Using Machine Learning for formulating new wall functions for Detached Eddy Simulation: presented at ETMM14 [2]

Lars Davidson

Lars Davidson, M2 Fluid Dynamics
Chalmers University of Technology
Gothenburg, Sweden

- Machine learning (ML) is a method where known data are used for teaching the algorithm to classify another set of data.

- Machine learning (ML) is a method where known data are used for teaching the algorithm to classify another set of data.
  - Photographs where the machine learning algorithm should recognize, e.g., traffic lights [5].

- Machine learning (ML) is a method where known data are used for teaching the algorithm to classify another set of data.
  - Photographs where the machine learning algorithm should recognize, e.g., traffic lights [5].
  - ECG signals where the machine learning algorithm should recognize certain unhealthy conditions of the heart [3].

- Machine learning (ML) is a method where known data are used for teaching the algorithm to classify another set of data.
  - Photographs where the machine learning algorithm should recognize, e.g., traffic lights [5].
  - ECG signals where the machine learning algorithm should recognize certain unhealthy conditions of the heart [3].
  - Detecting fraud for credit card payments [4].

- Machine learning (ML) is a method where known data are used for teaching the algorithm to classify another set of data.
  - Photographs where the machine learning algorithm should recognize, e.g., traffic lights [5].
  - ECG signals where the machine learning algorithm should recognize certain unhealthy conditions of the heart [3].
  - Detecting fraud for credit card payments [4].
  - Machine learning methods such as Support Vector Machines (SVM) and neural networks are used for solving this type of problems.

- Machine learning (ML) is a method where known data are used for teaching the algorithm to classify another set of data.
  - Photographs where the machine learning algorithm should recognize, e.g., traffic lights [5].
  - ECG signals where the machine learning algorithm should recognize certain unhealthy conditions of the heart [3].
  - Detecting fraud for credit card payments [4].
  - Machine learning methods such as Support Vector Machines (SVM) and neural networks are used for solving this type of problems.
  - Through as much data as possible at ML?

- Machine learning (ML) is a method where known data are used for teaching the algorithm to classify another set of data.
  - Photographs where the machine learning algorithm should recognize, e.g., traffic lights [5].
  - ECG signals where the machine learning algorithm should recognize certain unhealthy conditions of the heart [3].
  - Detecting fraud for credit card payments [4].
  - Machine learning methods such as Support Vector Machines (SVM) and neural networks are used for solving this type of problems.
  - Through as much data as possible at ML?
- In my case, input and output are numerical values. Regression methods (SVR or NN) should be used [3]; I use support vector regression (SVR) methods in Python.

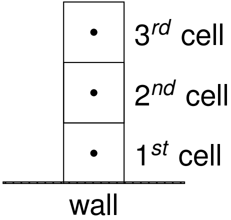
$$
\begin{aligned}
\frac{\partial \bar{v}_i}{\partial x_i} &= 0 \\
\frac{\partial \bar{v}_i}{\partial t} + \frac{\partial}{\partial x_j}\left(\bar{v}_i \bar{v}_j\right) &= -\frac{\partial \bar{p}}{\partial x_i} + \frac{\partial}{\partial x_j}\left[(\nu + \nu_{sgs})\frac{\partial \bar{v}_i}{\partial x_j}\right]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

- Fully-developed Channel flow
- IDDES. $96 \times 96 \times 96$, Reynolds number is 5 200
- Database: average in x and z

$$
\begin{aligned}
\bar{U}_{1^{st}}(x, z) &= \frac{1}{\Delta X \Delta Z} \int_{x,z}^{x+\Delta X, z+\Delta Z} \bar{u} dx dz \\
\bar{u}_\tau(x, z) &= \frac{1}{\Delta X \Delta Z} \int_{x,z}^{x+\Delta X, z+\Delta Z} u_\tau dx dz
\end{aligned}
$$

MACHINE LEARNING

- LES with wall functions: the object is to develop a model for the wall shear stress, $\tau_w = \rho u_\tau^2$

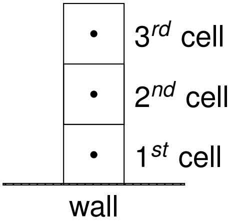


1^{st} cell	$\langle \Delta y^+ \rangle$
Location 1	12
Location 2	31
Location 3	49
Location 4	66
Location 5	76
Location 6	88
Location 7	135
Location 8	155
Location 9	207

300 independent instantaneous samples of $\bar{U}$ stored at all 3×9 cells

MACHINE LEARNING

- LES with wall functions: the object is to develop a model for the wall shear stress, $\tau_w = \rho u_\tau^2$
- Input data: U_P, y_P, $\partial \bar{U}/\partial y$, $\partial^2 \bar{U}/\partial^2 y$

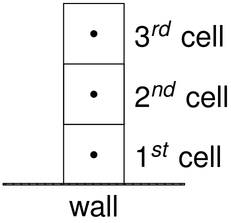


1^{st} cell	$\langle \Delta y^+ \rangle$
Location 1	12
Location 2	31
Location 3	49
Location 4	66
Location 5	76
Location 6	88
Location 7	135
Location 8	155
Location 9	207

300 independent instantaneous samples of $\bar{U}$ stored at all 3×9 cells

Machine Learning

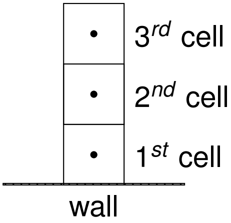
- LES with wall functions: the object is to develop a model for the wall shear stress, $\tau_w = \rho u_\tau^2$
- Input data: U_P, y_P, $\partial \bar{U}/\partial y$, $\partial^2 \bar{U}/\partial^2 y$
- output data: u_τ



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

300 independent instantaneous samples of $\bar{U}$ stored at all 3×9 cells

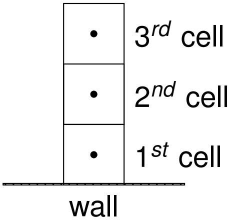
- LES with wall functions: the object is to develop a model for the wall shear stress, $\tau_w = \rho u_\tau^2$
- Input data: U_P, y_P, $\partial \bar{U}/\partial y$, $\partial^2 \bar{U}/\partial^2 y$
- output data: u_τ
- Non-dimensional: $\frac{u_\tau}{\langle u_\tau \rangle} = f\left(Re, y^+, T\partial \bar{U}/\partial y, \partial^2 \bar{U}/\partial y^2/(\bar{U}T^2)\right)$



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

300 independent instantaneous samples of $\bar{U}$ stored at all 3×9 cells

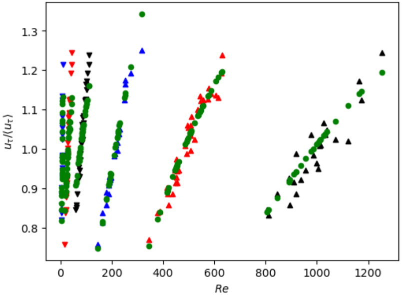
- LES with wall functions: the object is to develop a model for the wall shear stress, $\tau_w = \rho u_\tau^2$
- Input data: U_P, y_P, $\partial \bar{U}/\partial y$, $\partial^2 \bar{U}/\partial^2 y$
- output data: u_τ
- Non-dimensional: $\frac{u_\tau}{\langle u_\tau \rangle} = f\left(Re, y^+, T\partial \bar{U}/\partial y, \partial^2 \bar{U}/\partial y^2/(\bar{U}T^2)\right)$
- $T = \nu/\bar{U}^2$



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

300 independent instantaneous samples of $\bar{U}$ stored at all 3×9 cells

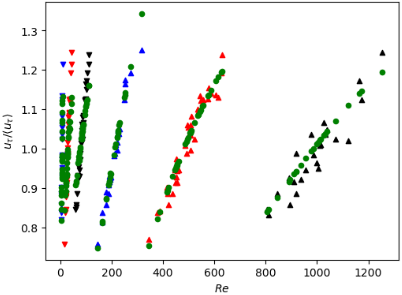
PREDICTED OUTPUT USING ML: 1ST ATTEMPT



(A) ▲: IDDES, Location 1; ▲: IDDES, Location 2; ▲: IDDES, Location 3; ▼: IDDES, Location 4; ▼: IDDES, Location 5; ▼: IDDES, Location 6. ○: `svr`.

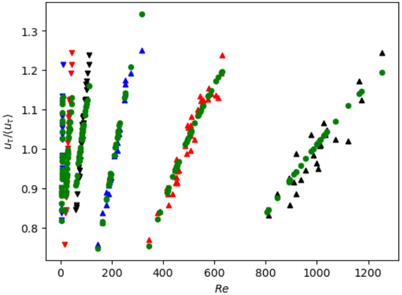
- Output on *y* axis

PREDICTED OUTPUT USING ML: 1ST ATTEMPT



(A) ▲: IDDES, Location 1; ▲: IDDES, Location 2; ▲: IDDES, Location 3; ▼: IDDES, Location 4; ▼: IDDES, Location 5; ▼: IDDES, Location 6. ○: `svr`.

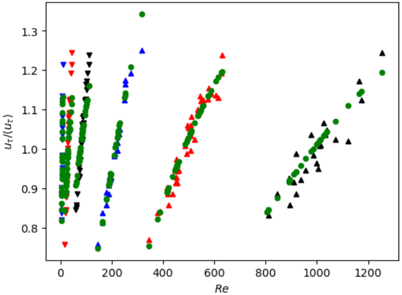
- Output on *y* axis
- Input on *x* axis



(A) ▲: IDDES, Location 1; ▲: IDDES, Location 2; ▲: IDDES, Location 3; ▼: IDDES, Location 4; ▼: IDDES, Location 5; ▼: IDDES, Location 6. ○: `svr`.

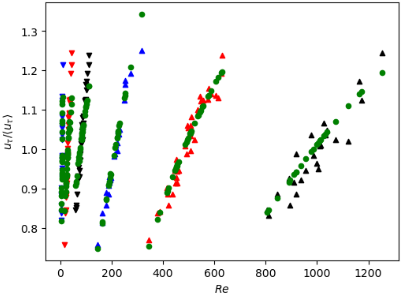
- Output on *y* axis
- Input on *x* axis
- Location 1 – 6 of data

PREDICTED OUTPUT USING ML: 1ST ATTEMPT



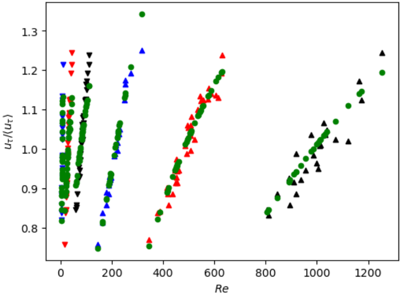
(A) ▲: IDDES, Location 1; ▲: IDDES, Location 2; ▲: IDDES, Location 3; ▼: IDDES, Location 4; ▼: IDDES, Location 5; ▼: IDDES, Location 6. ○: `svr`.

- Output on *y* axis
- Input on *x* axis
- Location 1 – 6 of data
- Difficult to interpolate



(A) ▲: IDDES, Location 1; ▲: IDDES, Location 2; ▲: IDDES, Location 3; ▼: IDDES, Location 4; ▼: IDDES, Location 5; ▼: IDDES, Location 6. ○: svr.

- Output on y axis
- Input on x axis
- Location 1 – 6 of data
- Difficult to interpolate
- Remedy: I included $\langle y^+ \rangle$ as input parameter = Location



(A) ▲: IDDES, Location 1; ▲: IDDES, Location 2; ▲: IDDES, Location 3; ▼: IDDES, Location 4; ▼: IDDES, Location 5; ▼: IDDES, Location 6. ○: `svr`.

- Output on y axis
- Input on x axis
- Location 1 – 6 of data
- Difficult to interpolate
- Remedy: I included $\langle y^+\rangle$ as input parameter = Location
- $\frac{u_\tau}{\langle u_\tau\rangle} = f\left(Re, \langle y^+\rangle\right)$

Bad choice of input/output: 2nd attempt

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

Bad choice of input/output: 2nd attempt

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

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

$$
\begin{array}{rcl}
y^+ & : & \text{influence par.} \\
U^+ & : & \text{output par.} \\
u_\tau & : & \bar{u}/U^+
\end{array}
$$

BAD CHOICE OF INPUT/OUTPUT: 2ND ATTEMPT

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

$$
\begin{aligned}
y^+ &: \quad \text{influence par.} \\
U^+ &: \quad \text{output par.} \\
u_\tau &: \quad \bar{u}/U^+
\end{aligned}
$$

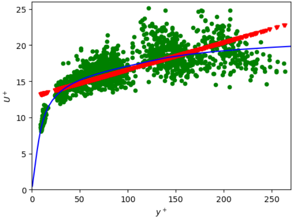
$$
\begin{aligned}
\rho u_\tau^2 &: \quad \tau_w \text{ in } \bar{\text{u}} \text{ eq.} \\
C_\mu^{-1/2} u_\tau^2 &: \quad \text{fix k} \\
\frac{u_\tau^3}{\kappa y} &: \quad \text{fix } \varepsilon
\end{aligned}
$$

Bad choice of input/output: 2nd attempt

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

$$
\begin{aligned}
y^+ &: \quad \text{influence par.}\\
U^+ &: \quad \text{output par.}\\
u_\tau &: \quad \bar{u}/U^+
\end{aligned}
$$

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



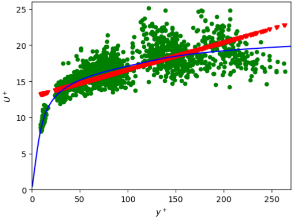
: $\langle \bar{u} \rangle$, IDDES; ▼: `svrLINEAR`: ●: IDDES, test data. 9% normalized error.

BAD CHOICE OF INPUT/OUTPUT: 2ND ATTEMPT

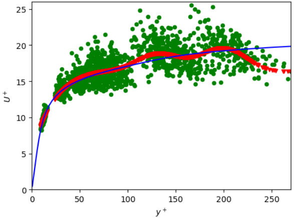
- Traditional wall laws: $\frac{U}{u_\tau} = f\left(\frac{u_\tau y}{\nu}\right)$
- Do the same in ML

$$
\begin{array}{rcl}
y^+ & : & \text{influence par.} \\
U^+ & : & \text{output par.} \\
u_\tau & : & \bar{u}/U^+
\end{array}
$$

$$
\begin{array}{rcl}
\rho u_\tau^2 & : & \tau_w \text{ in } \bar{\mathrm{u}} \text{ eq.} \\
C_\mu^{-1/2} u_\tau^2 & : & \text{fix k} \\
\dfrac{u_\tau^3}{\kappa y} & : & \text{fix } \varepsilon
\end{array}
$$



: $\langle \bar{u} \rangle$, IDDES; ▼: `svrLINEAR`: ●: IDDES, test data. 9% normalized error.



: $\langle \bar{u} \rangle$, IDDES; ▼: `svr`; ●: IDDES, test data. 9%

- The machine-learning wall functions will be compared to wall functions based on Reichardt's law

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

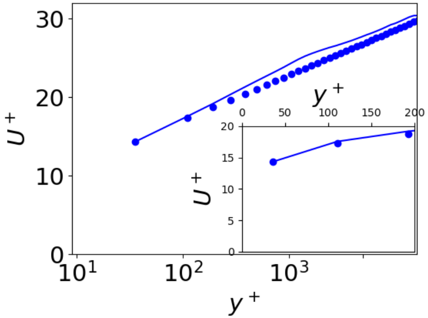
- The friction velocity is then obtained by solving the implicit equation

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

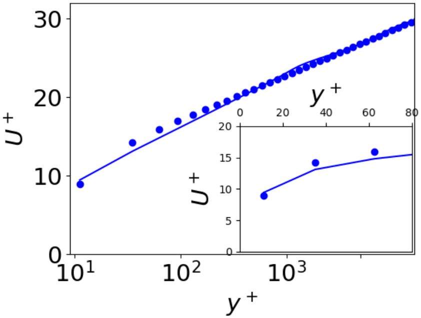
using the Newton-Raphson method `scipy.optimize.newton` in Python.

- $\bar{u}_P$ denotes the wall-parallel velocity in the first, second or third wall-adjacent cell.

RESULTS, CHANNEL FLOW, ML, $Re_\tau = 16\,000$

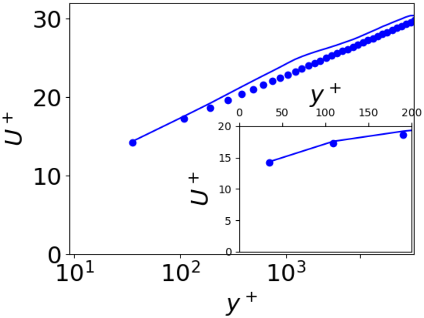


(A) $N_y = 66$, stretching 11%.

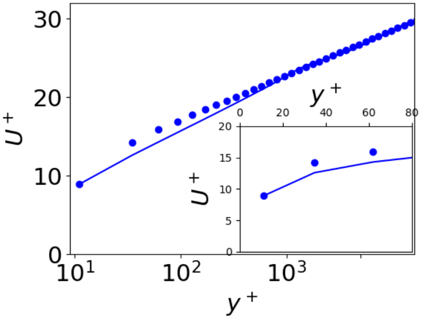


(B) $N_y = 68$, stretching 14.7%.

FIGURE: Channel flow. `svr`. $Re_\tau = 16\,000$. Velocity. $\bullet$: Reichardt's law.

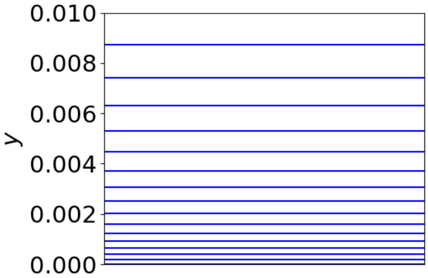


(A) $N_y = 66$, stretching 11%.

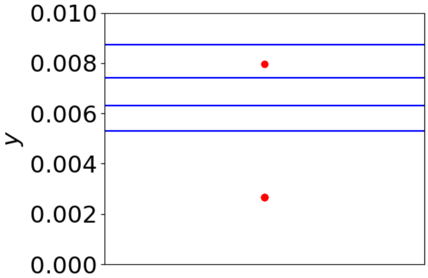


(B) $N_y = 68$, stretching 14.7%.

Figure: Channel flow. Reichardt's wall function. $Re_\tau = 16\,000$. Velocity. •: Reichardt's law.



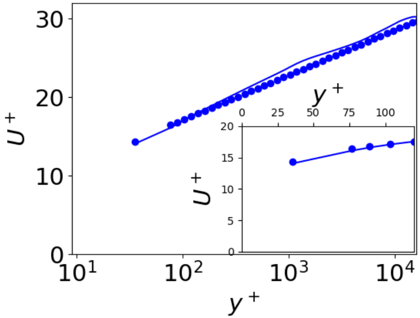
(A) Low-Re number IDDES grid.



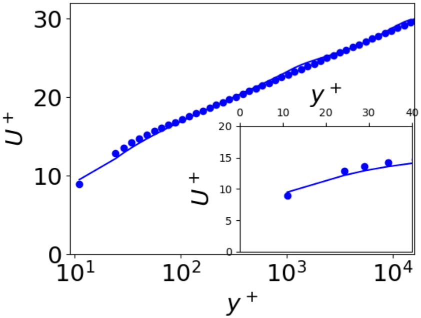
(B) Wall function grid. New grid strategy.

FIGURE: Different grids. —— : grid lines.

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

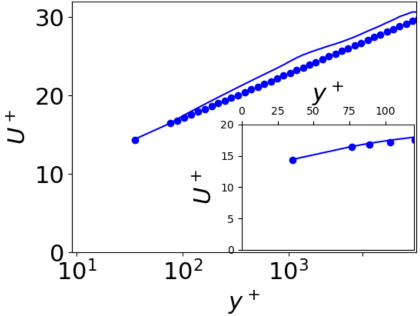


(A) $N_y = 78$.

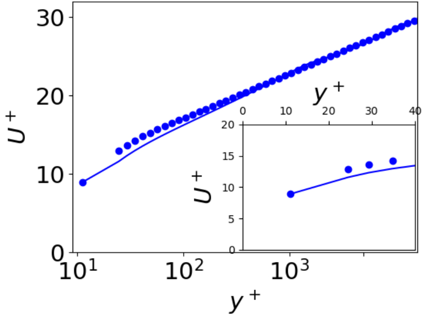


(B) $N_y = 92$.

FIGURE: Channel flow. $Re_\tau = 16\,000$. Velocity. `svr`. $\bullet$: Reichardt's law.

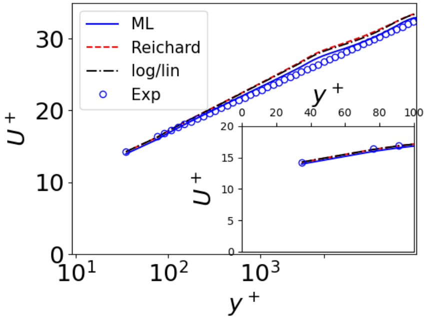


(A) $N_y = 78$.

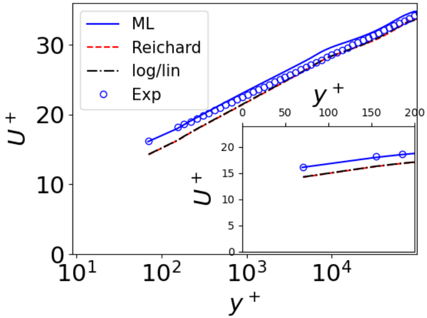


(B) $N_y = 92$.

Figure: Channel flow. $Re_\tau = 16\,000$. Velocity. Reichardt's wall function. •: Reichardt's law.



(A) $N_y = 92.\ y_1^+ = 35.\ \frac{(\Delta y)_1}{(\Delta y)_2} \simeq 5\ Re_\tau = 50\,000$.

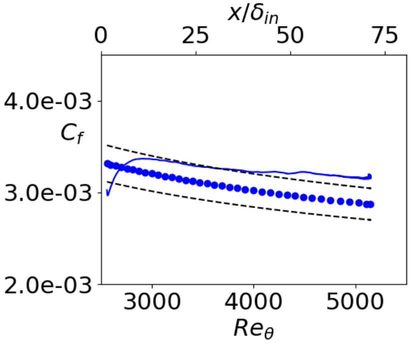


(B) $N_y = 92.\ y_1^+ = 70.\ \frac{(\Delta y)_1}{(\Delta y)_2} \simeq 5\ Re_\tau = 100\,000$.

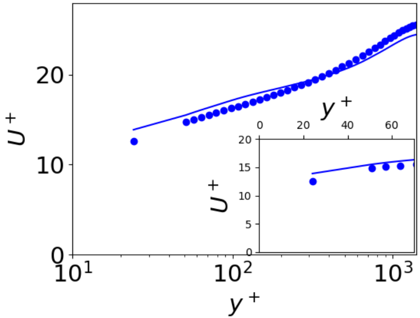
FIGURE: Channel flow. Velocity.

- $Re_\theta = U_{free}\theta/\nu = 2\,550$ at the inlet.
- Domain $(96 \times 7 \times 5)\delta_{in}$.
- Grid $(550 \times 82 \times 64)$.

- u_τ computed using 3^{rd} cell

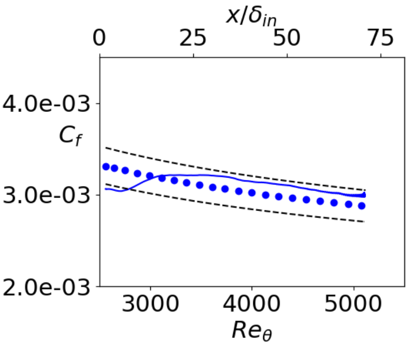


(A) Skin friction.

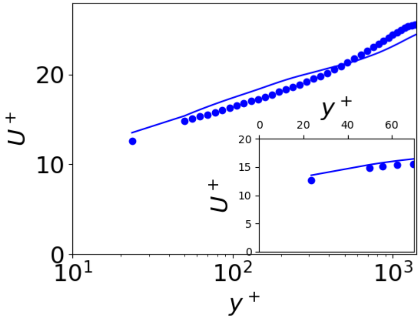


(B) Velocity at $Re_\theta = 4\,000$. Markers: DNS [6]

FIGURE: Boundary layer flow. $Re_\theta = 2\,500$ at inlet. `svr`. $N_y = 82$



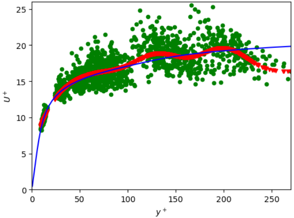
(A) Skin friction.



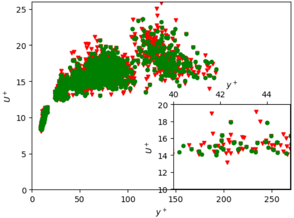
(B) Velocity at $Re_\theta = 4\,000$. Markers: DNS [6]

FIGURE: Boundary layer flow. `svr`. $N_y = 82$, $N_k = 32$, $\Delta x_{in} = 2\Delta x_{in,base}$

- Instantaneous data are used for training `svr`
- `svr` finds the time-averaged regression line (shown by ▼ in Fig. A)
- If I want *instantaneous* u_τ, I could find nearest neighbour (shown by ● in Fig. B)

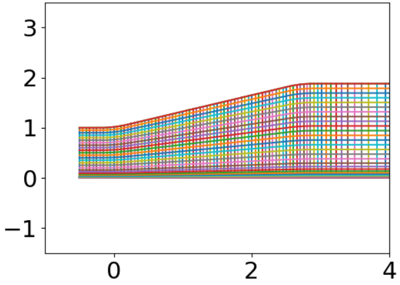


(A) —— : $\langle \bar{u} \rangle$, IDDES; ▼: `svr`: ●: IDDES, target data. 9% normalized error.

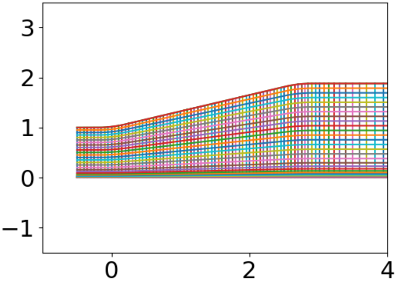


(B) Nearest neighbor using Python's `scipy.spatial.KDTree` ▼: KDTree; ●: IDDES, target data; 0.7% normalized error.

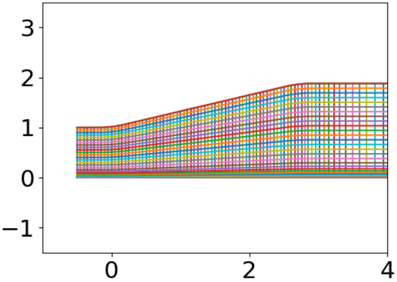
- Inlet: precursor wall-resolved LES of flow in a half-channel at $Re_\tau = 2\,000$ ($Re_b = 50\,000$). $600 \times 150 \times 300$, $0.3 < \Delta y^+ < 22$, $\Delta z^+ = 11$, $\Delta x^+ = 22$



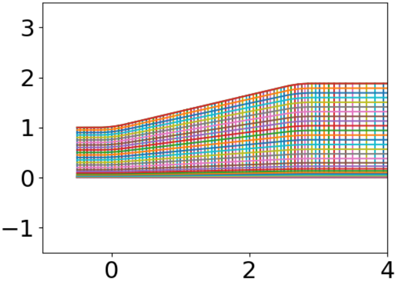
- Inlet: precursor wall-resolved LES of flow in a half-channel at $Re_\tau = 2\,000$ ($Re_b = 50\,000$). $600 \times 150 \times 300$, $0.3 < \Delta y^+ < 22$, $\Delta z^+ = 11$, $\Delta x^+ = 22$
- Diffuser: same mash at inlet; for $x \geq 2.9$, 1.005% stretching.



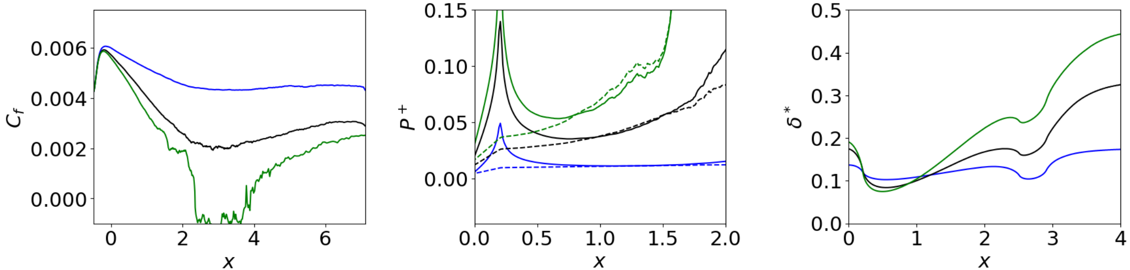
- Inlet: precursor wall-resolved LES of flow in a half-channel at $Re_\tau = 2\,000$ ($Re_b = 50\,000$). $600 \times 150 \times 300$, $0.3 < \Delta y^+ < 22$, $\Delta z^+ = 11$, $\Delta x^+ = 22$
- Diffuser: same mash at inlet; for $x \geq 2.9$, 1.005% stretching.
- Diffusion angle: $0 \leq \theta \leq 18^o$



- Inlet: precursor wall-resolved LES of flow in a half-channel at $Re_\tau = 2\,000$ ($Re_b = 50\,000$). $600 \times 150 \times 300$, $0.3 < \Delta y^+ < 22$, $\Delta z^+ = 11$, $\Delta x^+ = 22$
- Diffuser: same mash at inlet; for $x \geq 2.9$, 1.005% stretching.
- Diffusion angle: $0 \leq \theta \leq 18^o$
- Upper wall: slip



DIFFUSER, RESULTS: $6 \leq \theta \leq 18^o$



(A) Skin friction.

(B) Pressure. $P^+ = \dfrac{\nu \partial \bar{p}/\partial x}{u_\tau^3}$; dashed lines: Bernoulli

(C) Displacement thickness.

FIGURE: blue line: $\alpha = 6^o$; red line: $\alpha = 10^o$; black line: $\alpha = 14^o$; green line: $\alpha = 18^o$; .

CONCLUSIONS

- Machine Learning (`svr`) wall functions have been developed
- Good results for channel flow placing the wall-adjacent cell at different locations
- Good results for developing boundary layer flow
- Training the `svr` with steady or instantaneous data: same results
- Training nearest neighbor (Python's `scipy.spatial.KDTree`) with instantaneous data: same results

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. 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.

[3] Andreas Lindholm, Niklas Wahlström, Fredrik Lindsten, and Thomas Schön. *Machine Learning: A First Course for Engineers and Scientists*. Cambridge University Press, 2022.

[4] Menneni Rachana, Jegadeesan Ramalingam, Gajula Ramana, Adigoppula Tejaswi, Sagar Mamidala, and G Srikanth. Fraud detection of credit card using machine learning. *GIS-Zeitschrift für Geoinformatik*, 8:1421–1436, 10 2021.

[5] Sudarshana S Rao and Santosh R Desai. Machine learning based traffic light detection and ir sensor based proximity sensing for autonomous cars. In *Proceedings of the International Conference on IoT Based Control Networks & Intelligent Systems*