

USING NEURAL NETWORK FOR WALL FUNCTIONS INCLUDING PRESSURE GRADIENTS

Lars Davidson

NFFP8 E-WMLES, 12 Dec 2023

TRAINING: I NEED A TARGET DATABASE

$$\frac{\partial \bar{v}_i}{\partial x_i} = 0$$
$$\frac{\partial \bar{v}_i}{\partial t} + \frac{\partial}{\partial x_j} (\bar{v}_i \bar{v}_j) = -\frac{\partial \bar{p}}{\partial x_i} + \frac{\partial}{\partial x_j} \left[(\nu + \nu_{sgs}) \frac{\partial \bar{v}_i}{\partial x_j} \right]$$

- Diffuser flow.
- **pyCALC-LES** [2] is used for all simulations

INPUT/OUTPUT

- Traditional wall laws: $\frac{U}{u_\tau} = f\left(\frac{u_\tau y}{\nu}\right)$

INPUT/OUTPUT

- Traditional wall laws: $\frac{U}{u_\tau} = f\left(\frac{u_\tau y}{\nu}\right)$
- Do the same in ML

INPUT/OUTPUT

- Traditional wall laws: $\frac{U}{u_\tau} = f\left(\frac{u_\tau y}{\nu}\right)$
- Do the same in ML

$$\begin{aligned} y_P^+ & : && \text{influence/inlet parameter} \\ P^+ = \nu(\partial \bar{p} / \partial x_1) / u_\tau^3 & : && \text{influence/inlet parameter} \\ U^+ & : && \text{output parameter} \\ u_\tau & : && \bar{u}_P / U^+ \end{aligned}$$

INPUT/OUTPUT

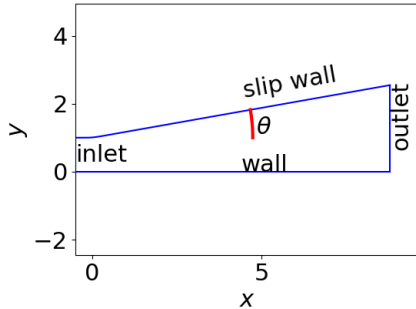
- Traditional wall laws: $\frac{U}{u_\tau} = f\left(\frac{u_\tau y}{\nu}\right)$
- Do the same in ML

$$\begin{aligned} y_P^+ & : && \text{influence/inlet parameter} \\ P^+ = \nu(\partial \bar{p} / \partial x_1) / u_\tau^3 & : && \text{influence/inlet parameter} \\ U^+ & : && \text{output parameter} \\ u_\tau & : && \bar{u}_P / U^+ \end{aligned}$$

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

DIFFUSER, LES WITH WALE MODEL, PRESSURE GRADIENT

- Well resolved LES, $600 \times 150 \times 300$, $0.3 < \Delta y^+ < 22$, $\Delta z^+ = 11$, $\Delta x^+ = 22$
- Inlet: precursor wall-resolved LES of flow in a half-channel at $Re_\tau = 2\,000$ ($Re_b = 50\,000$)
- Diffusion angle, short diffuser: $6 \leq \theta \leq 14^\circ$
- Diffusion angle, long diffuser: $8 \leq \theta \leq 12^\circ$



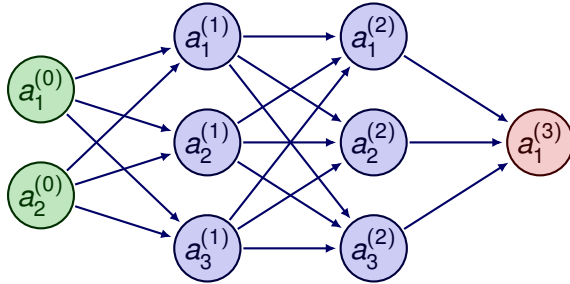
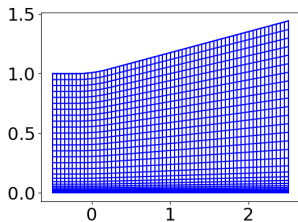
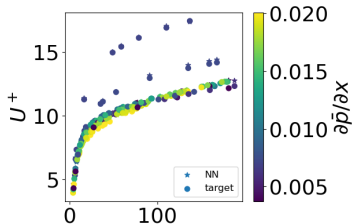


FIGURE: The Neural Network with two inputs variables, $a_1^{(0)} = y^+$ and $a_2^{(0)} = P^+$ and one output variable, $a_1^{(3)} = U^+$. There are three neurons in this figure; in the simulations I have 50.

LES, TRAINING TIME-AVERAGED DATA FOR NN, 10^0

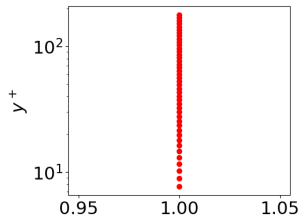


(A) LES grid.

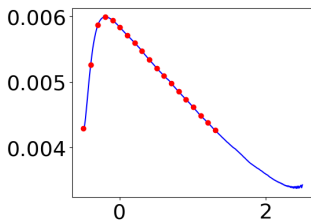


(B) U^+ v. y^+ . error

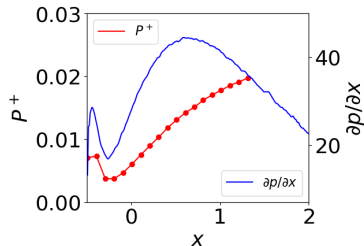
$$= \frac{\text{std}(U_{NN}^+ - U_{LES}^+)}{\text{mean}(U_{LES}^+)} < 0.01$$



(C) 43 Sampling points in y direction.



C_f



x

IDDES, WALL FUNCTIONS: SETUP

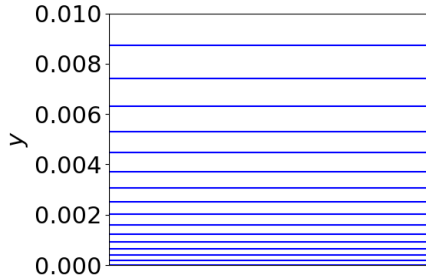
- Wall functions based on Neural Network (NN) or Reichardt wall functions
- Wall functions on Reichardt's law

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

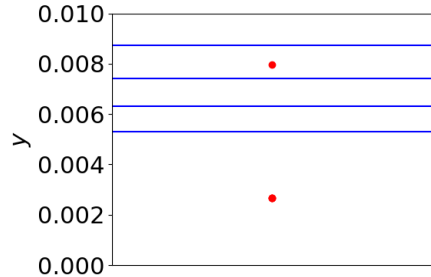
is solved using the Newton-Raphson method `scipy.optimize.newton` in Python.

- A course wall-adjacent cell and then finer cells further away from the wall (as in [3])
- Turbulence model: IDDES based on the AKN low-Re $k - \varepsilon$ model
- Pre-cursor channel IDDES with Reichardt's wall function
- Grid; $150 \times 73 \times 64$

GRID STRATEGY



(A) Low-Re number IDDES grid.

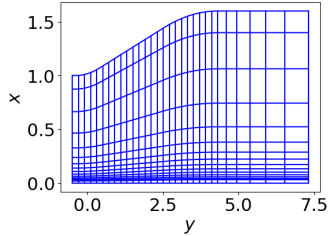


(B) Wall function grid. New grid strategy.

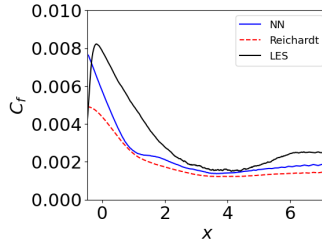
FIGURE: Different grids. — : grid lines.

- This strategy was used in [1] for channel flow and impinging jets (RANS)

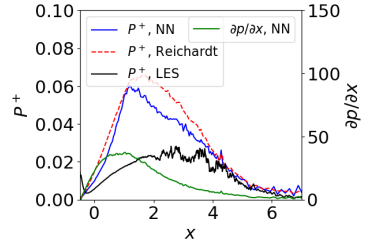
NN AND REICHARDT. $P_{min}^+ = 0.002$



(A) Grid

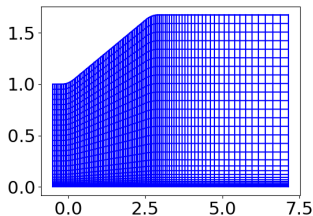


(B) C_f

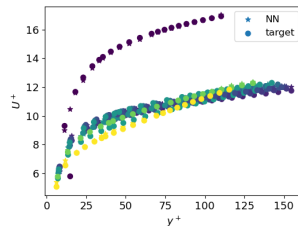


(C) P^+ and $\partial \bar{p} / \partial x$

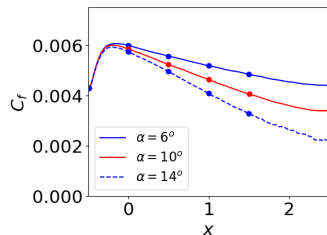
LES, TRAINING DATA FOR NN, 6° , 10° , 14° AND CHANNEL FLOW



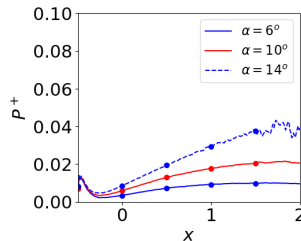
(A) LES grid, $\alpha = 14^\circ$.



(B) U^+ v. y^+ . error = $\frac{\text{std}(U_{NN}^+ - U_{LES}^+)}{\text{mean}(U_{LES}^+)} < 0.02$

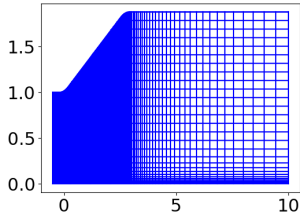


(C) C_f

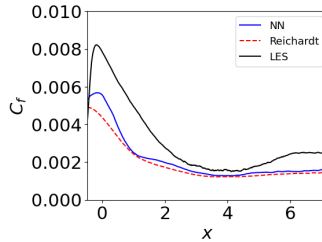


(D) P^+

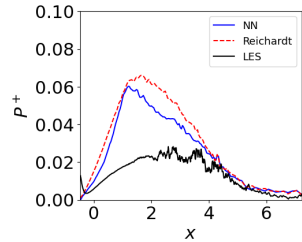
R. AND NN, 6° , 10° , 14° AND CHANNEL FLOW. $P_{min}^+ = -0.005$



(A) Grid, $\alpha = 18^\circ$

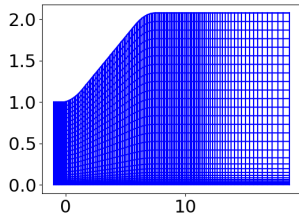


(B) C_f

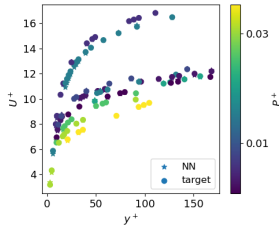


(C) P^+

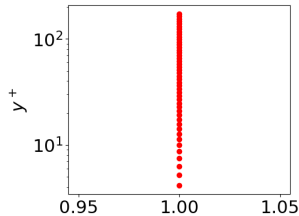
LES, TRAINING DATA FOR NN, 10^0 AND CHANNEL FLOW



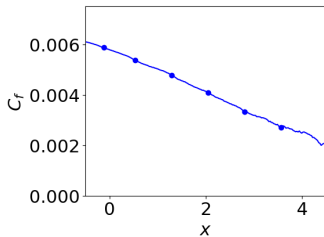
(A) LES grid.



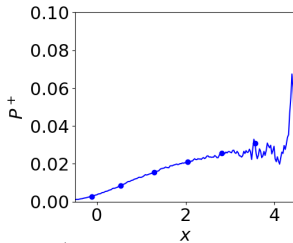
(B) U^+ v. y^+ . error < 0.01



(C) 46 Sampling points in y direction.

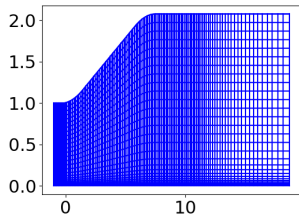


(D) C_f

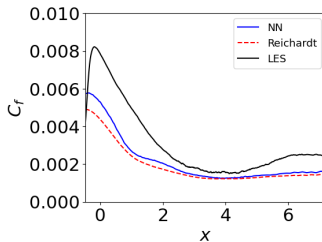


(E) P^+

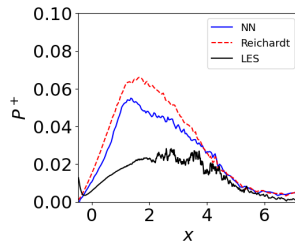
REICHARDT AND NN, 10^0 AND CHANNEL FLOW



(A) Grid



(B) C_f



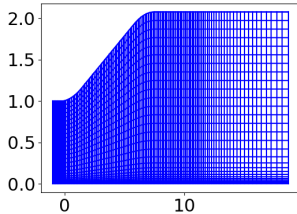
(C) P^+

- The input pressure gradient reads

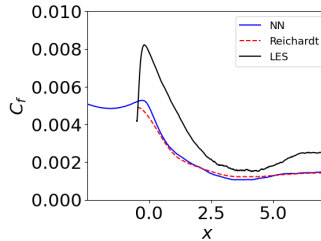
$$P^+ = \nu \frac{\partial \bar{p} / \partial x_1}{u_\tau^3}$$

- Both $\partial \bar{p} / \partial x_1$ and u_τ^3 are very unsteady and P^+ can become very large when u_τ gets small
- I always limit the input variable to min/max of training data: typical values of P_{min}^+ and P_{max}^+ are -0.005 and 0.02 , respectively.
- Instantaneous values can be $\pm 10^6$ close to end of the diffuser

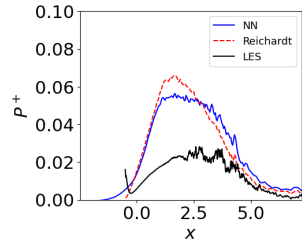
REICHARDT AND NN, 10^0 AND CHANNEL FLOW, P^+ AVERAGED



(A) Grid

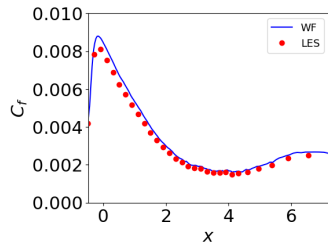


(B) C_f

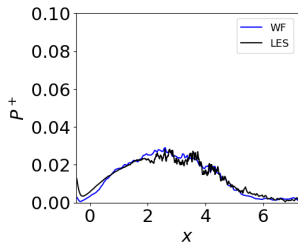


(C) P^+

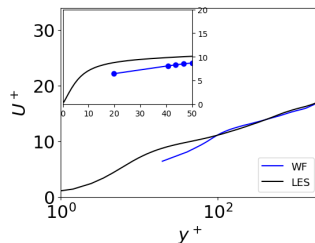
PREDICTIONS WITH WALL-FUNCTIONS: $\langle \tau_w \rangle(x)$ FROM LES



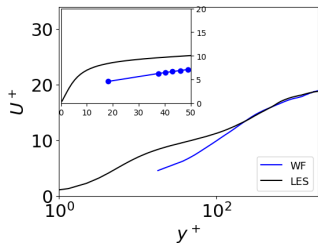
(A) C_f



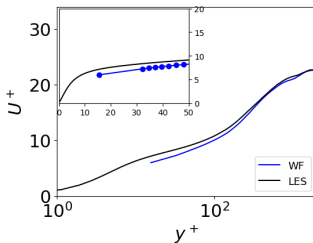
(B) P^+



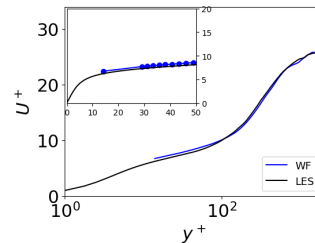
(C) $x = 0$



(D) $x = 1$

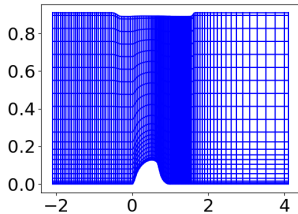


(E) $x = 2$



(F) $x = 3$

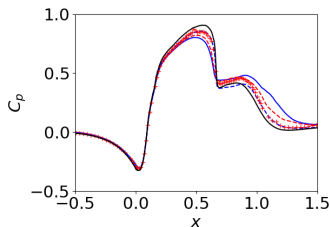
HUMP FLOW



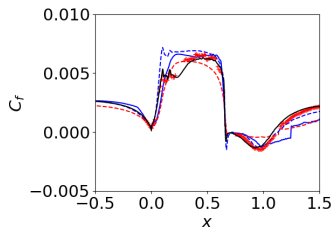
(A) Grid

- The Reynolds number is $Re_c = 936\,000$
- The spanwise extent is $z_{max} = 0.2$.
- The mesh has $582 \times 106 \times 32$ cells (x, y, z)
- Inlet b.c.
 - Mean from 2D RANS
 - Inlet turbulence: fluctuation from STG
 - Inlet k and ε : 2D RANS plus commutation term in k eq.
- Comparison with
 - Experiments [5, 4]
 - Well-resolved LES [6, 7, 8]. 420M cells, $\Delta x^+ \simeq 25$, $\Delta y^+ \simeq 12.5$, $\Delta z^+ \simeq 0.8$.
 - IDDES on the same $x - z$ mesh as WF but with $y^+ \simeq 1$

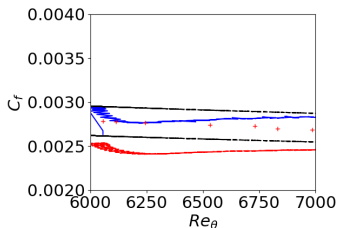
HUMP FLOW: PRESSURE & SKINFRICTION. $\langle P^+ \rangle_{\tau t}$



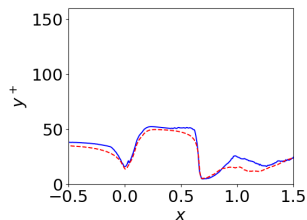
(A) Presssure coefficient



(B) C_f



(C) C_f downstream of the inlet



(D) y^+ for the wall-adjacent cells

FIGURE: — : WF, NN; - - : WF, Reichardt; + : exp; — : LES [6, 7, 8]; - - : low-Re IDDES

HUMP FLOW: VELOCITY. $\langle P^+ \rangle_{\tau t}$

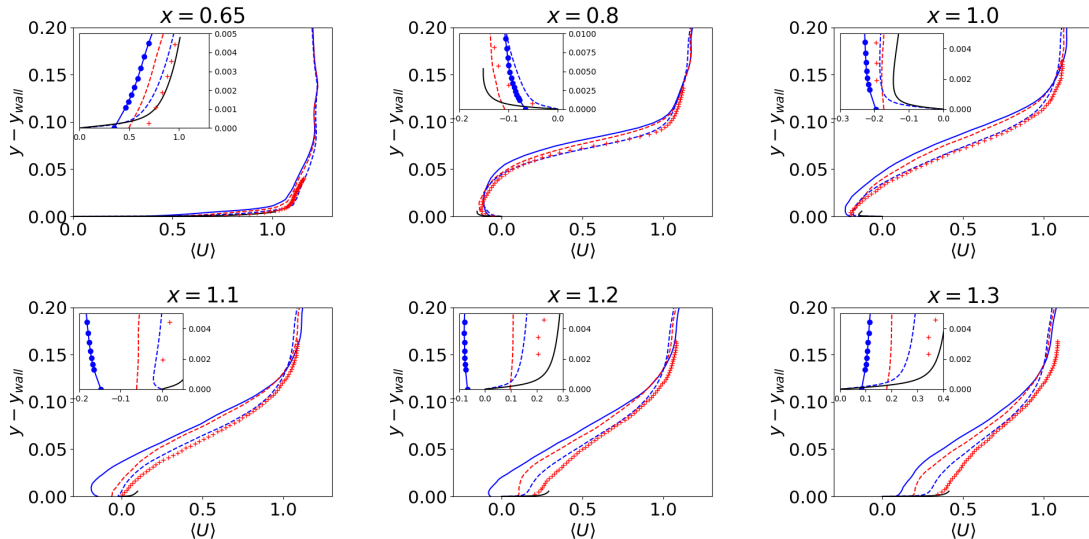
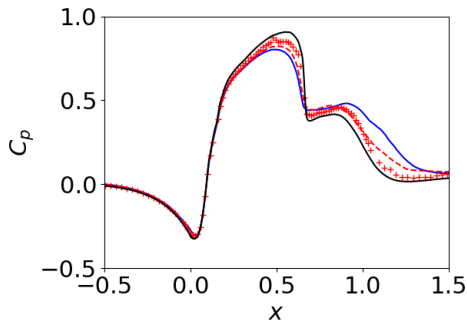


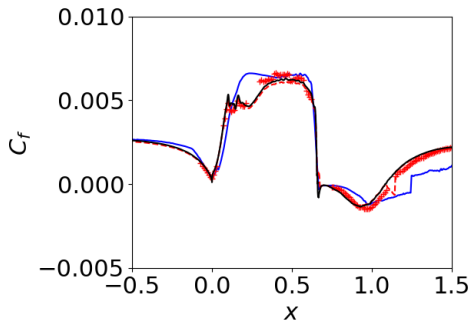
FIGURE: — : WF, NN; - - : WF, Reichardt; + : exp; — : LES [6, 7, 8] - - : low-Re IDDES

HUMP FLOW: PRESSURE & SKINFRICTION.

- Here I take $\langle \tau_w \rangle_{zt}$ from LES [6, 7, 8]



(A) Pressure coefficient



(B) C_f

FIGURE: — : WF, NN; - - : τ_w from LES; + : exp; — : LES [6, 7, 8]

HUMP FLOW: VELOCITY. $\langle \tau_w \rangle_{zt}$ FROM LES [6, 7, 8]

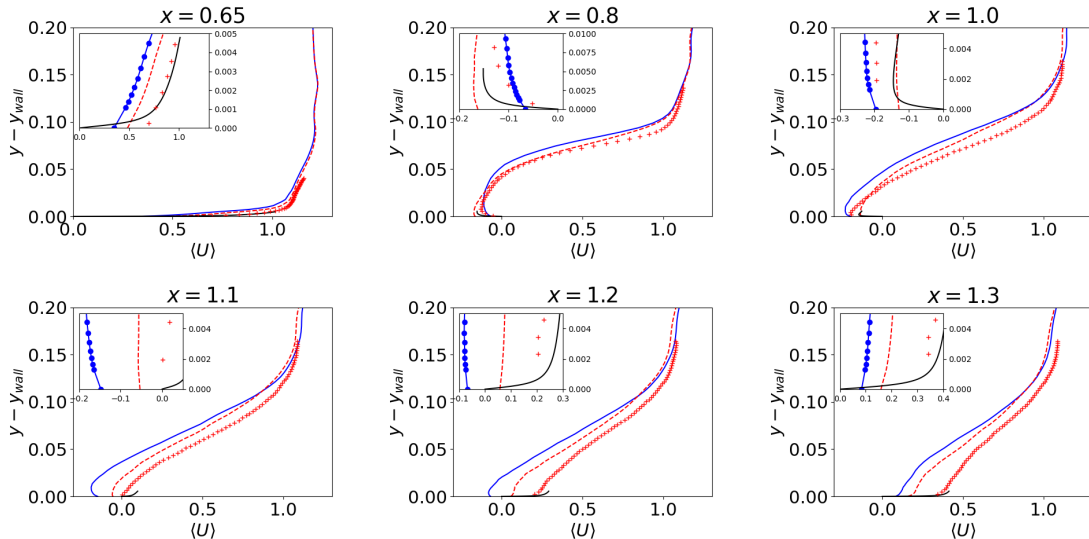
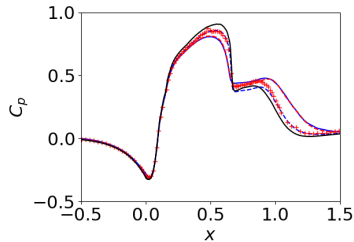
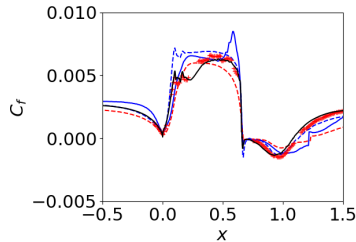


FIGURE: — : WF, NN; - - : C_f from LES; + : exp; — : LES [6, 7, 8]

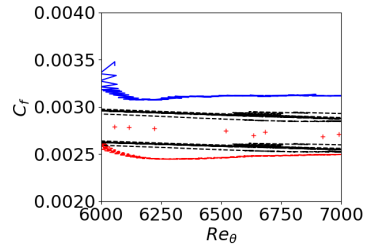
HUMP FLOW: PRESSURE & SKINFRICTION. INST. P^+



(A) Pressure coefficient



(B) C_f



(C) C_f downstream of the inlet and upstream of the hump

FIGURE: — : WF, NN; - - : WF, Reichardt; + : exp; — : LES [6, 7, 8]; - - : low-Re IDDES

HUMP FLOW: VELOCITY. INST. P^+

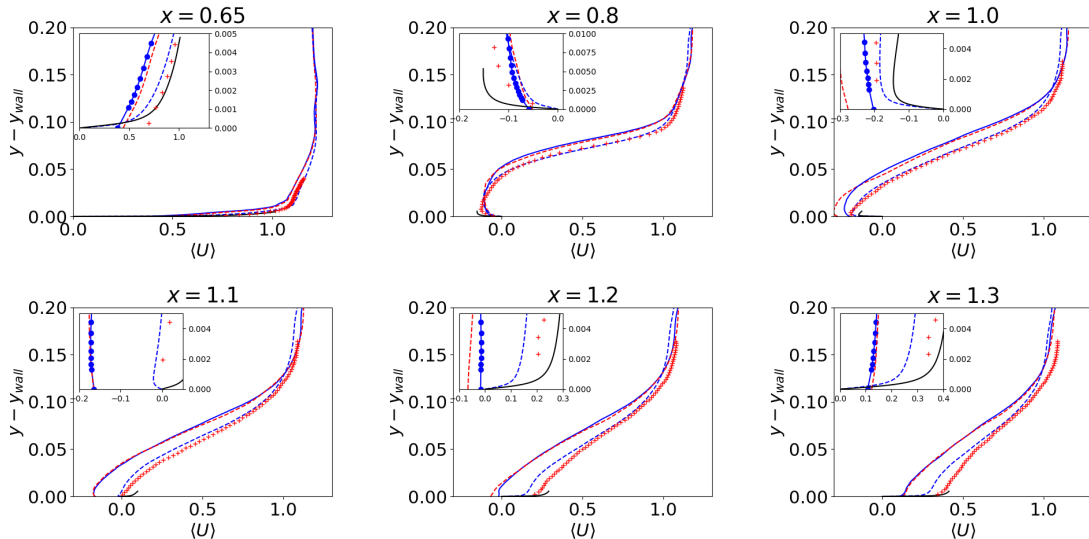
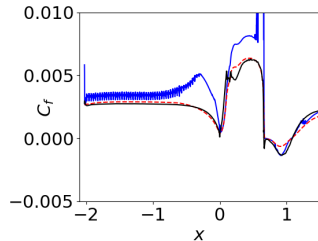
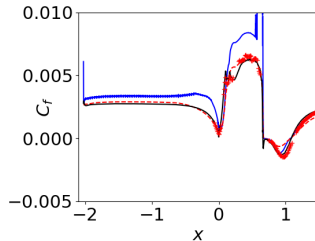


FIGURE: — : WF, NN; - - : WF, Reichardt; + : exp; — : LES [6, 7, 8] - - : low-Re IDDES

INPUT: LES DATA[6, 7, 8] (NO CFD). $y^+ \simeq 35$ AT THE INLET

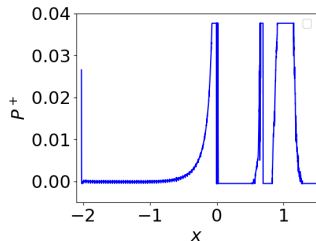


(A) $P^+ = \nu(\partial \bar{p} / \partial x_1) / u_\tau^3$

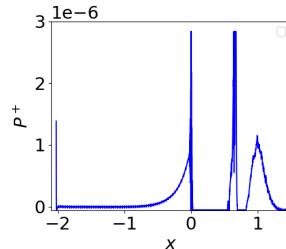


(B) $P^+ = \nu(\partial \bar{p} / \partial x_1) / U_b^3$

— : WF, NN
- - : WF, Reichardt

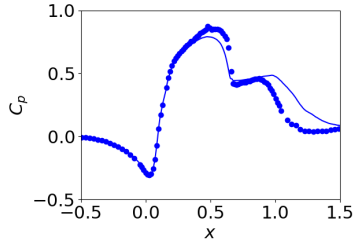


(C) $P^+ = \nu(\partial \bar{p} / \partial x_1) / u_\tau^3$

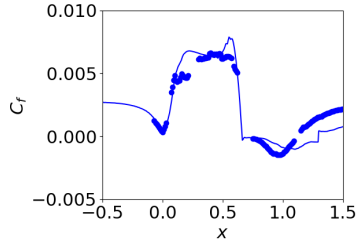


(D) $P^+ = \nu(\partial \bar{p} / \partial x_1) / U_b^3$

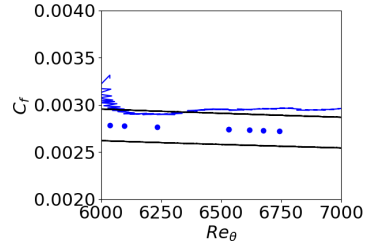
HUMP FLOW: PRESSURE & SKINFRICTION. $P^+ = \nu(\partial \bar{p} / \partial x_1) / U_b^3$



(A) Pressure coefficient



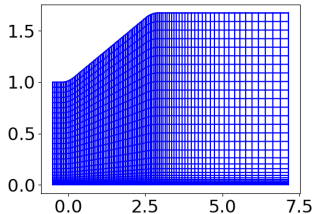
(B) C_f



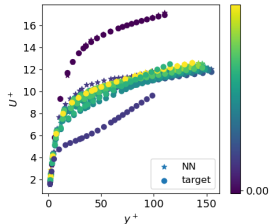
(C) C_f downstream of the inlet and upstream of the hump

FIGURE: — : WF, NN; ○ : exp; —

TRAINING DATA FOR NN, $6^\circ, \dots$, CHANNEL FLOW FOR $y^+ = 1 - 140$

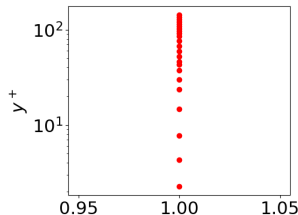


(A) LES grid, $\alpha = 14^\circ$.

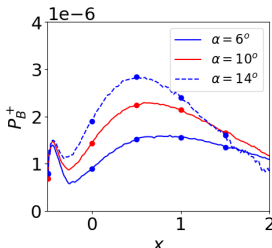
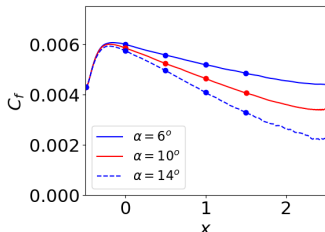


(B) U^+ v. y^+ . error

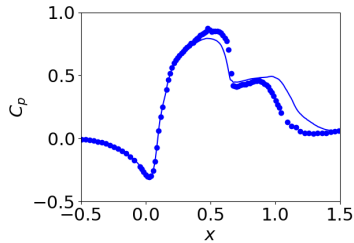
$$= \frac{\text{std}(U_{NN}^+ - U_{LES}^+)}{\text{mean}(U_{LES}^+)} < 0.023$$



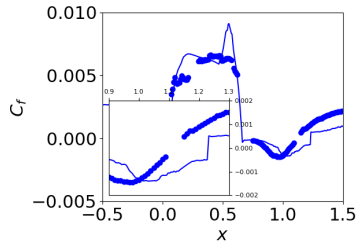
(C) 23 Sampling points in y direction.



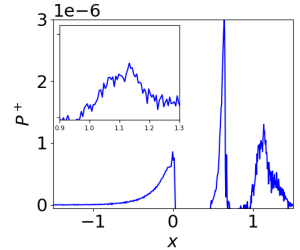
HUMP FLOW: PRESSURE & SKINFRICTION. $P^+ = \nu(\partial \bar{p} / \partial x_1) / U_b^3$, $y^+ = 1 - 140$



(A) Pressure coefficient



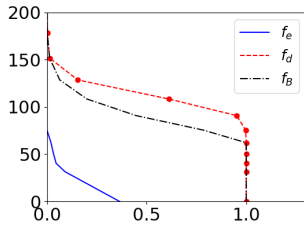
(B) C_f



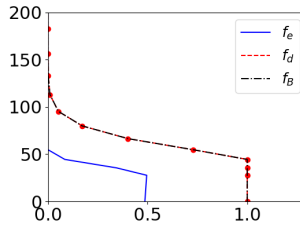
(C) P^+

FIGURE: — : WF, NN; ○: exp; —

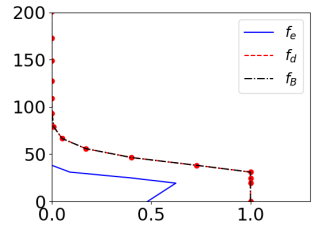
HUMP FLOW: IDDES DAMPING FUNCTIONS



(A) $x = 0.65$

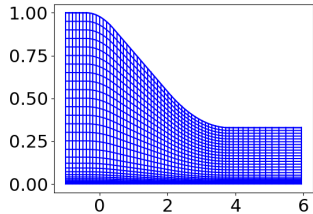


(B) $x = 1.10$

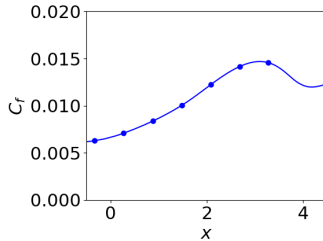


(C) $x = 1.30$

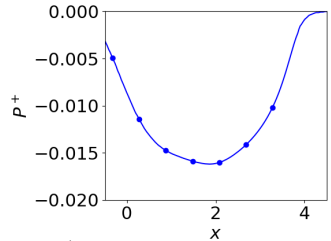
LES, TRAINING DATA, CONVERGENT CHANNEL OF 13°



(A) LES grid.



(B) C_f



(C) P^+

- These data can be added to, e.g, diffuser and channel data.

CONCLUSIONS

- The NN wall-function works well in channel and boundary flow, but not for recirculating flow

CONCLUSIONS

- The NN wall-function works well in channel and boundary flow, but not for recirculating flow
- I've trained the NN wall-function on high-resolved LES data; I think it's better to train it on IDDES data.

CONCLUSIONS

- The NN wall-function works well in channel and boundary flow, but not for recirculating flow
- I've trained the NN wall-function on high-resolved LES data; I think it's better to train it on IDDES data.
 - The objective would then be to make the NN wall-function as accurate as IDDES

CONCLUSIONS

- The NN wall-function works well in channel and boundary flow, but not for recirculating flow
- I've trained the NN wall-function on high-resolved LES data; I think it's better to train it on IDDES data.
 - The objective would then be to make the NN wall-function as accurate as IDDES
 - I can then create a database (using IDDES) where I store the **instantaneous** $\tau_{w,inst}$

CONCLUSIONS

- The NN wall-function works well in channel and boundary flow, but not for recirculating flow
- I've trained the NN wall-function on high-resolved LES data; I think it's better to train it on IDDES data.
 - The objective would then be to make the NN wall-function as accurate as IDDES
 - I can then create a database (using IDDES) where I store the **instantaneous** $\tau_{w,inst}$
 - I can then use $\tau_{w,inst}$ as an **exact** wall-function b.c.

CONCLUSIONS

- The NN wall-function works well in channel and boundary flow, but not for recirculating flow
- I've trained the NN wall-function on high-resolved LES data; I think it's better to train it on IDDES data.
 - The objective would then be to make the NN wall-function as accurate as IDDES
 - I can then create a database (using IDDES) where I store the **instantaneous** $\tau_{w,inst}$
 - I can then use $\tau_{w,inst}$ as an **exact** wall-function b.c.
 - This would tell me how good the NN wall-function can be.

CONCLUSIONS

- The NN wall-function works well in channel and boundary flow, but not for recirculating flow
- I've trained the NN wall-function on high-resolved LES data; I think it's better to train it on IDDES data.
 - The objective would then be to make the NN wall-function as accurate as IDDES
 - I can then create a database (using IDDES) where I store the **instantaneous** $\tau_{w,inst}$
 - I can then use $\tau_{w,inst}$ as an **exact** wall-function b.c.
 - This would tell me how good the NN wall-function can be.
- I will start with diffuser flow (e.g. 10°) at $Re_\tau = 5\,200$ (and longer inlet region)

CONCLUSIONS

- The NN wall-function works well in channel and boundary flow, but not for recirculating flow
- I've trained the NN wall-function on high-resolved LES data; I think it's better to train it on IDDES data.
 - The objective would then be to make the NN wall-function as accurate as IDDES
 - I can then create a database (using IDDES) where I store the **instantaneous** $\tau_{w,inst}$
 - I can then use $\tau_{w,inst}$ as an **exact** wall-function b.c.
 - This would tell me how good the NN wall-function can be.
- I will start with diffuser flow (e.g. 10°) at $Re_\tau = 5\,200$ (and longer inlet region)
- Then move on to the hump flow

CONCLUSIONS

- The NN wall-function works well in channel and boundary flow, but not for recirculating flow
- I've trained the NN wall-function on high-resolved LES data; I think it's better to train it on IDDES data.
 - The objective would then be to make the NN wall-function as accurate as IDDES
 - I can then create a database (using IDDES) where I store the **instantaneous** $\tau_{w,inst}$
 - I can then use $\tau_{w,inst}$ as an **exact** wall-function b.c.
 - This would tell me how good the NN wall-function can be.
- I will start with diffuser flow (e.g. 10°) at $Re_\tau = 5\,200$ (and longer inlet region)
- Then move on to the hump flow
- If it does not work well I will train a ML wall-function using instantaneous IDDES data



CONCLUSIONS

- The NN wall-function works well in channel and boundary flow, but not for recirculating flow
- I've trained the NN wall-function on high-resolved LES data; I think it's better to train it on IDDES data.
 - The objective would then be to make the NN wall-function as accurate as IDDES
 - I can then create a database (using IDDES) where I store the **instantaneous** $\tau_{w,inst}$
 - I can then use $\tau_{w,inst}$ as an **exact** wall-function b.c.
 - This would tell me how good the NN wall-function can be.
- I will start with diffuser flow (e.g. 10°) at $Re_\tau = 5\,200$ (and longer inlet region)
- Then move on to the hump flow
- If it does not work well I will train a ML wall-function using instantaneous IDDES data
 - I should then probably use Random Forest (`RandomForestRegressor`) or Nearest neighbor (`scipy.spatial.KDTree`)


CONCLUSIONS

- The NN wall-function works well in channel and boundary flow, but not for recirculating flow
- I've trained the NN wall-function on high-resolved LES data; I think it's better to train it on IDDES data.
 - The objective would then be to make the NN wall-function as accurate as IDDES
 - I can then create a database (using IDDES) where I store the **instantaneous** $\tau_{w,inst}$
 - I can then use $\tau_{w,inst}$ as an **exact** wall-function b.c.
 - This would tell me how good the NN wall-function can be.
- I will start with diffuser flow (e.g. 10°) at $Re_\tau = 5\,200$ (and longer inlet region)
- Then move on to the hump flow
- If it does not work well I will train a ML wall-function using instantaneous IDDES data
 - I should then probably use Random Forest (`RandomForestRegressor`) or Nearest neighbor (`scipy.spatial.KDTree`)
- I'm still worried about the hack in C_f for the hump flow near re-attachment ...

REFERENCES

- [1] J.-A. Bäckar and L. Davidson. Evaluation of numerical wall functions on the axisymmetric impinging jet using OpenFOAM. *International Journal of Heat and Fluid Flow*, 67:27–42, 2017.
- [2] L. Davidson. pyCALC-LES: a Python code for DNS, LES and Hybrid LES-RANS . Division of Fluid Dynamics, Dept. of Mechanics and Maritime Sciences, Chalmers University of Technology, Gothenburg, 2021.
- [3] L. Davidson. 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*.
- [4] D. Greenblatt, K. B. Paschal, C.-S. Yao, and J. Harris. A separation control CFD validation test case Part 1: Zero efflux oscillatory blowing. AIAA-2005-0485, 2005.

REFERENCES

- [5] D. Greenblatt, K. B. Paschal, C.-S. Yao, J. Harris, N. W. Schaeffler, and A. E. Washburn. A separation control CFD validation test case. Part 1: Baseline & steady suction. AIAA-2004-2220, 2004.
- [6] A. Uzun and M. R. Malik. LES: Compressible 2-D NASA Wall-Mounted Hump. Database . Langley Research Center, Turbulence Modeling Resource, 2017.
- [7] Ali Uzun and Mujeeb R. Malik. Wall-resolved large-eddy simulation of flow separation over nasa wall-mounted hump. In *55th AIAA Aerospace Sciences Meeting*, 2017.
- [8] Ali Uzun and Mujeeb R. Malik. Large-eddy simulation of flow over a wall-mounted hump with separation and reattachment. *AIAA Journal*, 56(2):715–730, 2018.