wall pressure spectra. Their input data are boundary-layer thicknesses (physical, displacement and momentum),

[1] A hyperplane is a plane whose number of dimension is the same the number of influence parameters. For example, a two-dimensional hyperplane has two influence parameters.

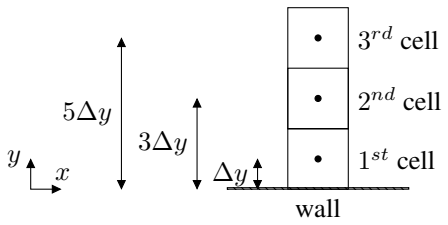


Figure 2: IDDES database.

streamwise pressure gradient and wall shear stress which are taken from experiments and high-fidelity DNS/LES in the literature. In Bae and Koumoutsakos (2022) they use an overly complicated neural network to create a pre-multiplication factor of the velocity-based wall model (VWM) and a log-law based wall model (LLWM). Then they introduce a reward factor, r_n, at each time step n.

The paper is organized as follow: first, I present the numerical model. It is followed by a description on how to generate the target database. Then I present to standard wall functions which are used for comparison. Next, the new ML-based wall function is presented. It is followed by a presentation of the results and in the final section I make some concluding remarks.

2 Numerical method

The finite volume code **pyCALC-LES** (Davidson, 2021) is used. It is written in Python and is fully vectorized (i.e. no `for` loops). Second-order central differencing is used in space for the momentum equations and Crank-Nicolson is used in time. For k and ε, the hybrid central/upwind scheme is used together with first-order fully-implicit time discretization. All discretized equation (i.e. the sparse-matrix system) are solved on the GPU using the Algebraic MultiGrid library `pyAMGx` (Olson and Schroder, 2018) based on AMGX.

3 Creating the database

To create a database which can be used for training the SVR, I carry out simulations using IDDES of fully-developed channel flow. The equations read

$$\begin{aligned} \frac{\partial \bar{u}_i}{\partial x_i} &= 0 \\ \frac{\partial \bar{u}_i}{\partial t} + \frac{\partial}{\partial x_j}\left(\bar{u}_i \bar{u}_j\right) &= \delta_{1i} \\ -\frac{\partial \bar{p}}{\partial x_i} &+ \frac{\partial}{\partial x_j}\left[\left(\nu + \nu_{sgs}\right)\frac{\partial \bar{u}_i}{\partial x_j}\right] \end{aligned} \qquad (1)$$

The underlying RANS turbulence model for IDDES reads (Abe et al., 1994)

$$\begin{aligned} \frac{\partial k}{\partial t} + \frac{\partial \bar{u}_j k}{\partial x_j} &= \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\nu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right] + P_k - \psi\varepsilon \\ \frac{\partial \varepsilon}{\partial t} + \frac{\partial \bar{u}_j \varepsilon}{\partial x_j} &= \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\nu_t}{\sigma_\varepsilon}\right)\frac{\partial \varepsilon}{\partial x_j}\right] \\ +C_{\varepsilon 1} P_k \frac{\varepsilon}{k} &- C_{\varepsilon 2} f_2 \frac{\varepsilon^2}{k}, \quad \psi = \frac{k^{3/2}}{\varepsilon L_{ref}} \\ C_{\varepsilon 1} &= 1.5, \quad C_{\varepsilon 2} = 1.9, \quad C_\mu = 0.09 \\ \nu_t &= C_\mu f_\mu \frac{k^2}{\varepsilon}, \quad \sigma_k = 1.4, \quad \sigma_\varepsilon = 1.4 \end{aligned}$$

where the damping functions are defined as

$$\begin{aligned} f_2 &= \left[1 - \exp\left(-\frac{y^*}{3.1}\right)\right]^2 \left\{1 - 0.3\exp\left[-\left(\frac{R_t}{6.5}\right)^2\right]\right\} \\ f_\mu &= \left[1 - \exp\left(-\frac{y^*}{14}\right)\right]^2 \left\{1 + \frac{5}{R_t^{3/4}}\exp\left[-\left(\frac{R_t}{200}\right)^2\right]\right\} \end{aligned}$$

The size of the channel is $x_{max} = 3.2$ (streamwise, x or x_1), $y_{max} = 2$ (wall normal, y or x_2) and $z_{max} = 1.6$ (spanwise, z or x_3). The mesh has $96 \times 96 \times 96$ cells and the Reynolds number is 5 200 based on the friction velocity, $\langle u_\tau \rangle$ ($\langle \cdot \rangle$ denotes average in time, x_1 and x_3), and the half-channel width, δ. DNS was used in Davidson (2022a) on a much finer mesh and a lower Reynolds number. However, the accuracy of IDDES is consider to be sufficient in the present study.

Figure 4 in Davidson (2022b) presents comparison of the predicted velocity field and RSM fluctuations with DNS and it is found that the agreement is reasonable.

The instantaneous velocity, $\bar{u}$, is stored at nine locations in the database along with the friction velocity at the same (x, z) position. It may be noted that in Davidson (2022a) $\bar{u}$ was integrated over $2\Delta y$, see Fig. 2 (giving $\bar{U}$) where $2\Delta y$ was relevant for the wall-normal cell size when using wall functions. At the end of Davidson (2022a), it was found that the correlation between $\bar{U}$ and the friction velocity is very low and it was concluded that integrating in y is not a good idea.

The grid in the wall-parallel plane is finer than in a typical wall-function mesh. Hence, the instantaneous $\bar{u}$ velocities and the friction velocities are integrated over ΔX and ΔZ where ΔX and ΔZ correspond to typical cell size in a wall-function mesh. ΔX and ΔZ are set to 0.1 and 0.05, respectively. The nine locations of the first cell are given in Table 1. The locations of the second and third cells are shown in Fig. 2. The $\bar{U}$ velocity at the second and the third cells are stored in order to be able to compute the first and the second velocity derivative which could be used as additional input (i.e. influence) parameters. However, they are not used in the present work.

4 Standard wall functions

The ML wall functions will be compared to wall functions based on Reichardt's law

$$\begin{aligned} \frac{\bar{u}_P}{u_\tau} \equiv U^+ = \frac{1}{\kappa}\ln(1 - 0.4y^+) + 7.8\left[1 - \exp\left(-y^+/11\right)\right. \\ \left. - \left(y^+/11\right)\exp\left(-y^+/3\right)\right] \end{aligned}$$

	$\langle \Delta y^+ \rangle$
Location 1	12
Location 2	31
Location 3	49
Location 4	66
Location 5	76
Location 6	88
Location 7	135
Location 8	155
Location 9	207

Table 1: IDDES database for wall functions. Δy is defined in Fig. 2. The locations of the second and the third cell are obtained from Fig. 2.

The friction velocity is then obtained by re-writing this equation and solving the implicit equation

$$u_\tau - \bar{u}_P\{\ln(1-0.4y^+)/\kappa + 7.8[1-\exp\left(-y^+/11\right) -(y^+/11)\exp\left(-y^+/3\right)]\}^{-1} = 0 \quad (2)$$

using the Newton-Raphson method `scipy.optimize.newton` in Python. $\bar{u}_P$ and y^+ denote the wall-parallel velocity and non-dimensional wall distance, respectively, at the first, second or third wall-adjacent cells. Unless otherwise stated, the first wall-adjacent cells are used.

The ML wall functions will also be compared to the standard log law, i.e.

$$\frac{\bar{u}_P}{u_\tau} = \frac{1}{\kappa}\ln\left(Ey^+\right) \quad \Rightarrow \quad u_\tau = \frac{\kappa \bar{u}_P}{\ln\left(Eu_\tau \delta y/\nu\right)}$$

where $E = 9.0$ and δy is the distance to the wall from the cell center of the wall-adjacent cells. For $x_2^+ \leq 11.63$, the friction velocity is computed from the linear law

$$u_\tau = \left(\nu \frac{\bar{u}_P}{\delta y}\right)^{1/2}$$

5 A new wall function based on Machine Learning (ML)

The ML method (Support Vector Regression, SVR) consists of a learning part and a testing part. In the learning part, the ML method is trained and in the testing part it is tested. The `svr` package in Python is used.

First, I need to determine which input variable (influence parameters) that should be used. In standard wall functions, the input parameters are the wall-parallel velocity, $\bar{u}_P$, and the non-dimensional wall distance, y^+ (which includes the friction velocity, u_τ); the output is the friction velocity. Hence, the friction velocity is used both as input and output. In order to avoid this, I chose – in my first attempt (Davidson, 2022a) – the local Reynolds number, $\bar{u}_p y/\nu$, the non-dimensional velocity gradient and the time-averaged non-dimensional wall distance, $\langle y^+ \rangle$ as influence parameters. This ML wall function gave fairly good results in fully-developed channel flow (Davidson, 2022a) but it was later found that it fails in flat-plate boundary layers.

Hence, I decided to pick a different set of influence and output parameters. I take guidance from the traditional wall laws (the linear law, the log law and the Reichard's law) which all three are written on the form $U^+ \equiv \frac{\bar{u}}{u_\tau} = f\left(y^+\right) \equiv f\left(\frac{u_\tau y}{\nu}\right)$. In the present work, I choose to use the same output and input in the ML wall functions, i.e. y^+ as influence parameter (input) and U^+ as output parameter. The potential drawback is that u_τ appears both in the input and output parameter which could necessitate an iterative procedure predicting U^+ a couple of times with updated y^+. In the results section it is found that that is not needed.

Next, I train the ML method using IDDES data at $Re_\tau = 5\,200$. I use 300 independent, instantaneous samples at each of the nine locations (Table 1). I pick 80% of the data randomly and define that as the training set. The remaining 20% is then used for testing, i.e. predicting. The error, e, between the predicted U^+ (testing samples) using `svr` and the IDDES database is 9% (Davidson, 2022b).

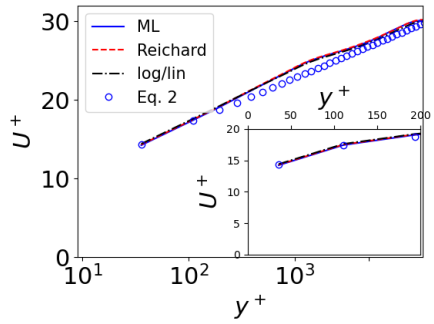
The predicted friction velocity is used – both in the ML model, and Reichardt's law and the log/linear wall functions – for setting $k = C_\mu^{-1/2} u_\tau^2$ and $\varepsilon = \frac{u_\tau^3}{\kappa \delta y}$ in the wall-adjacent modes. The friction velocity is also used to set $\tau_w = u_\tau^2$ (i.e. boundary condition for the $\bar{u}_1$ equation) at the walls.

6 Results using the ML wall function

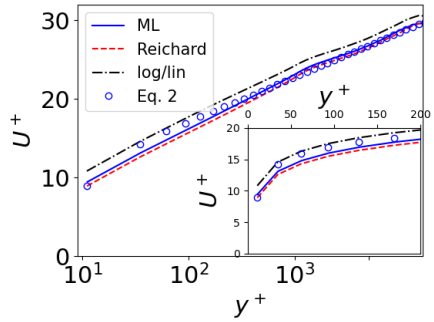
The Python code **pyCALC-LES** is used for all simulations together with the IDDES model. The Reynolds number in the channel flow is $Re_\tau = 16\,000$ based on the friction velocity and channel half width. The extent of the domain in the x and z direction is 3.2 and 1.6, respectively, covered by 32 cells in each direction.

First, I use a typical wall-function mesh ($N_y = 66$, stretching 11%) placing the wall-adjacent cells at $y^+ = 34$. The velocity profiles are shown in Fig. 3a and the agreement with Reichardt's law near the wall is excellent. However, when the stretching is increased to 14.7% the agreement near the wall deteriorates, see Fig. 3b. The reason for the poor agreement is probably that the cells further away from the wall (2nd, 3rd ... cell) are too coarse.

Since the grid seems to be too coarse away from the wall in Fig. 3b a new grid strategy is proposed. Figure 4 presents two distributions of grid lines in the wall-normal direction. A low-Reynolds grid used in RANS and IDDES in Fig. 4a with stretching of 15%. Figure. 4b present the new grid in which a number of near-wall cells in the low-Reynolds grid are merged

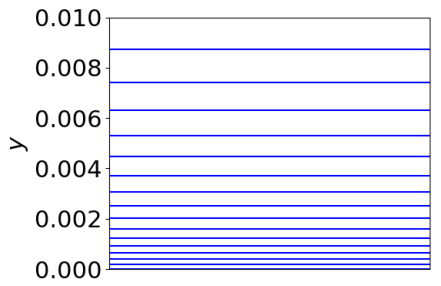


(a) $N_y = 66$, stretching 11%.

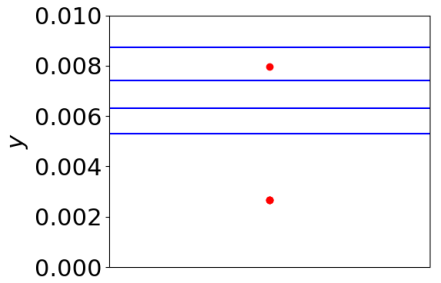


(b) $N_y = 68$, stretching 14.7%.

Figure 3: Channel flow. `svr`. $Re_\tau = 16\,000$. Velocity.



(a) Low-Re number IDDES grid.



(b) Wall function grid. New grid strategy.

Figure 4: Different grids. —— : grid lines; ●: cell center for traditional wall function grid.

into one large wall-adjacent cell where the cell center should be located at $10 \leq y^+ \leq 100$. This strategy was used in Bäckar and Davidson (2017) for channel flow and impinging jets (RANS). The new grid strategy is used in all simulations presented below.

Figure 5 presents the predicted velocity profiles for two different grids using three different wall functions where the friction velocity is computed using ML, the log-law/linear law and Reichardt's law. The ML wall function gives slightly better agreement than Reichardt's law and much better than the standard log/linear law. The markers show the location of the cell centers (compare Fig. 4b). The ratio of the cell size of the second wall-adjacent cell to the first is given in the caption.

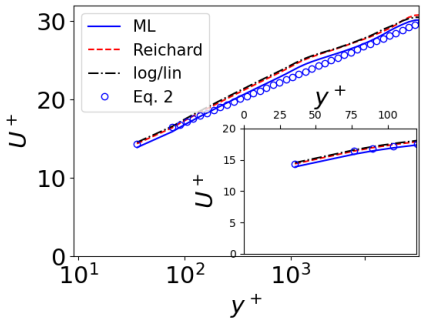
Next, I present predictions of a flat-plate boundary layer. The flat-plate boundary layer is $Re_\theta = 2\,550$. The mean inlet profiles are taken from a 2D RANS solution and synthetic fluctuations (Shur et al., 2014; Carlsson et al., 2022) are superimposed. The mesh has $315 \times 82 \times 32$ cells (x, y, z). The domain size is $8.5 \times 4.6 \times 3.2$. The first term on the right-side of Eq. 1 is omitted. From the second wall-adjacent cell, the grid is stretched by 10% for $\Delta y < 0.51$ and $y > 2.3$ but Δy is not allow to exceed 0.1. In the streamwise direction $\Delta x_{in} = 0.086$ and a 0.1% stretching is used. The inlet boundary-layer thickness is $\delta_{in} \simeq 0.8$.

Figure 6 presents the skin friction and the mean velocity. All three wall functions over-predict the skin friction. However, when y^+ is taken at the third wall-adjacent cell (Figure 7), the predictions are improved. As for the channel flow, the ML model performs better than the Reichardt's law and the standard log/linear wall functions.

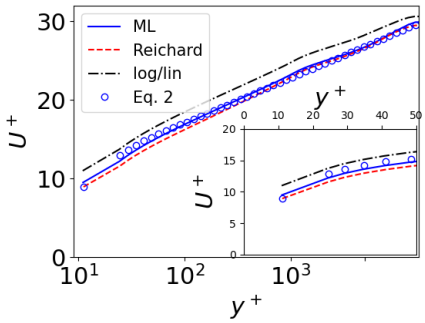
7 Conclusions

A Machine Learning (ML) method (`svr`) is proposed for wall functions. IDDES is used for creating the training data. The IDDES is also used when doing the wall-function simulations. Good results are obtained for channel flow and flat-plate boundary for the Machine-Learning-Based wall functions, slightly better than wall functions based on Reichardt's law and much better than the shandard log law. In the channel-flow simulation the results are insensitive if y^+ used for obtaining U^+ is taken at the first, second or third wall-adjacent cells. For the flat-plate boundary layer, better results are obtained by placing using $\bar{U}$ and y^+ at the third wall-adjacent cell than at the first.

Instantaneous IDDES data have been used for training `svr` (green markers in Fig. 8a) and the predicted `svr` data (ML model) are shown with red markers; the `svr` data follow the time-averaged IDDES data (blue line in Fig. 8a)). If I'm interested in predicting the *instantaneous* u_τ, I could find try to find nearest neighbour using Python's `scipy.spatial.KDTree` (shown by ● in Fig. 8b). The error between the `svr` predicted data and the ID-

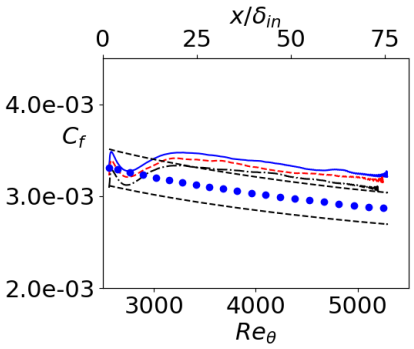


(a) $N_y = 78$. $y_1^+ = 35$. $\frac{(\Delta y)_1}{(\Delta y)_2} \simeq 6$.

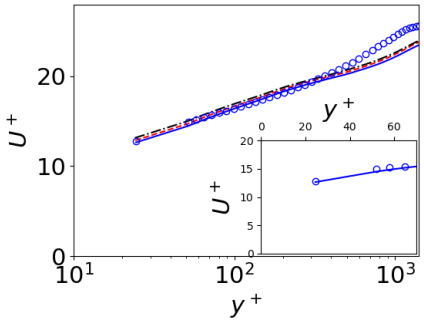


(b) $N_y = 92$. $y_1^+ = 11$. $\frac{(\Delta y)_1}{(\Delta y)_2} \simeq 5$.

Figure 5: Channel flow. $Re_\tau = 16\,000$. Velocity.

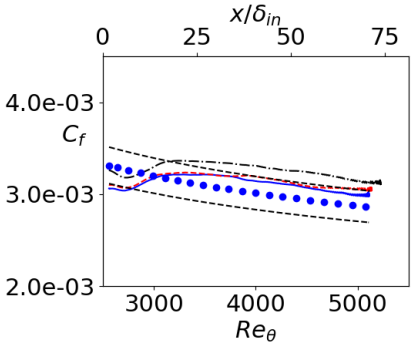


(a) ●: $2\left(\frac{\ln(Re_\theta)+4.127}{0.384}\right)^{-2}$; ▬ ▬ : ±6%.

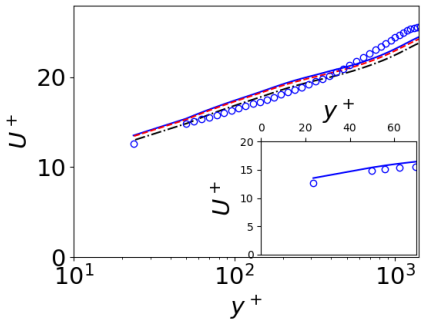


(b) Velocity at $Re_\theta = 4\,000$. ○: DNS (Sillero et al., 2014)

Figure 6: Boundary layer flow. $y_1^+ = 24$. $(\Delta y)_1/(\Delta y)_2 \simeq 8.4$.

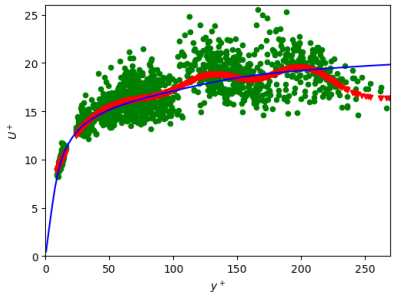


(a) ●: $2\left(\frac{\ln(Re_\theta)+4.127}{0.384}\right)^{-2}$; ▬ ▬ : ±6%.

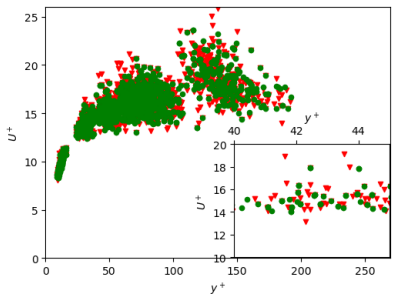


(b) Velocity at $Re_\theta = 4\,000$. ○: DNS (Sillero et al., 2014)

Figure 7: Boundary layer flow. The input, y^+, for all wall functions are taken at the the thrird wall-adjacent cells.



(a) ▬▬ : $\langle \bar{u} \rangle$, IDDES; ▼: `svr`: ●: IDDES, target data. 9% normalized error.



(b) Nearest neighbor using Python's `scipy.spatial.KDTree` ▼: KDTree; ●: IDDES, target data; 0.7% normalized error.

Figure 8: Conmparing `Ktree` and `svr`

DES data is then reduced from 9% (svr, Fig. 8a) to 0.7%, Fig. 8b). I call this model a *data-driven* wall function. I have evaluated this model for the present test cases and the predicted results are almost identical to those presented above.

An alternative could be to train the ML using *time-averaged* data rather than instantaneous. In this case many time-averaged $\langle\bar{u}\rangle$ (and $\langle\bar{p}\rangle$) profiles can be used for training, both in attached and separated flows. I have evaluated this model for the present test cases and the predicted results are almost identical to those presented above.

I'm currently creating databases at $Re_\tau = 2\,000$ of flow in diffusers using well-resolved LES with the WALE model. Inlet boundary conditions are taken from a pre-cursos LES of channel flow. The upper boundary of the diffuser is a slip wall with opening angles between 2 and 10 degrees. The plan is that the databases will be used for training ML a wall function using y^+ and the streamwise pressure gradient as input.

Acknowledgments

This study was partly financed by *Strategic research project on Chalmers on hydro- and aerodynamics*.

The computations were enabled by resources provided by the Swedish National Infrastructure for Computing (SNIC) at NSC partially funded by the Swedish Research Council through grant agreement no. 2018-05973.

8 References

K. Abe, T. Kondoh, and Y. Nagano. A new turbulence model for predicting fluid flow and heat transfer in separating and reattaching flows - 1. Flow field calculations. *Int. J. Heat Mass Transfer*, 37(1):139–151, 1994.

J.-A. Bäckar and L. Davidson. Evaluation of numerical wall functions on the axisymmetric impinging jet using OpenFOAM. *Int. J. Heat and Fluid Flow*, 67:27–42, 2017.

H.J Bae and P. Koumoutsakos. Scientific multi-agent reinforcement learning for wall-models of turbulent flows. *Nat Commun*, 13(1443), 2022. doi: https://doi.org/10.1038/s41467-s̄022-s̄28957-s̄7. URL https://doi.org/10.1038/s41467-s̄022-s̄28957-s̄7.

M. Carlsson, L. Davidson, S.-H. Peng, and S. Arvidson. Investigation of turbulence injection methods in large eddy simulation using a compressible flow solver. In *AIAA Science and Technology Forum and Exposition, AIAA SciTech Forum*, 2022.

L. Davidson. pyCALC-LES: a Python code for DNS, LES and Hybrid LES-RANS. Division of Fluid Dynamics, M2, Chalmers University of Technology, Gothenburg, 2021.

L. Davidson. Using Machine Learning for formulating new wall functions for Large Eddy Simulation: A first attempt. Technical report, Division of Fluid Dynamics, M2, Chalmers University of Technology, Gothenburg, 2022a.

L. Davidson. Using Machine Learning for formulating new wall functions for Large Eddy Simulation: A second attempt. Technical report, Division of Fluid Dynamics, M2, Chalmers University of Technology, Gothenburg, 2022b.

J. Dominique, J. Van den Berghe, C. Schram, and M. A. Mendez. Artificial neural networks modeling of wall pressure spectra beneath turbulent boundary layers. *Physics of Fluids*, 34(3):035119, 2022. doi: 10.1063/5.0083241. URL https://doi.org/10.1063/5.0083241.

M. Lee and R. D. Moser. Direct numerical simulation of turbulent channel flow up to $Re_\tau \approx 5200$. *J. of Fluid Mechanics*, 774:395–415, 2015. doi: 10.1017/jfm.2015.268. URL https://doi.org/10.1017/jfm.2015.268.

A. Lindholm, N. Wahlström, F. Lindsten, and T. Schön. *Machine Learning: A First Course for Engineers and Scientists*. Cambridge University Press, 2022. ISBN 9781108843607. doi: 10.1017/9781108919371. URL http://smlbook.org/.

J. Ling, M. F. Barone, W. Davis, K. Chowdhary, and J. Fike. Development of machine learning models for turbulent wall pressure fluctuations. In *55th AIAA Aerospace Sciences Meeting*, 2017. doi: 10.2514/6.2017-s̄0755. URL https://arc.aiaa.org/doi/abs/10.2514/6.2017-s̄0755.

L. N. Olson and J. B. Schroder. PyAMG: Algebraic multigrid solvers in Python v4.0, 2018. URL https://github.com/pyamg/pyamg. Release 4.0.

M. Rachana, J. Ramalingam, G. Ramana, A. Tejaswi, S. Mamidala, and G Srikanth. Fraud detection of credit card using machine learning. *GIS-Zeitschrift für Geoinformatik*, 8:1421–1436, 10 2021.

S. S Rao and S. R Desai. Machine learning based traffic light detection and ir sensor based proximity sensing for autonomous cars. In *Proceedings of the Int. Conf. on IoT Based Control Networks & Intelligent Systems – ICICNIS*, 2021. URL http://dx.doi.org/10.2139/ssrn.3883931.

M. Shur, P.R. Spalart, M.K. Strelets, and A.K. Travin. Synthetic turbulence generators for RANS-LES interfaces in zonal simulations of aerodynamic and aeroacoustic problems. *Flow, Turbulence and Combustion*, 93:69–92, 2014.

M. L. Shur, P. R. Spalart, M. Kh. Strelets, and A. K. Travin. A hybrid RANS-LES approach with delayed-DES and wall-modelled LES capabilities. *Int. J. of Heat and Fluid Flow*, 29:1638–1649, 2008. URL https://doi.org/10.1016/j.ijheatfluidflow.2008.07.001.

J.A. Sillero, J. Jimenez, and R.D. Moser. One-point statistics for turbulent wall-bounded flows at Reynolds numbers up to $\delta^+ \simeq 2000$. *Physics of Fluids*, 25(105102), 2014.

Lorenzo Tieghi, Alessandro Corsini, Giovanni Delibra, and Francesco Aldo Tucci. A machine-learnt wall function for rotating diffusers. volume Volume 1: Aircraft Engine; Fans and Blowers of *Turbo Expo: Power for Land, Sea, and Air*, 09 2020. doi: 10.1115/GT2020-s̄15353. URL https://doi.org/10.1115/GT2020-s̄15353.