Hybrid LES/RANS for flows including separation: a new wall function using machine learning based on binary search trees

Citation for the original published paper (version of record):
Davidson, L. (2025). Hybrid LES/RANS for flows including separation: a new wall function using machine learning
based on binary search trees. Journal of Turbulence, 26(2-3).
http://dx.doi.org/10.1080/14685248.2025.2476430

Journal of Turbulence

ISSN: 1468-5248 (Online) Journal homepage: www.tandfonline.com/journals/tjot20

Hybrid LES/RANS for flows including separation: a new wall function using machine learning based on binary search trees

Lars Davidson

To cite this article: Lars Davidson (2025) Hybrid LES/RANS for flows including separation: a new wall function using machine learning based on binary search trees, Journal of Turbulence, 26:2-3, 2476430, DOI: 10.1080/14685248.2025.2476430

To link to this article: https://doi.org/10.1080/14685248.2025.2476430

Published online: 14 Mar 2025.

JOURNAL OF TURBULENCE
2025, VOL. 26, NOS. 2–3, 90–111
https://doi.org/10.1080/14685248.2025.2476430

Taylor & Francis
Taylor & Francis Group

Hybrid LES/RANS for flows including separation: a new wall function using machine learning based on binary search trees

Lars Davidson

Division of Fluid Dynamics, Mechanics and Maritime Sciences (M2), Chalmers University of Technology, Gothenburg, Sweden

ABSTRACT

Machine Learning (ML) is used for developing wall functions for Improved Delayed Detached Eddy Simulations (IDDES). The ML model is based on **KDTree** which essentially is a fast look-up table. It searches the nearest target datapoint(s) for which y^+ and U^+ are closest to the CFD y^+ and U^+ cells. The target y^+ value gives the friction velocity. Two target databases – diffuser flow with opening angle $\alpha = 15^\circ$ and hump flow – are created from time-averaged data of wall-resolved IDDES (WR-IDDES, i.e. wall-adjacent cells at $y^+ < 1$). The new ML wall function is used to predict five test cases: diffuser flow with opening angles $\alpha = 15^\circ$ and $\alpha = 10^\circ$, the hump flow, channel flow at $Re_\tau = 16{,}000$ and flat-plate boundary layer. A novel grid strategy is used. The wall-adjacent cells are large. But further away from the wall, the wall-normal cell distribution is identical to that of a WR-IDDES grid. This new grid is found to improve the predictions compared to a standard wall-function grid. It is found that the number of cells for a wall-resolved IDDES grid (grid stretching 15%) is a factor of $0.2\ln(Re_\tau)$ larger than that of a standard wall-functions mesh (constant wall-normal grid cells). The new ML wall function is found to perform well compared to the WR-IDDES and better than the Reichardt's wall function.

ARTICLE HISTORY
Received 26 October 2024
Accepted 2 March 2025

KEYWORDS
Large Eddy Simulations; detached Eddy Simulations; wall functions; machine learning; binary search trees

1. Introduction

Wall-resolved Large Eddy Simulation (LES) in which the wall-adjacent cell centre are located at $y^+ < 1$ is affordable only at low Reynolds number (x, y, z denote streamwise, wall-normal and spanwise direction, respectively; $y^+ = u_\tau y/\nu$ where u_τ and ν denote friction velocity and kinematic viscosity, respectively). At high Reynolds number, LES can be combined with a URANS treatment of the near-wall flow region. There are different methods for combining LES and URANS such as Detached Eddy Simulation (DES) [1–3], hybrid LES/RANS [4, 5] and Scale-Adapted Simulations (SAS) [6, 7]; for a review, see Fröhlich and von Terzi [8]. The two first classes of models take the SGS length scale from the cell size whereas the last (SAS) involves the von Kármán lengthscale.

In DES, the interface between the LES region and the URANS region is supposed to be located in the outer part of the boundary layer. However, it was found that in some flows – for example, near the trailing edge of a airfoil – it may occur that DES locates the interface too close to the wall because the streamwise grid size is small. Hence the flow in the inner part of the boundary layer is treated in LES mode with too a coarse mesh. This results in a poorly resolved LES and inaccurate predictions. Different proposals have been made to modify DES. The S-A DES model is modified by replacing the original DES lengthscale $\tilde{d}$ with $\tilde{d}_{DDES} = d - f_d \max(0, d - C_{DES}\Delta)$ [9] where d is the wall distance, C_{DES} is a constant, Δ is the LES lengthscale and f_d is a blending function which is zero at the wall and one in the LES region. In the $k-\omega$ SST DDES model (k and ω denote modelled turbulent kinetic energy and specific dissipation, respectively), a shielding function is introduced which protects the boundary layer [10, 11]. IDDES is an Improved DDES [3], see Section 2.3.

In the works cited above, URANS is used all the way to the wall (the wall-adjacent cells centres are located at $y^+ < 1$). However, when making LES with wall functions (i.e wall modelled LES or wall stress models [12]),

CONTACT Lars Davidson lada@chalmers.se

the wall-adjacent cells are located in the fully turbulent region. The first work on LES and wall functions is that of Deardorff [13]. The paper presents LES in channel flow with periodic boundary conditions.

The wall distance and the local velocity are usually input parameters in wall function and they are normally taken at the wall-adjacent cells. Kawai and Larsson [12], Larsson et al. [14] proposed to take these two quantities further away from the wall. The argument is that the LES is not well-resolved at the wall-adjacent cells.

It is interesting to compare the grid resolution requirements for different wall treatments. Choi and Moin [15] find that the number of cells, N, for a wall-modelled LES (cubic cells) of a flat-plate boundary layer can be estimated as $N \sim Re_L$ and for wall-resolved LES as $N \sim Re_L^{13/7}$ (Re_L is the Reynolds number based on the free-stream velocity and the length of the plate). Yang and Griffin [16] reports slightly higher values, $N \sim Re_L^{1.14}$ and $N \sim Re_L^{2.72}$, respectively. Larsson et al. [14] makes an estimate of the resolution requirement for wall-resolved LES of a NACA0012 at an angle of attack of 2.5° at three different Reynolds numbers. They take into account that the boundary-layer thickness varies along the surfaces. Their approximations are $N \sim Re_c^{0.4}$ and $N \sim Re_c^2$ (Re_c denotes the Reynolds number based on the free-stream velocity and the chord) for the outer and inner part of the boundary layer, respectively.

Agrawal et al. [17] investigates the grid-resolution requirements for separated flow. They define a length scale based on the stream-wise pressure gradient $l_p = \nu/u_p$ where $u_p = ((\nu/\rho)dP/dx)^{1/3}$ (P and ρ denote pressure and density, respectively). They report that in flows which experience a stream-wise pressure gradient, the friction velocity is smaller than u_p (i.e. $u_\tau < u_p$) which means that the timescale related to the pressure gradient is faster than the viscous timescale. Hence, the authors define a resolution requirement based on the stream-wise pressure gradient. They find that the minimum resolution requirement is $\Delta / \min(l_p) \sim 10$, where Δ denotes the cell size.

The estimates of required grid resolution for wall-function meshes are given above. Now we will make an estimate of a hybrid LES/URANS mesh for which the wall-adjacent cell centres are located at $y^+ < 1$. We set the wall parallel cell sides to $\Delta x = \Delta z = 1/N_0$ and $\Delta y_{\max} = 1/N_0$ (when making the estimates of required grids for wall-function meshes in Choi and Moin [15], Δy was taken as constant and set to $1/N_0$). A geometric grid stretching, γ , is used in the wall-normal direction but the stretching is stopped when $\Delta y = 1/N_0$. If the stretching is low, $\Delta y < 1/N_0$ for all cells. This occurs when γ and N_0 satisfies the equation (see Appendix 1)

$$Re_\tau(1-\gamma) = 1 - \gamma^{\ln(Re_\tau/N_0)/\ln(\gamma)} \tag{1}$$

and when γ is even smaller. Re_τ is the friction Reynolds number based on the friction velocity and the boundary-layer thickness, δ. It is shown in Appendix 1 that the ratio of the number of required cells in the wall normal-direction, N_y, for a wall-stretched grid to that for a wall-function grid is $\ln(Re_\tau)$ when Equation (1) is satisfied ($\gamma \simeq 1.04$) and $0.2\ln(Re_\tau)$ for $\gamma = 1.15$.

Hence it is found – asymptotically – that the reduction in CPU cost is rather small. Nevertheless, Herr and Probst [18] reports an CPU reduction of 61% when using wall functions compared to using WR-DES. Other advantages of using wall functions is that the viscous restriction on the time step in compressible solvers is reduced. Furthermore, the ratio of the cell sides (i.e. the ratio of the streamwise cell side to the wall-normal one) is also reduced which means that the condition number of matrix, M, in the discretised equations $MU = b$ (U and b are the solution vector (velocity components, pressure …) and the right-hand side, respectively) is reduced. Another advantage is that some works [12, 14, 17, 19] report that the problem of log-layer mismatch (LLM) is eliminated when using wall functions.

There are not too many studies in the literature on ML for improving wall functions. Zhou et al. [20] use Neural Network to train an ML wall function. They make wall-resolved DES (i.e. $y^+ < 1$ for the wall-adjacent cells) of periodic hill flow at three different Reynolds numbers. They use instantaneous snapshots to train the ML wall function. The output parameter is the wall-stress vector and the input parameters are the wall-normal distance, the velocity vector and the pressure gradient. They use the ML wall function for predicting channel flow at $Re_\tau = 1000$ and $Re_\tau = 5200$ and they find that the ML wall function performs very well. Then they apply the ML wall function to the periodic and they get very poor agreement with wall-resolved LES. In Tieghi et al. [21] they use a time-averaged high-fidelity IDDES simulation to train a neural network for improving the predicted modelled turbulent kinetic used in wall functions (RANS). In Ling et al. [22] 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. Dominique et al. [23] use neural network to predict the wall pressure spectra. Their input data are boundary-layer thicknesses (physical, displacement and momentum), streamwise pressure gradient and wall shear stress which are taken from experiments and high-fidelity DNS/LES in the literature. In Bae and Koumoutsakos [24] they use a 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. Davidson [19] presents a new ML wall-function based on Support Vector Regression (SVR). The SVR is trained using instantaneous y^+ and U^+ in fully developed channel flow at $Re_\tau = 5200$. The ML model is then used for predicting channel flow at $Re_\tau = 16{,}000$ and a flat-plate boundary layer. Good agreement with experimental data is obtained.

A novel grid strategy is used in the present work. The wall-adjacent cells (cell number $j = 0$) are large. But further away from the wall (cell number $j > 0$), the wall-normal cell distribution is identical to that of a WR-IDDES grid. The advantage of this grid strategy is that the gradients are much more accurately computed since the grid for cells $j > 0$ are much finer than for a standard wall-function grid. This new grid is found to improve the predictions compared to a standard wall-function grid.

Gritskevich et al. [25] made a comparison of IDDES with wall functions and WR-IDDES. They report that wall functions were essentially as accurate as WR-IDDES provided that the wall-normal grid spacing when using wall functions was smaller than one percent of the boundary-layer thickness. Their finding supports the use of the novel grid strategy.

Romanelli et al. [26] use machine-learning for improving wall-functions for RANS (Reynolds–Averagared Navier–Stokes). They split their computational grid (with wall-adjacent cell centres at $y^+ < 1$) into a near-wall region and an outer region. The separation line between the two regions is denoted the interface. No discretised equations are solved in the near-wall region, but the solution variables (velocity componenents, pressure, …) in this region are set and used as ghost cells and serve as boundary conditions for the outer region. Incidentally, the resulting grid strategy in their work is the same as the novel grid strategy employed in the present study.

The ML model used in the present work is based on Python's **KDTree** which essentially is a fast look-up table. **KDTree** computes the distance between the vectors as $\mathbf{X}_i$ (target database) and $\mathbf{x}_j$ (the wall-adjacent cells in the CFD simulation) for all samples i and j and finds the K smallest distances for each j. The vectors $\mathbf{X}_i$ and $\mathbf{x}_j$ have two columns, y^+ and U^+. The wall-shear stress for the wall-adjacent CFD cell j is obtained by finding the closest cell (in terms of y^+ and U^+) in the target database. The friction velocity is then obtained from y^+_{target} which is used for setting boundary conditions for the wall-parallel velocity, k and ε.

The paper is organised as follows. The numerical method and turbulence model are presented in the next section. Then we show how the two target databases are created using wall-resolved IDDES (WR-IDDES). In the next section we present the ML model followed by a description of the Reichardt's wall function. Then the results are presented and the paper ends with concluding remarks.

2. Numerical method and turbulence model

The finite volume code **pyCALC-LES** [27] is used. It is written in Python and is fully vectorised (i.e. no `for` loops). The solution procedure is based on fractional step. 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 discretisation. The code runs fully on the GPU. On a desktop, it runs approximately 50 times faster on the GPU (Nvidia RTX A6000) than on the CPU (Intel i7-13700) for a hump flow simulation using the present ML wall function.

2.1. The governing equations

The equations read

$$\frac{\partial \bar{v}_i}{\partial x_i} = 0 \tag{2}$$

$$\frac{\partial \bar{v}_i}{\partial t} + \frac{\partial \bar{v}_i \bar{v}_j}{\partial x_j} = -\frac{1}{\rho}\frac{\partial \bar{p}}{\partial x_i} + \frac{\partial}{\partial x_j}\left[(\nu + \nu_t)\frac{\partial \bar{v}_i}{\partial x_j}\right] \tag{3}$$

where $\bar{v}_i, \rho, p, \nu, \nu_t$ denote velocity vector, density, pressure, kinematic viscosity and kinematic, turbulent (i.e. modelled) viscosity, respectively.

2.2. The underlying RANS turbulence model

The AKN low-Reynolds number for IDDES (see Section 2.3) reads [28]

$$\begin{aligned}
\frac{\partial k}{\partial t} + \frac{\partial \bar{v}_j k}{\partial x_j} &= \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\nu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right] + P_k - \psi\varepsilon \\
\frac{\partial \varepsilon}{\partial t} + \frac{\partial \bar{v}_j \varepsilon}{\partial x_j} &= \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\nu_t}{\sigma_\varepsilon}\right)\frac{\partial \varepsilon}{\partial x_j}\right] + C_{\varepsilon 1} P_k \frac{\varepsilon}{k} - C_{\varepsilon 2} f_2 \frac{\varepsilon^2}{k} \\
\nu_t = C_\mu f_\mu \frac{k^2}{\varepsilon}, &\quad P_k = \nu_t\left(\frac{\partial \bar{v}_i}{\partial x_j} + \frac{\partial \bar{v}_j}{\partial x_i}\right)\frac{\partial \bar{v}_i}{\partial x_j} \\
C_{\varepsilon 1} = 1.5, \quad C_{\varepsilon 2} = 1.9, &\quad C_\mu = 0.09, \quad \sigma_k = 1.4, \quad \sigma_\varepsilon = 1.4
\end{aligned} \tag{4}$$

where k and ε denote the modelled turbulent kinetic energy and its dissipation, respectively. ψ is defined in Equation (6). The damping functions are defined as

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

where d denote the distance to the wall. The wall boundary condition is implemented by setting ε at the wall-adjacent cells as

$$\varepsilon = 2\nu\frac{k}{d^2} \tag{5}$$

When wall functions are used, the boundary conditions are set according to Equation (17).

2.3. The IDDES model

The Improved Delayed Detached Eddy Simulation method [3] is used when creating the database as well as when using wall functions.

The coefficient ψ in Equation (4) is computed as

$$\psi = \frac{l_u}{L_{hyb}}, \quad l_u = \frac{k^{3/2}}{\varepsilon} \tag{6}$$

where L_{hyb} is the usual IDDES length scale [3]. The IDDES is an improved DES model which is a hybrid RANS-LES method in which URANS is used near the wall and LES is used further away from the wall. The IDDES includes two branches, WM-LES (wall-modelled LES) and DDES (delayed DES). In WM-LES, the interface between URANS and LES is located in the logarithmic regin whereas in DDES the entire boundary layer is treated in URANS mode. In the present work, the WM-LES branch is used in all simulations.

For convenience, the procedure how to obtain the IDDES length scale is summarised below.

$$L_{hyb} = f_d(1+f_e)l_u + (1-f_d)l_c, \quad l_c = C_{DES}\Delta \tag{7}$$

where the Δ length scale is defined as

$$\Delta = \min\left\{\max\left[C_w d_w, C_w h_{\max}, h_{wn}\right], h_{\max}\right\}$$

and $C_w = 0.15$, d_w is the distance to the closest wall and h_{wn} is the grid step in the wall normal direction. The blending functions f_d and f_e read

$$f_d = \max\left\{\left(1-f_{dt}\right), f_B\right\} \tag{8}$$

$$f_e = \max\left\{\left(f_{e1}-1\right), 0\right\} f_{e2} \tag{9}$$

where the functions f_{dt} and f_B entering Equation (8) are given by

$$f_{dt} = 1 - \tanh\left[(8r_{dt})^3\right] \tag{10}$$

$$f_B = \min\left\{2\exp\left(-9\alpha^2\right), 1\right\} \tag{11}$$

with

$$\alpha = 0.25 - d_w/h_{\max}.$$

The functions f_{e1} and f_{e2} in Equation (9) read

$$f_{e1} = \begin{cases} 2\exp\left(-11.09\alpha^2\right) & \text{if } \alpha \geq 0 \\ 2\exp\left(-9\alpha^2\right) & \text{if } \alpha < 0 \end{cases}$$

and

$$f_{e2} = 1 - \max\left\{f_t, f_l\right\}$$

where the functions f_t and f_l are given by

$$f_t = \tanh\left[\left(c_t^2 r_{dt}\right)^3\right]$$

$$f_l = \tanh\left[\left(c_l^2 r_{dl}\right)^{10}\right].$$

The constants c_t and c_l given the same values as in the $k-\omega$ SST model, i.e. $c_t = 1.87$ and $c_l = 5$ [3]. The quantities r_{dt} (also entering Equation (10)) and r_{dl}, are defined as follows

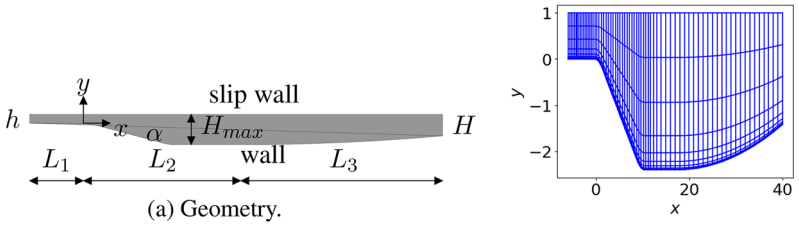
$$r_{dt} = \frac{\nu_t}{\kappa^2 d_w^2 \max\left\{|\bar{s}|, 10^{-10}\right\}}$$

$$r_{dl} = \frac{\nu}{\kappa^2 d_w^2 \max\left\{|\bar{s}|, 10^{-10}\right\}}$$

3. Creating the database

A database for **KDTree** is created by making a simulation of the flow in a diffuser, see Figure 1(a). The diffuser flow is selected because it is a relatively simple test case including a flow region with adverse-pressure gradient. Compared to channel flow and flat-plate boundary layer flow, it adds the complexity of a small re-circulation and $dP/dx > 0$. At first, a well-resolved LES was made with a grid of 600 × 150 × 300 cells with $0.3 < \Delta y^+ < 22$ ($\Delta z^+ = 11$, $\Delta x^+ = 22$ at the inlet) at a low Reynolds number ($Re_\tau = 2000$ at the inlet). The disadvantage of this procedure is that we cannot require an IDDES with wall functions to perform as well as a well-resolved LES. In the former case, inaccuracies may appear not only due to the wall functions but also due to insufficient resolution and errors due to the turbulence model (i.e. IDDES).

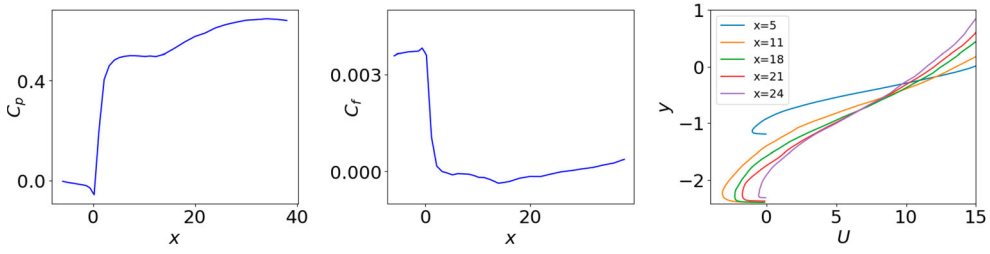
Instead, we use WR-IDDES (i.e. wall-adjacent cells at $y^+ < 1$) with the underlying AKN $k-\varepsilon$ turbulence model (see Sections 2.3 and 2.2) when creating the database. The advantage is that when validating



(a) Geometry.

(b) Grid, $x-y$ plane (not to scale). 700×90 cells. Every 10^{th} grid line is shown.

Figure 1. Diffuser, $\alpha = 15^{\circ}$. (a) Geometry and (b) Grid, x—y plane (not to scale). 700 × 90 cells. Every 10th grid line is shown.



(a) Pressure coefficient. (b) Skin friction. (c) Velocity.

Figure 2. Diffuser flow. Target database. (a) Pressure coefficient. (b) Skin friction and (c) Velocity.

the **KDTree** wall function, we can use an identical grid in the wall-parallel planes. In this case it is reasonable to require that the wall-functions should give similar results as the WR-IDDES.

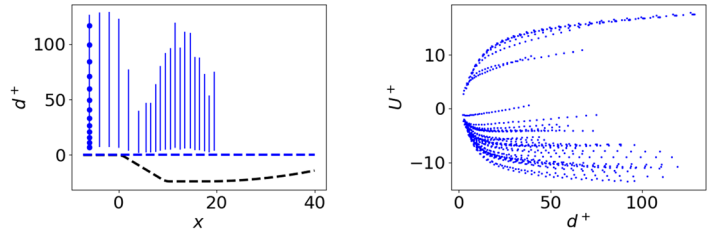
Figure 1(a) shows the domain of the diffuser which is used for creating the first database. All length are made non-dimensional using the inlet height, h. The opening angle is 15°. The maximal height is $H_{\max} = 3.4$. The length of the straight channel downstream of the inlet is $L_1 = 6$. The extent of the diffuser and the following straight channel is $L_2 = 17.5$. The length of the weak contraction is $L_3 = 22.5$; the contraction is used in order to avoid backflow at the outlet. The friction Reynolds number at the inlet is $Re_\tau = 5200$ based on the friction velocity and the inlet height, h.

The boundary conditions are set as follows. The top and lower boundaries are slip and no-slip wall, respectively. At the wall, $\bar{u} = \bar{v} = \bar{w} = k = 0$ and the dissipation is given by Equation (5). Periodic boundary conditions are set for all variables in the spanwise direction (z). Homogeneous Neumann boundary conditions are used for all variables at the outlet. A pre-cursor WR-IDDES of channel flow (half-channel width) is used for setting the instantaneous inlet boundaries conditions for all variables except the pressure. The boundary condition for pressure at the four boundaries in Figure 1(a) is homogeneous Neumann. The grid has 700 × 90 × 96 in x (streamwise), y (wall-normal) and z (spanwise) directions.

Figure 2 shows the predicted pressure coefficient, skin friction and velocity profiles at $x = 5$, 11, 18, 21 and $x = 24$. The diffuser creates a pressure increase of approximately half a dynamic pressure height and the flow is separated between $x = 3.15$ and $x = 27$.

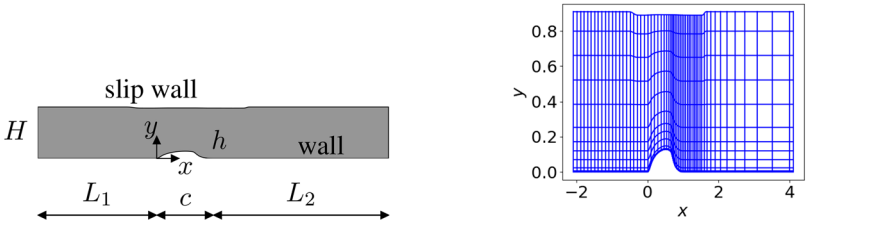
The created database consists of 41 time-averaged profiles of U^+ vs. d^+ with 26 points in each profile, see Figure 3(a) (d denotes the wall distance). The bullets show every second d location. The vertical lines show the x location of the profiles and the extent of these lines show the largest d^+ at each x; the largest wall distance is approximately the same for each x location since the largest d is defined by a wall-parallel grid line. Figure 3(b) shows that the magnitudes of the separated (negative) U^+ velocity profiles are similar to those of the attached profiles. The reason for the large magnitude of the separated velocity profiles is that the friction velocity, u_τ, is small in this region.

A second target database is created by computing the flow over a hump. The Reynolds number is $Re_c = 936{,}000$ based on the inlet bulk velocity and the length of the hump, c. Figure 4(a) shows the geometry. All length and velocities are scaled with the hump length, c, and the inlet bulk velocity, respectively. It is a two-dimensional hump mounted between two glass endplate frames and both leading edge and trailing edges are



(a) Data points of d^+ vs. x. (b) Scatter plot of U^+ vs. d^+.

Figure 3. Diffuser flow. d is the wall distance. •: location of cell centres. The dashed black line shows the contour of the lower wall (not to scale). Every second x line and d point are shown. (a) Data points of d^+ vs. x and (b) Scatter plot of U^+ vs. d^+.



(a) Geometry. Height of hump, $h = 0.128$. $H = 0.909$, $L_1 = 2.1$, $L_2 = 4.1$.

(b) Grid, $x - y$ plane. Every 10^{th} grid line is shown.

Figure 4. Hump flow. (a) Geometry. Height of hump, $h = 0.128$. $H = 0.909$, $L_1 = 2.1$, $L_2 = 4.1$ and (b) Grid, x–y plane. Every 10th grid line is shown.

faired smoothly with a wind tunnel splitter plate. The experiments were carried out at NASA [32, 33] (see also [34]). It is a challenging flow involving developing boundary layer, accelerating flow, separation and a recovery region.

This test case has been used in both NASA [35] workshop as well as EU projects such as ATAACK [36] and Go4Hybrid [37]. Uzun and Malik [30] made wall-resolved LES using an SGS model by Vreman [38]. The largest computational grid had 850 million cells. They report good agreement with experiments.

The spanwise extent is $z_{\max} = 0.2$. The grid has $582 \times 128 \times 64$ cells (x, y, z), see Figure 4(b). The mean inlet velocity is taken from a pre-cursor 2D RANS of a flat-plate boundary layer. Instantaneous turbulent fluctuations are superimposed on the mean RANS velocity profile. The inlet k and ε are also taken from the 2D RANS simulation.

In order to reduce the modelled turbulence to a suitable LES level a commutation term is added to the k equation. The method was proposed for the PANS model [39]. Although the PANS model is not used in this work is was shown in Friess and Davidson [40] that IDDES and PANS can be formulated in a way so that PANS and IDDES are equivalent because of the equivalence criterion [41]. In PANS, the equation for the modelled turbulent kinetic energy, k, is derived by multiplying the k_{tot} equation ($k_{tot} = k_{res} + k$, where k_{res} denotes resolved, turbulent kinetic energy) by f_k where $f_k = k/k_{tot}$ (f_k denotes the ratio of modelled to total turbulent kinetic energy and is one in RANS regions and smaller than one ($0.2 \lesssim f_k \lesssim 0.6$) in LES regions). The convective term in the k equation – i.e. the left-hand side of the equation – with constant f_k is then obtained as (D/Dt denotes the material derivative)

$$f_k \frac{Dk_{tot}}{Dt} = \frac{D(f_k k_{tot})}{Dt} = \frac{Dk}{Dt} \tag{12}$$

Now, if f_k varies in space, we get instead

$$f_k \frac{Dk_{tot}}{Dt} = \frac{D(f_k k_{tot})}{Dt} - k_{tot}\frac{Df_k}{Dt} = \frac{Dk}{Dt} - k_{tot}\frac{Df_k}{Dt} \tag{13}$$

Consider this equation near the inlet. Since the normal vector of the inlets pointing into the domain in the present work is $(1, 0, 0)$ and f_k is constant in time, $Df_k/Dt = \bar{u}\partial f_k/\partial x$. The last term in Equation (13) is the

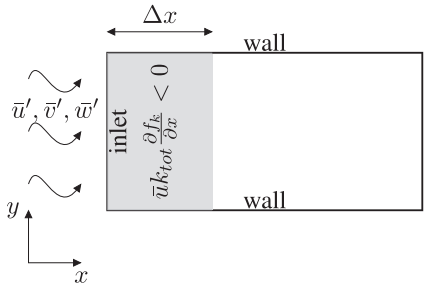


Figure 5. Setting inlet boundary conditions. Synthetic inlet fluctuations Shur et al. [29], $\bar{u}', \bar{v}'\bar{w}'$ are superposed on a mean RANS inlet velocity profile. The commutation term, $\bar{u}\partial f_k/\partial x$ (see Equation (14)), is added to the first cell layer.

commutation term which is discretised as

$$k_{tot}\bar{u}\frac{\partial f_k}{\partial x} = k_{tot}\bar{u}\frac{f_{k,0.5\Delta x} - f_{k,x=0}}{0.5\Delta x} \tag{14}$$

where Δx denotes the distance between the inlet and the cell centre adjacent to the inlet and the inlet is located at $x = 0$, see Figure 5. The value of f_k at the inlet is one (k_{RANS} is prescribed as inlet boundary condition) and f_k at the cell centre, $x = 0.5\Delta x$, is computed as [40]

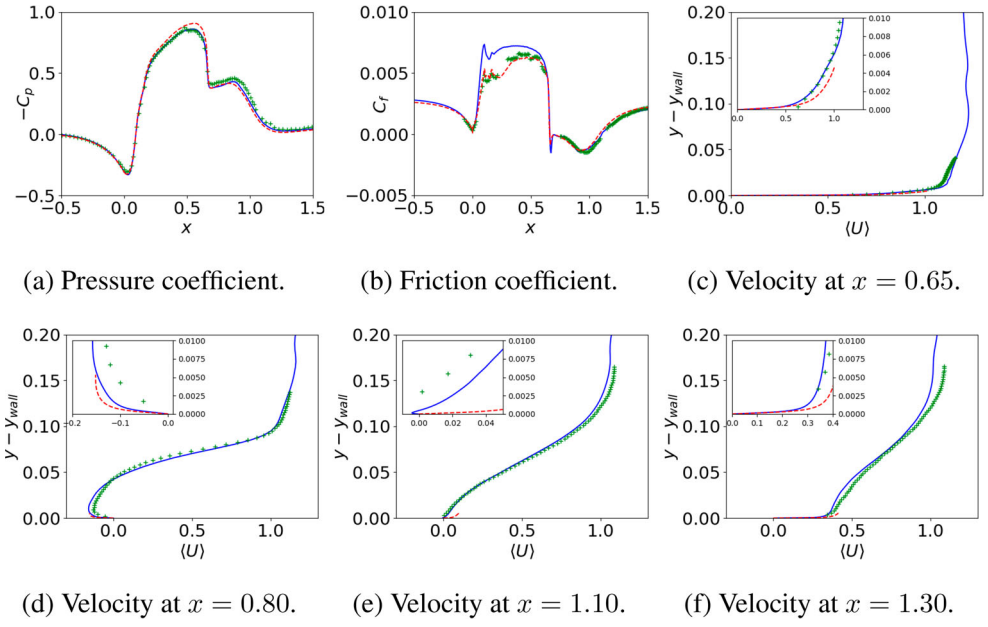
$$f_k = \min\left[1, \max\left(0, \left(\frac{C_{\varepsilon 2} - C_{\varepsilon 1}\psi}{C_{\varepsilon 2} - C_{\varepsilon 1}}\right)^{1/3}\right)\right] \tag{15}$$

where ψ is given by Equation (6). f_k in Equation (15) is smaller than one (because $x = 0.5\Delta$ is in the LES region because of the synthetic, fluctuating inlet boundary conditions) and hence the commutation term in Equation (14) is positive on the left-hand of the k equation and negative on the right-hand side thereby reducing k near the inlet from RANS values to LES values. The resolved part of k_{tot} in Equation (14) is computed as a running average. No commutation term is added to the ε equation since the modelled dissipation, ε, is the same in RANS and LES regions.

The created database consists of time-averaged profiles of U^+ vs. d^+ at all x grid lines with 26 points in each profile, see Figure 8(a). The minimum ($\simeq -100$) and maximum ($\simeq 70$) data points of U^+ in Figure 8(b) stem from the separation region at $x \simeq 0.7$ where u_τ is very small. Many more database points are used for this flow than for the diffuser flow (see Figure 3(a)) because this flow is much more complex.

The predicted pressure coefficient, skin friction and velocity profiles at are presented in $x = 0.65, 0.80, 1.10$ and $x = 1.30$, see Figure 6. The agreement with experiments is good. However, the predicted total, turbulent shear stress is over-predicted by almost a factor of two; the agreement further downstream is better. The predicted shear stresses, see Figure 7, are similar to those reported in Garbaruk et al. [37]. The WR-LES data by Uzun and Malik [30] and Azun [31] are also shown. The WR-LES data using the fine grid, wide domain and modified top wall contour are used. Note that data only for $y - y_{wall} < 0.03$ are available. It can be seen that the skin friction for $0.1 < x < 0.3$ is much better predicted with WR-LES than WR-IDDES. The flow is accelerating in this region and the low C_f indicates that the turbulence is perhaps re-laminarising. Uzun and Malik [30] report that this indeed is the case since the relaminarisation parameter, $K \simeq 4.87 \cdot 10^{-6}$, is larger than the value $K = 3 \cdot 10^{-6}$ at which relaminarisation takes place. Although the WR-LES predicts the skin friction and the pressure coefficient in very good agreement with experiments it over-predicts the turbulence shear stress at $x = 0.65$ by a factor of three.

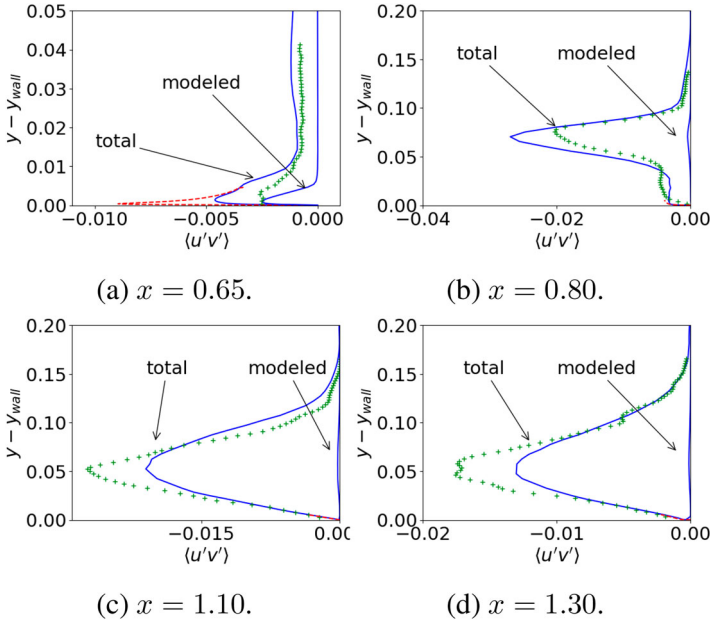
In Davidson [19] a new grid strategy was proposed for wall-functions grids (it was also used in Paik and Sotiropoulos [42]). Figure 9 shows three grids. A WR-IDDES grid (Figure 9(a)), a standard wall-function grid (Figure 9(b)) and the new proposed wall-function grid (Figure 9(c)). The centre of the first cell in a WR-IDDES grid is located at $y^+ < 1$ and then the wall-normal cell size increases by approximately 10% away from the wall. In a standard wall-function grid, the centre of the first cell is located in the logarithmic region, say, $30 < y^+ < 400$, and then the cells may increase by 10% away from the wall (or the cell size may stay constant). In the new wall-function grid strategy, a number of the cells in the WR-IDDES grid (Figure 9(a)) are merged into one large cell. Further away from the wall, the grid is identical to the WR-IDDES grid. The advantage of



(a) Pressure coefficient. (b) Friction coefficient. (c) Velocity at $x = 0.65$.

(d) Velocity at $x = 0.80$. (e) Velocity at $x = 1.10$. (f) Velocity at $x = 1.30$.

Figure 6. Hump flow. —— : WR-IDDES, see Figure 9(a); – – : WR-LES [30, 31]; +: experiments [32]. (a) Pressure coefficient. (b) Friction coefficient. (c) Velocity at $x = 0.65$. (d) Velocity at $x = 0.80$. (e) Velocity at $x = 1.10$ and (f) Velocity at $x = 1.30$.



(a) $x = 0.65$. (b) $x = 0.80$.

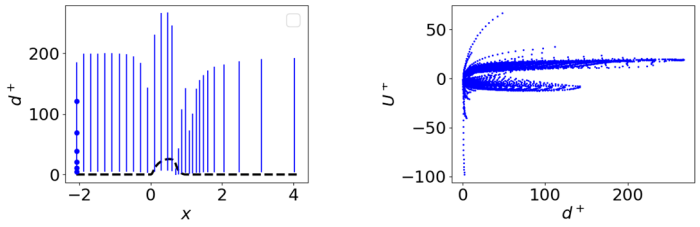
(c) $x = 1.10$. (d) $x = 1.30$.

Figure 7. Hump flow. Total turbulent shear stress. —— : WR-IDDES, see Figure 9(a); – – : WR-LES [30, 31]; +: experiments [32]. (a) $x = 0.65$. (b) $x = 0.80$. (c) $x = 1.10$ and (d) $x = 1.30$.

this new grid is that the velocity gradients away from wall (i.e. at $y > 0.005$, see Figure 9(c)) are resolved as accurately as at the WR-IDDES grid. Wall function grids such as that in Figure 9(c) are used unless otherwise stated.

4. Machine learning

We have two sets of data points. One is the target data set ($\mathbf{X} = [U^+_{\text{target}}, y^+_{\text{target}}]$) and the other one is the CFD data set ($\mathbf{x} = [U^+_{\text{CFD}}, y^+_{\text{CFD}}]$). The target data set is either taken from the diffuser flow (see Figure 3) or from the hump flow (see Figure 8). The CFD data set includes all wall-adjacent cells in the CFD simulation. The



(a) Target data points of d^+ vs. x. (b) Scatter plot of U^+ and d^+.

Figure 8. Hump flow. d is the wall distance. The dashed black line shows the contour of the lower wall (not to scale). The target database consists of time-averaged 582 profiles (all grid lines) of U^+ vs. d^+ with 24 points in each profile. Every 20th x line and every 4th d point are shown. (a) Target data points of d^+ vs. x and (b) Scatter plot of U^+ and d^+.

Python **KDTree** is used which computes the distance between the vectors as

$$\mathbf{d_s} = \mathbf{X}_i - \mathbf{x}_j \tag{16}$$

(see Line 49 in Listing 2 in Appendix 3) for all j and finds the K smallest distances for each j. Indices i and j denote the row of $\mathbf{X}$ and $\mathbf{x}$, respectively, which both have two columns. $\mathbf{X}_i$ (the database) is set at lines 21–24 in Listing 2) and $\mathbf{x}_j$ (the CFD values) is set at lines 44, 45 in Listing 2. In the present work we use $K = 1$ or $K = 5$. For $K = 1$, only the closest cell is used and for $K = 5$ the target is computed by averaging over the five closest points using the weight $1/|\mathbf{d_s}|$ (see Equation (16)). Unless otherwise stated, $K = 1$ is used for **KDTree** with hump-flow data and $K = 5$ is used for **KDTree** with diffuser flow data. The impact of using $K = 1$ or $K = 5$ is usually small; when it does have an impact we will present predictions using both options. For the boundary-layer flow we will show the influence of using $K = 1, 2, 5$ and $K = 10$.

The Python module `fix_k` in **pyCALC-LES** in which the ML wall function is implemented, is schematically given in Listing 2. It is called every global iteration.

The input parameters, y^+ and U^+, are taken either at the wall-adjacent cells (j_wall $=0$, see Line 31 in Listing 2) or further away from the wall (third cell) as suggested by Kawai and Larsson [12] and Larsson et al. [14]. Unless otherwise stated, the wall-adjacent cells are used.

As can be at the and of Listing 2, the output of this script is the friction velocity, u_τ. It is used for setting the boundary conditions for the wall-parallel velocity, k and ε as

$$\begin{aligned} \rho u_\tau^2 \quad &: \quad \bar{u} \text{ equation} \\ C_\mu^{-1/2} u_\tau^2 \quad &: \quad k \text{ equation} \\ \frac{u_\tau^3}{\kappa y} \quad &: \quad \varepsilon \text{ equation} \end{aligned} \tag{17}$$

The boundary condition for the velocity is a shear stress boundary condition. The modelled turbulent kinetic energy, k, and its dissipation, ε, are set at the wall-adjacent cells centres according to Equation (17). The complete Python script in Listing 2 and the databases can be found in Davidson [43].

It should be mentioned that databases with instantaneous data were also investigated (but no results are presented). In that case 200 data sets (independent in time) were created storing U^+ and y^+ at the same locations as in Figures 3(a) and 8(a). The instantaneous databases were hence 200 times larger than the time-averaged ones. We had hoped that instantaneous databases should be better than time-averaged data in capturing large, unsteady characteristics of separated flow regions. However, it was found that the instantaneous databases gave slightly worse results than the time-averaged ones.

5. Standard wall functions

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

$$\frac{\bar{u}_P}{u_\tau} \equiv U^+ = \frac{1}{\kappa} \ln(1 - 0.4y^+) + 7.8 \left[1 - \exp\left(-y^+/11\right) - (y^+/11) \exp\left(-y^+/3\right)\right] \tag{18}$$

The friction velocity is then obtained by re-arranging Equation (18) and solving the implicit equation

$$u_\tau - \bar{u}_P \left\{ \ln(1 - 0.4y^+)/\kappa + 7.8 \left[1 - \exp\left(-y^+/11\right) - (y^+/11) \exp\left(-y^+/3\right) \right] \right\}^{-1} = 0 \tag{19}$$

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

6. Results

6.1. Diffuser flow

Now we will validate the **KDTree** wall functions. The diffuser flow with 15° opening, i.e. the same test case as the target data, is the first test case. The first 21 cells near the wall are merged into one large cell, see Figure 9(c). We use the same grid in the wall-parallel planes as the target data simulations (see Section 3), i.e. $(N_x \times N_y \times N_z) = (700 \times 70 \times 96)$. At the inlet $y^+ = 35$ for the wall-adjacent cell centre. The instantaneous inlet boundary conditions ($\bar{u}$, $\bar{v}$, $\bar{w}$, k and ε) are taken from a precursor flow of fully-developed channel flow at $Re_\tau = 5200$ using IDDES with wall functions based on **KDTree** using hump flow data.

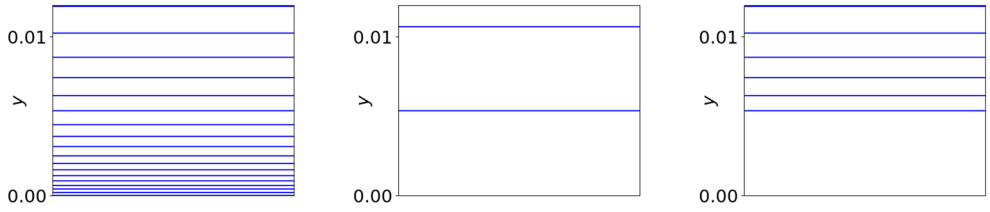
Next, the grid is coarsened by a factor two in both x and z direction compared to the target-data simulations. The domain is the same as in Figure 1(a). This gives a grid with $350 \times 70 \times 48$ cells. Finally, the diffuser angle is reduced to $\alpha = 10°$ and a coarsened grid is used. The contraction region is shorter (less chance of inflow at the outlet), $L_3 = 7$, which gives a grid with $275 \times 70 \times 48$ cells.

Three wall functions are used: **KDTree** using hump flow data (Section 4), **KDTree** using diffuser flow data (Section 4) and Reichardt's wall function (Section 5). They are compared with WR-IDDES. The predicted pressure coefficient, skin friction and two velocity profiles are shown in Figures 10, 11 and 12. Overall, the **KDTree** using the diffuser flow data gives slightly better results than the other two wall functions and the **KDTree** using the diffuser flow data predicts somewhat too large a skin friction in the inlet region for the fine mesh for $\alpha = 15°$ and for $\alpha = 10°$. It may be noted that the skin frictions on the fine mesh (Figure 10(b)) are larger than on the course mesh (Figure 11(b)). The reason is that on the fine mesh, larger, resolved turbulent fluctuations are created by the synthetic inlet fluctuation because the finer mesh is able to resolve more turbulence than the coarse mesh. For example, at $x = -5$ the magnitude of the peak absolute shear stress is 1.15 and 0.80 at the fine and coarse mesh, respectively, for **KDTree** using diffuser flow data.

6.2. Hump flow

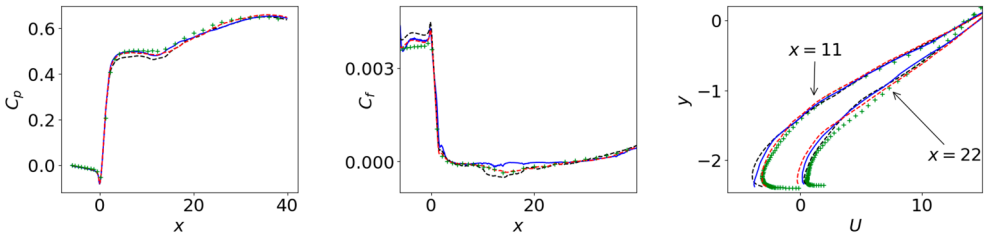
The inlet and outlet boundary conditions are the same as for the WR-IDDES simulations of the hump flow in Section 3 and Figure 5. Periodic boundary conditions are set in the spanwise direction. First, we use the same grid in the x–y plane as in the WR-IDDES. The first 24 cells near the wall are merged into one large cell. This gives a grid with $582 \times 106 \times 64$ cells.

The predictions using the three wall functions are presented in Figures 13 and 14. Overall, the **KDTree** with diffuser flow data shows the best agreement with experiments. The pressure suction peak for all wall functions



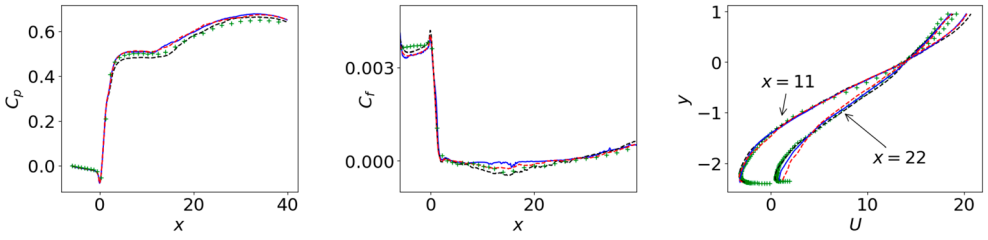
(a) WR-IDDES grid. (b) Wall function grid. (c) New wall function grid.

Figure 9. Different grids. —— : grid lines. (a) WR-IDDES grid. (b) Wall function grid and (c) New wall function grid.



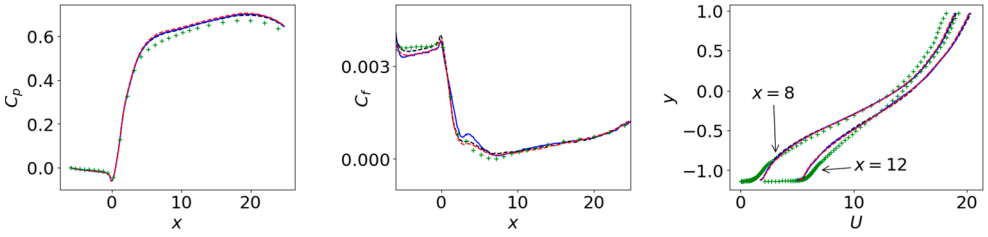
(a) Pressure coefficient. (b) Skin coefficient. (c) Velocity profiles.

Figure 10. Diffuser flow, $\alpha = 15^\circ$. $(N_x \times N_z) = (700 \times 96)$. ——: **KDTree** using hump flow data; ▬ ▬: **KDTree** using diffuser flow data; – –: Reichardt's wall function; +: WR-IDDES ($700 \times 90 \times 96$), see Figure 9(a). (a) Pressure coefficient. (b) Skin coefficient and (c) Velocity profiles.



(a) Pressure coefficient. (b) Skin coefficient. (c) Velocity profiles.

Figure 11. Diffuser flow, $\alpha = 15^\circ$. $(N_x \times N_z) = (350 \times 48)$. ——: **KDTree** using hump flow data; ▬ ▬: **KDTree** using diffuser flow data; – –: Reichardt's wall function; +: WR-IDDES ($700 \times 90 \times 96$), see Figure 9(a). (a) Pressure coefficient. (b) Skin coefficient and (c) Velocity profiles.

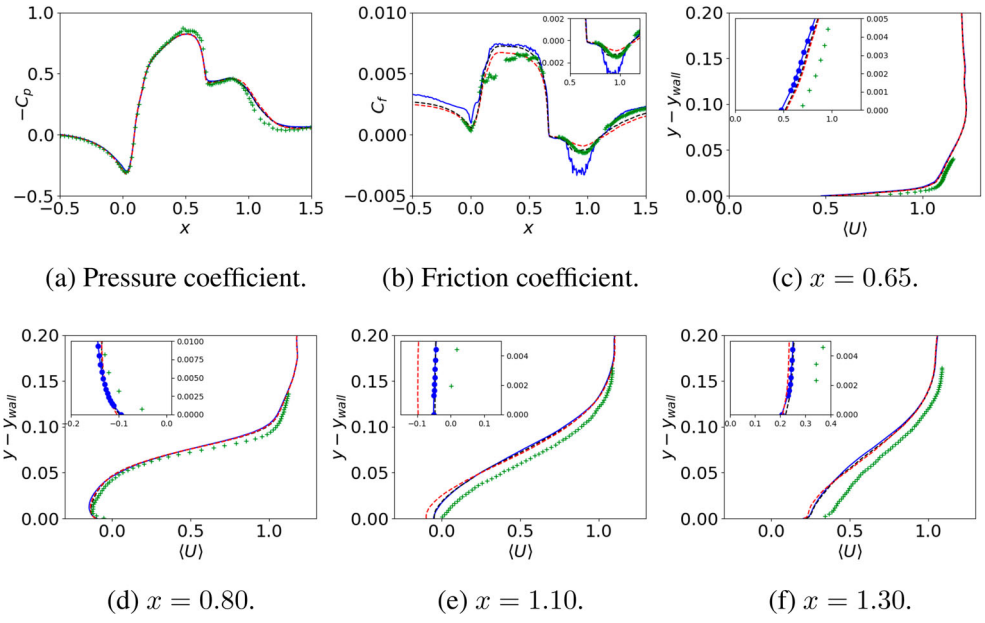


(a) Pressure coefficient. (b) Skin coefficient. (c) Velocity profiles.

Figure 12. Diffuser flow, $\alpha = 10^\circ$. $(N_x \times N_z) = (275 \times 48)$. ——: **KDTree** using hump flow data; ▬ ▬: **KDTree** using diffuser flow data; – –: Reichardt's wall function; +: WR-IDDES ($550 \times 96 \times 90$), see Figure 9(a). (a) Pressure coefficient. (b) Skin coefficient and (c) Velocity profiles.

is 4% too low compared to experiments and WR-IDDES and the velocity in the attached boundary layer at $x = 0.65$ is approximately 2% too low. This is probably connected to the under-predicted velocity profiles at $x = 1.10$ and $x = 1.30$. The skin friction on the upstream part of the hump is over-predicted by all wall functions. As mentioned when discussing Figure 6 above, the turbulence in the experiment experiences a tendency to re-laminarisation which all wall functions fail to capture. The **KDTree** with hump flow data exhibits too strong a backflow when looking at the skin friction whereas the Reichardt's wall function gives too weak a backflow. However, when looking at the velocity profiles and shear stresses, all three wall functions give similar results. Furthermore, it is seen that the predicted shear stresses in the attached boundary layer ($x = 0.65$) are much too large for all three wall functions. When compared with the stressed predicted with WR-IDDES in Figure 7 it is seen that it is the resolved stresses that are too large (the modelled one are of the same magnitude in Figures 14 and 7). It should be that peak of $\langle |u'v'| \rangle$ at $x = 0.65$ in the WR-LES simulations (Figure 7) is even larger than in Figure 14.

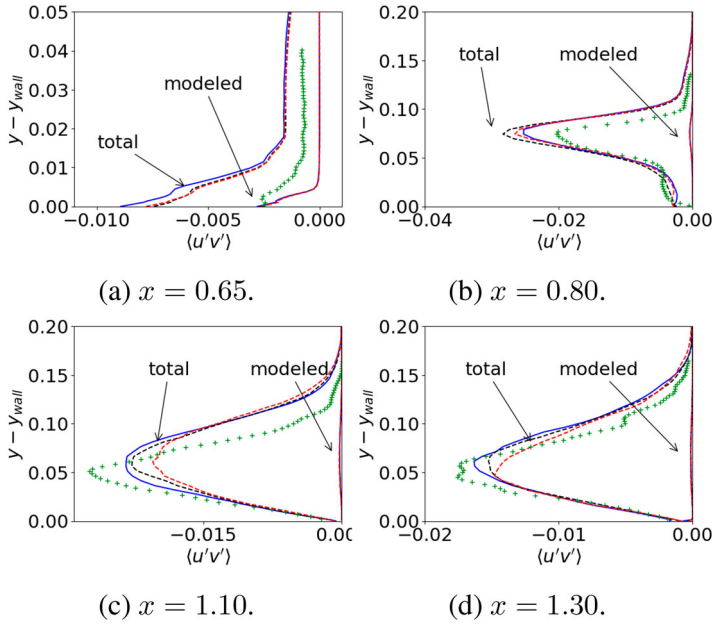
In the next case, the grid is coarsened in both x and z directions, i.e. $291 \times 106 \times 32$. Figures 15 and 16 present the results and the **KDTree** with diffuser flow data actually gives even better agreement than in Figures



(a) Pressure coefficient. (b) Friction coefficient. (c) $x = 0.65$.

(d) $x = 0.80$. (e) $x = 1.10$. (f) $x = 1.30$.

Figure 13. Hump flow. $(N_x \times N_z) = (582 \times 64)$. ——: **KDTree**, hump flow data with $K = 5$; ▬ ▬: **KDTree**, diffuser flow data; – –: Reichardt's wall function; •: location of cell centres; +: experiments [32]. (a) Pressure coefficient. (b) Friction coefficient. (c) $x = 0.65$. (d) $x = 0.80$. (e) $x = 1.10$ and (f) $x = 1.30$.

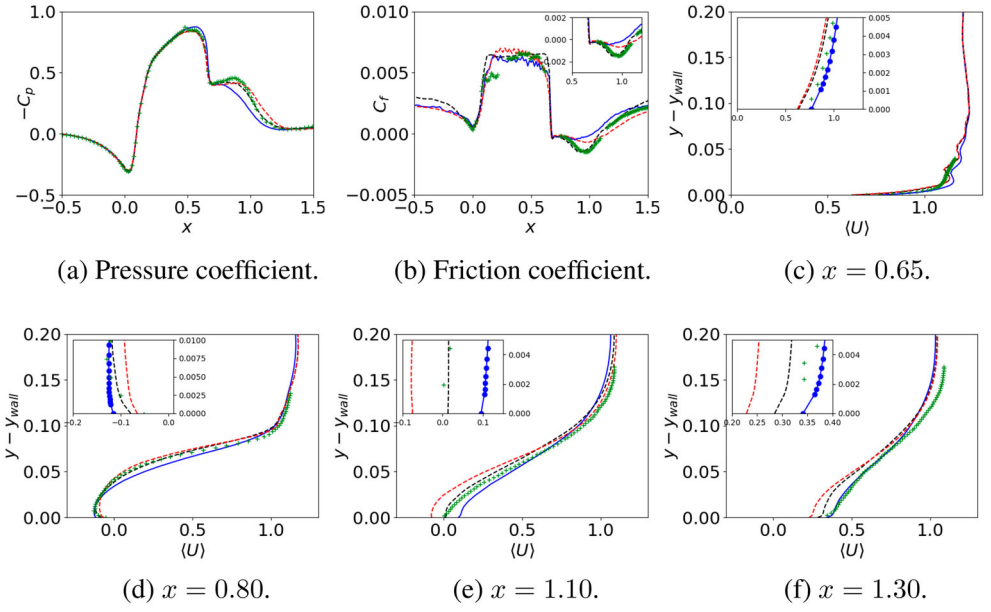


(a) $x = 0.65$. (b) $x = 0.80$.

(c) $x = 1.10$. (d) $x = 1.30$.

Figure 14. Hump flow. $(N_x \times N_z) = (582 \times 64)$. Total turbulent shear stress. ——: **KDTree**, hump flow data with $K = 5$; ▬ ▬: **KDTree**, diffuser flow data; – –: Reichardt's wall function; +: experiments [32]. (a) $x = 0.65$. (b) $x = 0.80$. (c) $x = 1.10$ and (d) $x = 1.30$.

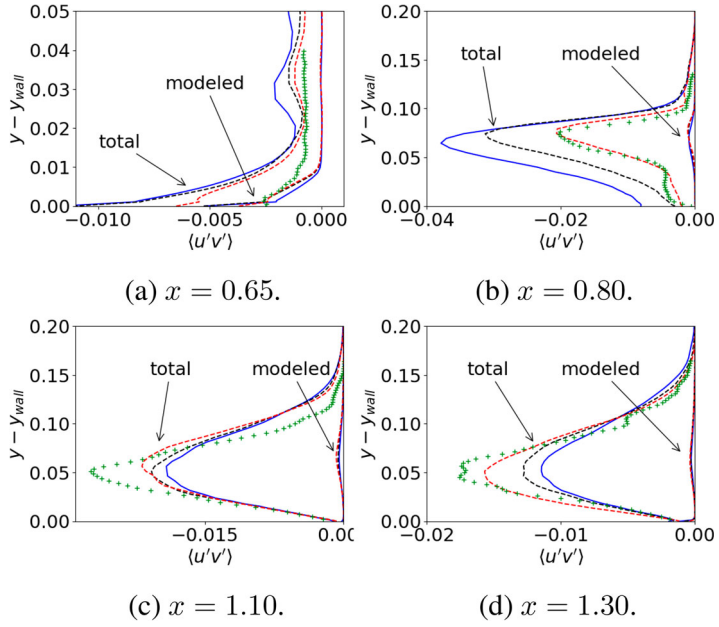
13 and 14. This is a 'lucky' coincidence due to a combination of modelled Reynolds stresses and discretisation errors. Here $K = 1$ for the **KDTree** with hump flow data and it can be seen that the predicted skin friction is rather poorly predicted (very similar to that predicted on the grid $582 \times 106 \times 64$ with $K = 1$, not shown). Although the **KDTree** with diffuser flow data predicts gives better agreement than the fine mesh, the total shear stresses are now even more over-predicted at $x = 0.65$. Furthermore, the velocity profiles at $x = 0.65$ exhibit small oscillations which probably stem from a combination of low turbulent viscosity (the modelled



(a) Pressure coefficient. (b) Friction coefficient. (c) $x = 0.65$.

(d) $x = 0.80$. (e) $x = 1.10$. (f) $x = 1.30$.

Figure 15. Hump flow. $(N_x \times N_z) = (291 \times 32)$. ——: **KDTree**, hump flow data; ▬ ▬: **KDTree**, diffuser flow data; – –: Reichardt's wall function; •: location of cell centres; +: experiments [32]. (a) Pressure coefficient. (b) Friction coefficient. (c) $x = 0.65$. (d) $x = 0.80$. (e) $x = 1.10$ and (f) $x = 1.30$.



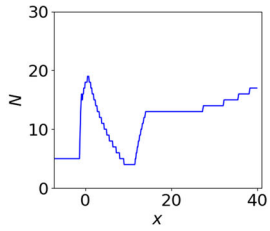
(a) $x = 0.65$. (b) $x = 0.80$.

(c) $x = 1.10$. (d) $x = 1.30$.

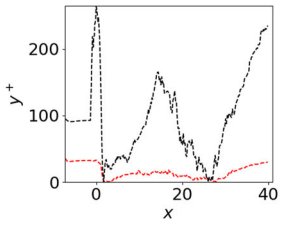
Figure 16. Hump flow. $(N_x \times N_z) = (291 \times 32)$. Total turbulent shear stress. ——: **KDTree**, hump flow data; ▬ ▬: **KDTree**, diffuser flow data; – –: Reichardt's wall function; +: experiments [32]. (a) $x = 0.65$. (b) $x = 0.80$. (c) $x = 1.10$ and (d) $x = 1.30$.

shear stress is small where the oscillations appear, see Figure 16), the central differencing scheme and a coarse grid.

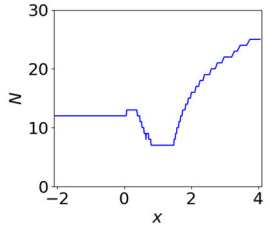
It is interesting to find the location of the URANS/LES interface. It is defined as the cell where f_d falls below one, see Equation (7). Figure 17(a,c) show how many cells are located in the URANS region. Figure 17(b,d) present the y^+ values of the URANS/LES interface and it is seen that it is located in the ranges $0.06 \leq y^+ \leq 265$ and $1 \leq y^+ \leq 1685$ for the diffuser flow and hump flow, respectively. The y^+ values of the wall-adjacent cells are also presented in Figure 17(b,d). They are in the range $0.06 \leq y^+ \leq 35$ and $0.25 \leq y^+ \leq 50$ for the diffuser flow and hump flow, respectively.



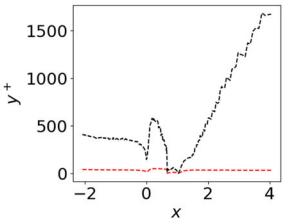
(a) Diffuser flow. Number of cells in URANS region



(b) Diffuser flow. – – : y^+ of wall-adjacent cells. ▬ ▬ : y^+ of URANS-LES interface.



(c) Hump flow. Number of cells in URANS region



(d) Hump flow.– – : y^+ of wall-adjacent cells. ▬ ▬ : y^+ of URANS-LES interface.

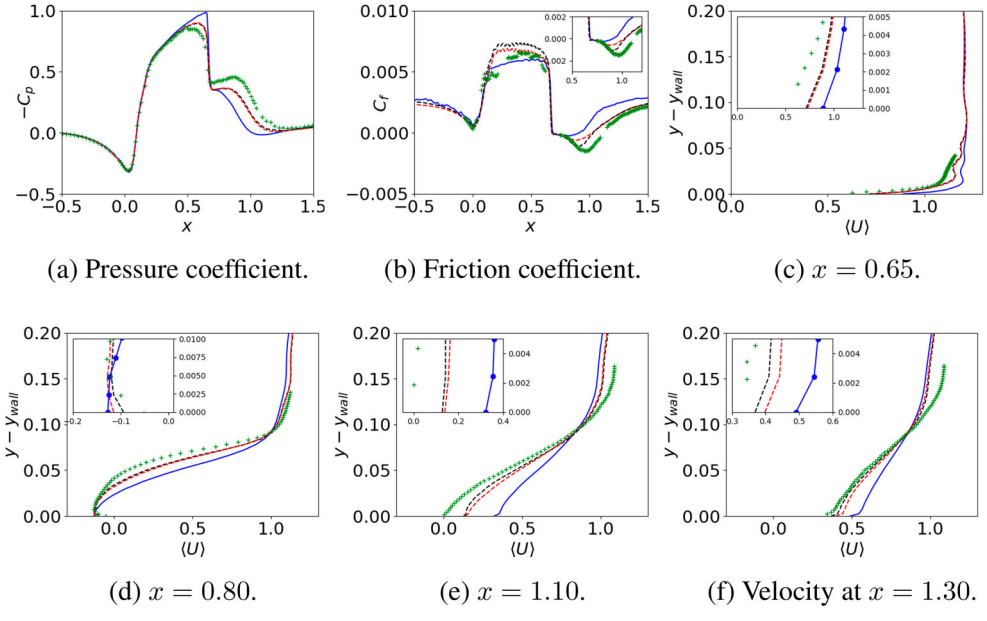
Figure 17. Location of URANS-LES interface. Diffuser flow, $\alpha = 15^\circ$. 350 × 70 × 48 and Hump flow, 291 × 106 × 32. (a) Diffuser flow. Number of cells in URANS region. (b) Diffuser flow. – – : y^+ of wall-adjacent cells. ▬ ▬ : y^+ of URANS-LES interface. (c) Hump flow. Number of cells in URANS region and (d) Hump flow.– – : y^+ of wall-adjacent cells. ▬ ▬ : y^+ of URANS-LES interface.

In the wall function simulations presented so far, we have used the new wall function grids, see Figure 9(c). In the next case, we will use a standard wall function grid, see Figure 9(b). The wall-adjacent cells ($j = 0$) have the same size as in the new wall function grid, and further away from the wall ($j > 0$) a geometrical stretching of 1.04 is used but with a limit $\Delta y < \Delta y_{\max} \equiv 0.014$ ($\Delta y_{\max} = 0.015$ in the new wall function grid). Figure 18 presents the results and it can be seen that the agreement with experiments is much worse than with the new wall function grid.

6.3. Flat-plate boundary layer flow

The next test case is a flat-plate boundary layer. It is a classic test case for verifying RANS turbulence models, for example the Reynolds stress model [45] and the SST model [46]. It has also been used for evaluation hybrid LES/RANS models [47] and embedded RANS-LES [39, 48]. The Reynolds number at the inlet is $Re_\theta = 2550$. The inlet and outlet boundary conditions are implemented in the same way as in the hump flow WR-IDDES simulations in Section 3 and Figure 5 (the mean profiles are taken from a pre-cursor 2D RANS at $Re_\theta = 2550$). Periodic boundary conditions are set in the spanwise direction. The grid has 550 × 85 × 64 cells (x, y, z). All length are non-dimensionalised by the inlet boundary-layer thickness is δ_{in} (unless viscous units are used). The domain size is 7.9 × 5.7 × 4.0. The first cell is obtained by merging 17 cells in the WR-IDDES grid. From the second wall-adjacent cell, the grid is stretched by 10% for $y < 0.64$ and $y > 2.5$ but Δy is not allowed to exceed 0.125. In the streamwise direction $\Delta x_{in} = 0.1$ and a 0.1% stretching is used and the wall-adjacent cell centre at the inlet is located at $y^+ = 14$.

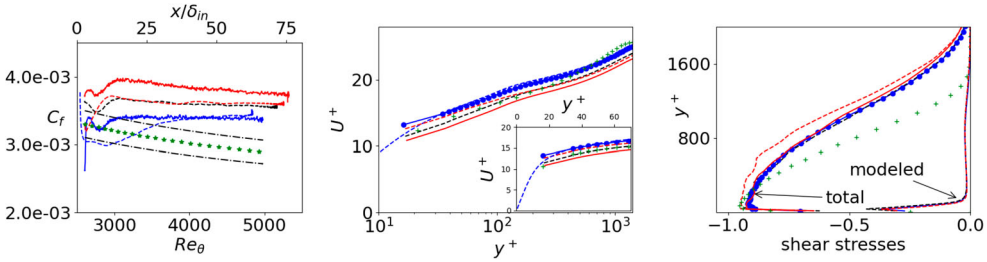
Figure 19 presents the skin friction, the velocity profiles and the turbulent shear stresses. It is found that the **KDTree** using hump flow data performs fairly well; it over-predicts C_f by 12% at $Re_\theta = 4000$ (similar to the WR-IDDES and better than the Reichardt's wall function) whereas the **KDTree** using diffuser flow over-predicts the skin friction**KDTree** by 18%. The skin friction predicted with the **KDTree** using diffuser flow data with $K = 1$ is also shown in Figure 19 and it is seen that C_f is over-predicted by 25%. At $Re_\theta = 4500$,



(a) Pressure coefficient. (b) Friction coefficient. (c) $x = 0.65$.

(d) $x = 0.80$. (e) $x = 1.10$. (f) Velocity at $x = 1.30$.

Figure 18. Hump flow. $(N_x \times N_z) = (291 \times 32)$. $N_y = 80$. Standard wall-function mesh, see Figure 9(b). ——: **KDTree**, hump flow data; ▬ ▬: **KDTree**, diffuser flow data; – –: Reichardt's wall function; •: location of cell centres; +: experiments [32]. (a) Pressure coefficient. (b) Friction coefficient. (c) $x = 0.65$. (d) $x = 0.80$. (e) $x = 1.10$ and (f) Velocity at $x = 1.30$.



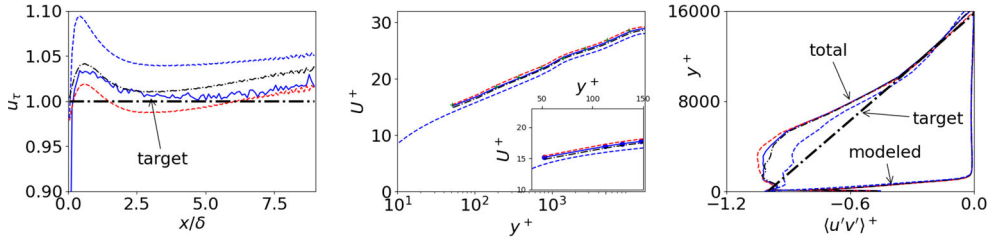
(a) Friction coefficient. (b) Mean velocity. (c) Shear stresses.

Figure 19. Boundary layer flow. u_τ is computed by using U^+ and y^+ at the 3rd cell. Velocity and shear stresses are shown at $Re_\theta = 4000$. ——: **KDTree**, hump flow data ▬ ▬: **KDTree**, diffuser flow data; ——: **KDTree**, diffuser flow data, $K = 1$; – –: Reichardt's wall function; •: location of cell centres; - - -: WR-IDDES; *: $C_f = 2(1/0.384 \ln(Re_\theta) + 4.127)^{-2}$; ▬ · ▬: ±6%; +: DNS [44] (every 10th cell). (a) Friction coefficient. (b) Mean velocity and (c) Shear stresses.

for example, the predicted skin friction is 0.0038, 0.0037, 0.0036 and 0.0035 for $K = 1$, $K = 2$, $K = 5$ and $K = 10$, respectively. All three wall functions predict fairly good velocities and shear stress profiles.

6.4. Channel flow

The last test case is channel flow. This test case is often used for evaluating inlet boundary conditions in LES [49, 50] and hybrid RANS/LES [51] as well as embedded RANS/LES [52]. The domain size is $9\delta \times 2\delta \times 1.6\delta$. The Reynolds number at the inlet is $Re_\tau = 16{,}000$ based on the friction velocity and the half-channel width, δ. All length are scaled with δ (except y^+). The inlet is located at $x = 0$ and the outlet is at $x = 9$. The inlet and outlet boundary conditions are implemented in the same way as in the hump flow WR-IDDES simulations in Section 3 and Figure 5 (the mean profiles are taken from a pre-cursor 1D RANS at $Re_\tau = 16{,}000$). Periodic boundary conditions are set in the spanwise direction. The walls are located at $y = 0$ and $y = 2$. The grid has $96 \times 77 \times 32$ cells (x, y, z). Periodic boundary conditions are used for all variables in the spanwise (z) direction. The under-lying WR-grid is stretched by 14% from the walls located at $y = 0$ and $y = 2$. A wall-adjacent cell in the wall-function grid is formed by merging 23 near-wall cells in the WR-grid. The wall-adjacent cell centre at the inlet is located at $y^+ = 45$.



(a) Friction velocity. (b) Mean velocity. (c) Shear stress.

Figure 20. Channel flow. $Re_\tau = 16{,}000$. Velocity and shear stress are shown at $x/\delta = 6$. ——— : **KDTree**, hump flow data ▬ ▬ : **KDTree**, diffuser flow data; - - - : WR-IDDES, see Figure 9(a); —— : **KDTree**, hump flow data, $K = 1$; ▬ ▬ : Reichardt's wall function; ●: location of cell centres; +: Reichardt's law, Equation (18). (a) Friction velocity. (b)Mean velocity and (c) Shear stress.

Figure 20 shows the predicted friction velocity, the mean velocity and the turbulent shear stresses. The Reichardt's wall function and the **KDTree** using hump flow data predict the skin friction very well. The **KDTree** using diffuser flow data and Reichardt's wall function predict a skin friction which is approximately 2% and 4% too large, respectively, which also is good. On the other hand, the WR-IDDES yields a skin friction which is 5% too high. The target $u_\tau = 1$ is also shown in Figure 20(a). The reason that no simulation reaches the target is that the time-averaged streamwise velocity does not perfectly match the inlet RANS velocity. All three wall functions give too large a total shear stress which shows that the flow is not fully developed at $x = 6$ (Figure 20(b)).

7. Conclusions

A new wall function based on binary search trees, using Python' **KDTree** has been presented. Two different target database have been used, namely the flow in a diffuser (opening angle, $\alpha = 15^\circ$) and the flow over a hump. Time-averaged U^+ and y^+ are used as target quantities.

The reported resolution requirement in the literature for LES/DES of a boundary layer using wall functions varies between Re_L and $Re_L^{1.14}$. In the present work it is shown that wall-resolved IDDES (wall-adjacent cell at $y^+ < 1$) requires an additional number of cells which vary with the friction Reynolds number as $\ln(Re_\tau)$ and $0.2\ln(Re_\tau)$ for wall-normal grid stretching of 1.04 and 1.15, respectively.

Five test cases have been used: the diffuser flow, $\alpha = 15^\circ$ and $\alpha = 10^\circ$, the hump flow, flat-plate boundary layer flow and channel flow. The new **KDTree** wall function using the diffuser flow database is found to perform well, better than when using the hump flow database and better than the Reichardt's wall function wall function. The reason why the diffuser flow database performs better is probably because this is a simpler flow which makes it easier for the **KDTree** search routine to find suitable y^+ and U^+. The new **KDTree** wall function performs better that WR-IDDES for both flat-plate boundary layer and channel flow. For the hump flow, however, both **KDTree** wall functions perform slightly worse than WR-IDDES. For example, the suction pressure peak is 4% too low and the velocity in the attached boundary layer upstream the re-circulation region is approximately 2% too low. This results in somewhat under-predicted velocities in the recovery region downstream the re-circulation region.

A new strategy is used for creating the wall-normal distribution of the grid which was found to be very beneficial. The suction peak of the pressure coefficient in the hump flow using the standard wall function grid is over predicted by 18% (Figure 18) whereas the peak is well predicted using the new grid strategy (Figure 15).

The **KDTree** using hump flow data over-predicts the skin friction of the boundary flow by 12% at $Re_\theta = 4000$ (same as WR-IDDES). Using diffuser flow data in **KDTree** gives an over-prediction of 18% (same as Reichardt's wall function). However, for this flow the **KDTree** using diffuser flow data is sensitive to how many nearest neighbours are used in **KDTree**. Using the nearest neighbour, $K = 1$ (instead of the baseline value of $K = 5$), gives a C_f which is 25% too large.

It should be mentioned that databases with instantaneous data were also investigated, see end of Section 4 (no results were presented). It was found that the instantaneous databases gave slightly worse results than the time-averaged ones.

Disclosure statement

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

Funding

This study was financed by Vinnova, NFFP, Grant No. 2023-01569, Strategic research project on Chalmers on hydro- and aerodynamics and Chalmers Transport Area of Advance, Grant No. C 2023-0125-19.

References

[1] Spalart PR, Jou W-H, Strelets M, et al. Comments on the feasability of LES for wings and on a hybrid RANS/LES approach. In: Liu C, Liu Z, editors. Advances in LES/DNS, first Int. conf. on DNS/LES, Louisiana Tech University. Columbus: Greyden Press; 1997.
[2] Spalart PR. Strategies for turbulence modelling and simulations. Int J Heat Fluid Flow. 2000;21:252–263. doi: 10.1016/S0142-727X(00)00007-2
[3] Shur ML, Spalart PR, Kh. Strelets M, et al. A hybrid RANS-LES approach with delayed-DES and wall-modelled LES capabilities. Int J Heat Fluid Flow. 2008;29:1638–1649. doi: 10.1016/j.ijheatfluidflow.2008.07.001
[4] Davidson L, Peng S-H. Hybrid LES-RANS: a one-equation SGS model combined with a $k-\omega$ for predicting recirculating flows. Int J Numer Methods Fluids. 2003;43(9):1003–1018. doi: 10.1002/fld.v43:9
[5] Temmerman L, Hadiabdi M, Leschziner MA, et al. A hybrid two-layer URANS-LES approach for large eddy simulation at high Reynolds numbers. Int J Heat Fluid Flow. 2005;26(2):173–190. doi: 10.1016/j.ijheatfluidflow.2004.07.006
[6] Menter FR, Egorov Y. The scale adaptive simulation method for unsteady turbulent flow predictions. Part 1: theory and description. Flow Turbul Combust. 2010;85:113–138. doi: 10.1007/s10494-010-9264-5
[7] Egorov Y, Menter FR, Lechner R, et al. The scale adaptive simulation method for unsteady flow predictions. Part 2: application to complex flows. Flow Turbul Combust. 2010;85:139–165. doi: 10.1007/s10494-010-9265-4
[8] Fröhlich J, von Terzi D. Hybrid LES/RANS methods for the simulation of turbulent flows. Prog Aerosp. 2008;44(5):349–377. doi: 10.1016/j.paerosci.2008.05.001
[9] Spalart P, Deck S, Shur M, et al. A new version of detached-eddy simulation, resistant to ambiguous grid densities. Theor Comput Fluid Dyn. 2006;20:181–195. ISSN 0935-4964. doi: 10.1007/s00162-006-0015-0
[10] Menter FR, Kuntz M. Adaption of eddy-viscosity turbulence models to unsteady separated flows behind vehicles. In: McCallen R, Browand F, Ross J, editors. The aerodynamics of heavy vehicles: trucks, buses, and trains, volume 19 of lecture notes in applied and computational mechanics. Berlin, Heidelberg: Springer Verlag; 2004.
[11] Strelets M. Detached eddy simulation of massively separated flows.In: AIAA paper 2001–0879, Reno, NV; 2001.
[12] Kawai S, Larsson J. Wall-modeling in large eddy simulation: length scales, grid resolution, and accuracy. Phys Fluids. 2012;24:Article ID 015105. doi: 10.1063/1.3678331
[13] Deardorff JW. A numerical study of the three-dimensional turbulent channel flow at large Reynolds numbers. J Fluid Mech. 1970;41:453–480. doi: 10.1017/S0022112070000691
[14] Larsson J, Kawai S, Bodart J, et al. Large eddy simulation with modeled wall-stress: recent progress and future directions. Mech Eng Rev. 2016;3(1):Article ID 15-00418. doi: 10.1299/mer.15-00418
[15] Choi H, Moin P. Grid-point requirements for large eddy simulation: chapman's estimates revisited. Phys Fluids. 2012 Jan;24(1):Article ID 011702. ISSN 1070-6631. doi: 10.1063/1.3676783
[16] Yang XIA, Griffin KP. Grid-point and time-step requirements for direct numerical simulation and large-eddy simulation. Phys Fluids. 2021 Jan;33(1):Article ID 015108. ISSN 1070-6631. doi: 10.1063/5.0036515
[17] Agrawal R, Bose ST, Moin P. Reynolds-number-dependence of length scales governing turbulent-flow separation in wall-modeled large eddy simulation. AIAA J. 2024;62(10):3686–3699. doi: 10.2514/1.J063909
[18] Herr M, Probst A. Efficient modelling of near-wall turbulence in hybrid RANS-LES simulations. In: Dillmann A, Heller G, Krämer E, Wagner C, editors. New results in numerical and experimental fluid mechanics XIII. Cham: Springer International Publishing; 2021. p. 615–624. ISBN 978-3-030-79561-0. doi: 10.1007/978-3-030-79561-0_58
[19] Davidson L. Using machine learning for formulating new wall functions for detached eddy simulation. In: 14th international ERCOFTAC symposium on engineering turbulence modelling and measurements (ETMM14), Barcelona/Digital, Spain 6–8 September; 2023.Available from: http://www.tfd.chalmers.se/lada/postscript_files/paper-davidson-etmm14.pdf.
[20] Zhou Z, He G, Yang X. Wall model based on neural networks for LES of turbulent flows over periodic hills. Phys Rev Fluids. 2021 May;6:Article ID 054610. doi: 10.1103/PhysRevFluids.6.054610
[21] Tieghi L, Corsini A, Delibra G, et al. A machine-learnt wall function for rotating diffusers. Volume 1: Aircraft Engine; Fans and Blowers of Turbo Expo: Power for Land, Sea, and Air; 2020 Sept. doi: 10.1115/GT2020-15353
[22] Ling J, Barone MF, Davis W, et al. Development of machine learning models for turbulent wall pressure fluctuations. In: 55th AIAA aerospace sciences meeting; 2017. doi: 10.2514/6.2017-0755
[23] Dominique J, Van den Berghe J, Schram C, et al. Artificial neural networks modeling of wall pressure spectra beneath turbulent boundary layers. Phys Fluids. 2022;34(3):Article ID 035119. doi: 10.1063/5.0083241

[24] Bae HJ, Koumoutsakos P. Scientific multi-agent reinforcement learning for wall-models of turbulent flows. Nat Commun. 2022;13(1):Article ID 1443. doi: 10.1038/s41467-022-28957-7
[25] Gritskevich MS, Garbaruk AV, Menter FR. A comprehensive study of improved delayed detached eddy simulation with wall functions. Flow Turbul Combust. 2017;98:461–479. doi: 10.1007/s10494-016-9761-2
[26] Romanelli M, Beneddine S, Mary I, et al. Data-driven wall models for Reynolds averaged Navier–Stokes simulations. Int J Heat Fluid Flow. 2023;99:Article ID 109097. ISSN 0142-727X. doi: 10.1016/j.ijheatfluidflow.2022.109097
[27] Davidson L. pyCALC-LES: a python code for DNS, LES and hybrid LES-RANS. Gothenburg: Division of Fluid Dynamics, Dept. of Mechanics and Maritime Sciences, Chalmers University of Technology; 2021. Available from: http://www.tfd.chalmers.se/lada/postscript_files/py-calc-les.pdf.
[28] Abe K, Kondoh T, Nagano Y. A new turbulence model for predicting fluid flow and heat transfer in separating and reattaching flows – 1. Flow field calculations. Int J Heat Mass Transf. 1994;37(1):139–151. doi: 10.1016/0017-9310(94)90168-6
[29] Shur M, Spalart PR, Strelets MK, et al. Synthetic turbulence generators for RANS-LES interfaces in zonal simulations of aerodynamic and aeroacoustic problems. Flow Turbul Combust. 2014;93:63–92. doi: 10.1007/s10494-014-9534-8
[30] Uzun A, Malik MR. Large-eddy simulation of flow over a wall-mounted hump with separation and reattachment. AIAA J. 2018;56(2):715–730. doi: 10.2514/1.J056397
[31] Azun A. LES: compressible 2-D NASA wall-mounted hump; 2017.Available from: https://turbmodels.larc.nasa.gov/Other_LES_Data/nasa_hump_uzun_2017.html.
[32] Greenblatt D, Paschal KB, Yao C-S, et al. A separation control CFD validation test case. Part 1: baseline & steady suction.In: AIAA-2004-2220; 2004. doi: 10.2514/6.2004-2220
[33] Naughton JM, Viken SA, Greenblatt DD. Skin-friction measurements on the NASA hump model. AIAA J. 2006;44(6):1255–1265. doi: 10.2514/1.14192
[34] NASA. Exp: CFDVAL2004 data from workshop on CFD validation of synthetic jets and turbulent separation control; 2004.Available from: https://turbmodels.larc.nasa.gov/nasa40percent.html.
[35] NASA. NASA's 40% challenge and CFD prediction error assessment workshop; 2018.Available from: https://turbmodels.larc.nasa.gov/nasa40percent.html.
[36] Schwamborn D, Strelets M. ATAAC – an EU-project dedicated to hybrid RANS/LES methods. Volume 117, 2012 July. ISBN 978-3-642-31817-7. doi: 10.1007/978-3-642-31818-4-5
[37] Garbaruk A, Guseva E, Shur M, et al. 2D wall-mounted hump. In: Mockett C, Haase W, Schwamborn D, editors. Go4Hybrid: grey area mitigation for hybrid RANS-LES methods, volume 134 of notes on numerical fluid mechanics and multidisciplinary design. Cham: Springer Verlag; 2018. p. 173–188. doi: 10.1007/978-3-319-52995-0
[38] Vreman AW. An eddy-viscosity subgrid-scale model for turbulent shear flow: algebraic theory and applications. Phys Fluids. 2004 Oct;16(10):3670–3681. ISSN 1070-6631. doi: 10.1063/1.1785131
[39] Davidson L. Zonal PANS: evaluation of different treatments of the RANS-LES interface. J Turbul. 2016;17(3):274–307. doi: 10.1080/14685248.2015.1093637
[40] Friess C, Davidson L. A formulation of PANS able to mimic IDDES. Int J Heat Fluid Flow. 2020;86:Article ID 108666. doi: 10.1016/j.ijheatfluidflow.2020.108666
[41] Friess C, Manceau R, Gatski TB. Toward an equivalence criterion for hybrid RANS/LES methods. Comput Fluids. 2015;122:233–246. doi: 10.1016/j.compfluid.2015.08.010
[42] Paik J, Sotiropoulos F. DES of turbulent flow over wall-mounted obstacles using wall functions. KSCE J Civ Eng. 2012;16:189–196. doi: 10.1007/s12205-012-0001-6
[43] Davidson L. Python scripts and databases for a ML wall-function. Gothenburg: Dept. of Mechanics and Maritime Sciences, Chalmers University of Technology; 2024. Available from: https://www.tfd.chalmers.se/lada/wall_function_ML_kdtree.html.
[44] Sillero JA, Jimenez J, Moser RD. One-point statistics for turbulent wall-bounded flows at Reynolds numbers up to $\delta^+ \simeq 2000$. Phys Fluids. 2014;25:Article ID 105102. doi: 10.1063/1.4823831
[45] Launder BE, Reece GJ, Rodi W. Progress in the development of a Reynolds-stress turbulence closure. J Fluid Mech. 1975;68(3):537–566. doi: 10.1017/S0022112075001814
[46] Menter FR. Two-equation eddy-viscosity turbulence models for engineering applications. AIAA J. 1994;32:1598–1605. doi: 10.2514/3.12149
[47] Deck S, Renard N, Laraufie R, et al. Zonal detached eddy simulation (ZDES) of a spatially eveloping flat plate turbulent boundary layer over the Reynolds number range $3150 \leq Re\_\theta \leq 14{,}000$. Phys Fluids A. 2014;26:Article ID 025116.
[48] Shur ML, Spalart PR, Kh. Strelets M, et al. A rapid and accurate switch from RANS to LES in boundary layers using an overlap region. Flow Turbul Combust. 2011;86(2):179–206. doi: 10.1007/s10494-010-9309-9
[49] Benhamadouche S, Jarrin N, Addad Y, et al. Synthetic turbulent inflow conditions based on a vortex method for large-eddy simulation. Prog Comput Fluid Dyn. 2001 Jan;6:50–56. doi: 10.1504/PCFD.2006.009482
[50] Mathey F, Cokljat D, Bertoglio JP, et al. Assessment of the vortex method for large eddy simulation inlet conditions. Prog Comput Fluid Dyn. 2006;6(1–3):58–67. doi: 10.1504/PCFD.2006.009483
[51] Davidson L, Billson M. Hybrid LES/RANS using synthesized turbulent fluctuations for forcing in the interface region. Int J Heat Fluid Flow. 2006;27(6):1028–1042. doi: 10.1016/j.ijheatfluidflow.2006.02.025

[52] Holgate J, Skillen A, Craft T, et al. A review of embedded large eddy simulation for internal flows. Arch Comput Methods Eng. 2019;26:865–882. doi: 10.1007/s11831-018-9272-5

Appendices

Appendix 1. Wall-normal stretched grids

We assume that the grid in the wall-normal direction is stretched the first m cells, i.e.

$$\Delta y + \gamma\,\Delta y + \gamma^2\Delta y + \cdots + \gamma^m\Delta y \tag{A1}$$

Setting $y^+ \equiv u_\tau\,\Delta y/\nu = 1$ for the wall-adjacent cell and limit the largest wall-normal cell to δ/N_0 (δ is the boundary-layer thickness) then the last term in Equation (A1) can be written as

$$\begin{aligned} \nu\gamma^m/u_\tau = \frac{\delta}{N_0} &\Rightarrow m\ln(\gamma) = \ln(u_\tau\delta/(N_0\nu)) \\ &\Rightarrow m = \ln(Re_\tau/N_0)/\ln(\gamma) \end{aligned} \tag{A2}$$

where Re_τ is the friction Reynolds number, $Re_\tau = u_\tau\delta/\nu$. We define the region, D_γ, in which the grid is stretched so that

$$\begin{aligned} D_\gamma &= \frac{\nu}{u_\tau}\left(1 + \gamma^1 + \gamma^2 + \cdots + \gamma^m\right) \\ &= \frac{\nu}{u_\tau}\sum_{k=0}^{m}\gamma^k \Rightarrow D_\gamma = \frac{\nu}{u_\tau}\frac{1-\gamma^m}{1-\gamma} \end{aligned} \tag{A3}$$

Hence, the wall-normal grid of the boundary layer is formed by m stretched cells which cover the region D_γ and $N_y - m$ cells of constant size δ/N_0 which cover the region $\delta - D_\gamma$ where N_y is the total number of wall-normal cells. For which grid stretching, γ, does D_γ extend all the way to $y = \delta$? By solving Equations (A2) and (A3) numerically, we can find the relation between γ and N_0. We set $D_\gamma = \delta$ in Equation (A3) (recall that $\Delta y_{\max} = \delta/N_0$, see Equation (A2))

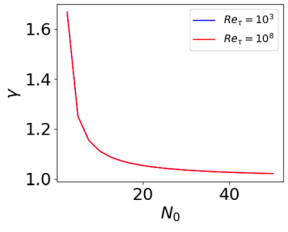
$$Re_\tau(1-\gamma) = 1 - \gamma^m \tag{A4}$$

and insert Equation (A2)

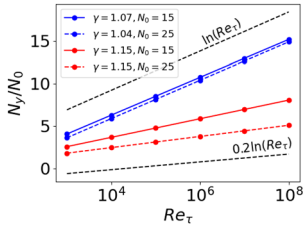
$$Re_\tau(1-\gamma) = 1 - \gamma^{\ln(Re_\tau/N_0)/\ln(\gamma)} \tag{A5}$$

This equation is conveniently solved using the Newton–Raphson method, see Listing 1 in Appendix 2.

Figure A1(a) shows the solution of Equation (A5) for different Reynolds numbers. As can be seen, γ vs. N_0 is virtually independent of Reynolds number. This can be understood by looking at Equation (A4). A large change in Re_τ gives only a tiny change in m (recall that $50 \lesssim m \lesssim 200$). The ratio N_y/N_0 vs. Reynolds number is in Figure A1(b) shown for two combinations of (γ, N_0) that satisfy Equation (A5), i.e. (1.07, 15) and (1.04, 25) which are shown by blue lines; recall that when Equation (A5) is satisfied $\Delta y < \delta/N_0$ for all cells and $m = N_y$. However, for the other two grids in Figure A1(b) (red lines), there are a number of cells of constant size, δ/N_0, outside the region with stretched cells. At $Re_\tau = 10^8$, for example, (N_y, m) are equal to (120, 113) and (127, 109) for $N_0 = 15$ and $N_0 = 25$, respectively.



(a) Grid stretching versus N_0.



(b) Ratio of number of cells in stretched grids to that in WF grids. Black dashed lines show upper $(\ln(Re_\tau))$ and lower trend $(0.2\ln(Re_\tau))$ versus Reynolds number

Figure A1. Comparison of stretched grids and wall-function grids. (a) Grid stretching vs. N_0 and (b) Ratio of number of cells in stretched grids to that in WF grids. Black dashed lines show upper $(\ln(Re_\tau))$ and lower trend $(0.2\ln(Re_\tau))$ vs. Reynolds number.

Appendix 2. Python script for comparing stretched and wall-function grids

```
import numpy as np
import sys
import matplotlib.pyplot as plt
from scipy.optimize import fsolve,root,newton
plt.rcParams.update({'font.size': 22})
plt.close('all')
plt.interactive(True)
def solve_gamma(gam,delta,dx):
    return  (1-gam**(np.log(Re/N_0)/np.log(gam)+1))\
              - Re*(1-gam)
# solve for n_N0 different N_0
n_N0 = 20
# solve for n_re different re
n_re = 6
# set lowest Reynolds number
Re = 1e3
gamma_vector = np.zeros((n_re,n_N0))
Re_vector = np.zeros((n_re))
N_0_vector = np.zeros((n_N0))
m_vector = np.zeros((n_re,n_N0))
dy_vector = np.zeros((n_re,n_N0))
# loop over Re
for r in range(0,n_re):
# first N_0
   N_0 = 2.5
   Re_vector[r] = Re
# loop over N_0
   for n in range(0,n_N0):
# intial gamma value into the solver
      gamma = 1.01
# call the Newton-Raphson solver
      gamma = newton(solve_gamma,x0=gamma,\
                args=(Re,N_0))
      N_0_vector[n] = N_0
# compute m
      m_vector[r,n] = np.log(Re/N_0)/np.log(gamma)
# largest dy
      dy_vector[r,n] = gamma**m_vector[r,n]
      gamma_vector[r,n] = gamma
# next N_0
      N_0 = N_0 +2.5
# next Re
   Re = Re*10
######################## plot gamma vs N_0
fig1,ax1 = plt.subplots()
plt.subplots_adjust(left=0.25,bottom=0.20)
# smallest Reynolds number
plt.plot(N_0_vector,gamma_vector[0,:],'b-')
# largest Reynolds number
plt.plot(N_0_vector,gamma_vector[-1,:],'r-')
plt.xlabel(r"$N_0$")
plt.ylabel(r"$\gamma$")
plt.savefig('gamma-vs-N0-all-N0-and-re.png')
```

Listing 1. Python script for solving Equation (24) for different N_0 and Re_τ

Appendix 3. Python script for KDTree wall function

```
def fix_k():
    from sklearn.preprocessing import MinMaxScaler
```

```
    from scipy.spatial import KDTree                                    3
    global X, tree, yplus_target                                        4
    if iter == 0 and itstep == 0:                                       5
#at t start-up: load target data                                        6
        data = xp.loadtxt('x-yplus-uplus-diffuser.txt')                 7
        x_target = data[:,0]                                            8
        yplus_target = abs(data[:,1])                                   9
        uplus_target = data[:,2]                                        10
        uplus_target = uplus_target.reshape(-1,1)                       11
        yplus_target = yplus_target.reshape(-1,1)                       12
# use MinMax scaler                                                     13
        scaler_yplus = MinMaxScaler()                                   14
        scaler_uplus = MinMaxScaler()                                   15
# store and scale target in X                                           16
       X=xp.zeros((len(yplus_target),2))                                17
       X[:,0] = scaler_uplus.fit_transform(\                            18
                uplus_target)[:,0]                                      19
       X[:,1] = scaler_yplus.fit_transform(\                            20
                yplus_target)[:,0]                                      21
# build the tree                                                        22
       tree = KDTree(X)                                                 23
# the code below is executed every CFD iteration                        24
# take old ustar from previous iteration/time step                      25
    ustar=cmu**0.25*k3d[:,0,:]**0.5                                     26
# create a 2D arraay with wall-parallel velocity                        27
    j_wall = 0                                                          28
    u2d_wall=u3d[:,j_wall,:]                                            29
# cell-center wall distance                                             30
    dy=dist3d[:,j_wall,:]                                               31
# compute yplus and uplus                                               32
    yplus_south = ustar*dy/viscos                                       33
    uplus_south = u2d_wall/ustar                                        34
    yplus = yplus_south.reshape(-1,1)                                   35
    uplus = uplus_south.reshape(-1,1)                                   36
# store and scale yplus and uplus from CFD in x                         37
    x=xp.zeros((len(uplus),2))                                          38
    x[:,0] = scaler_uplus.transform(uplus)[:,0]                         39
    x[:,1] = scaler_yplus.transform(yplus)[:,0]                         40
# find one (K=1) nearest neighbor at distance ds                        41
    K=1                                                                 42
    ds, inds =  tree.query(x_np, K)                                     43
# set yplus and reshape                                                 44
    yplus_kdtree = yplus_target[inds,0]                                 45
    yplus_predict = xp.reshape(yplus_kdtree,(ni,nk))                    46
# compute u_tau                                                         47
    ustar=yplus_predict*viscos/dy                                       48
# compute k (turb. kinetic energy)                                      49
    kwall=cmu**(-0.5)*ustar**2                                          50
# fix k at wall-adjacent cells                                          51
    aw3d[:,0,:]=0                                                       52
    ae3d[:,0,:]=0                                                       53
    as3d[:,0,:]=0                                                       54
    an3d[:,0,:]=0                                                       55
    al3d[:,0,:]=0                                                       56
    ah3d[:,0,:]=0                                                       57
    ap_max=xp.max(ap3d)                                                 58
    ap3d[:,0,:]=ap_max                                                  59
    su3d[:,0,:]=ap_max*kwall                                            60
    return aw3d,ae3d,as3d,an3d,al3d,ah3d,ap3d,su3d,sp3d                 61
```

Listing 2. The `fix_k` module which includes the ML (**KDTree**) wall function. Line number are shown to the right.