

USING NEURAL NETWORK FOR IMPROVING AN EXPLICIT ALGEBRAIC STRESS MODEL IN 2D FLOWS [2]

Flow-Induced Acoustics Seminar 2024

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

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

- Machine learning (ML) is often a method where known data are used for teaching the algorithm to classify a set of data.
 - **Photographs** where the machine learning algorithm should recognize, e.g., traffic lights [8].

- Machine learning (ML) is often a method where known data are used for teaching the algorithm to classify a set of data.
 - **Photographs** where the machine learning algorithm should recognize, e.g., traffic lights [8].
 - **ECG signals** where the machine learning algorithm should recognize certain unhealthy conditions of the heart [6].

- Machine learning (ML) is often a method where known data are used for teaching the algorithm to classify a set of data.
 - **Photographs** where the machine learning algorithm should recognize, e.g., traffic lights [8].
 - **ECG signals** where the machine learning algorithm should recognize certain unhealthy conditions of the heart [6].
 - **Detecting fraud** for credit card payments [7].

- Machine learning (ML) is often a method where known data are used for teaching the algorithm to classify a set of data.
 - **Photographs** where the machine learning algorithm should recognize, e.g., traffic lights [8].
 - **ECG signals** where the machine learning algorithm should recognize certain unhealthy conditions of the heart [6].
 - **Detecting fraud** for credit card payments [7].
- In my case, input and output are **numerical** values.

- Machine learning (ML) is often a method where known data are used for teaching the algorithm to classify a set of data.
 - **Photographs** where the machine learning algorithm should recognize, e.g., traffic lights [8].
 - **ECG signals** where the machine learning algorithm should recognize certain unhealthy conditions of the heart [6].
 - **Detecting fraud** for credit card payments [7].
- In my case, input and output are **numerical** values.
- The ML will then be some form of **regression method**.

TARGET DATABASES

- At the previous slide, the targets for the Neural Network training process are:
 - Real **photographs** of, e.g., traffic signs

TARGET DATABASES

- At the previous slide, the targets for the Neural Network training process are:
 - Real **photographs** of, e.g., traffic signs
 - **ECG signals** of healthy hearts

TARGET DATABASES

- At the previous slide, the targets for the Neural Network training process are:
 - Real **photographs** of, e.g., traffic signs
 - **ECG signals** of healthy hearts
 - Normal **credit card** transactions

TARGET DATABASES

- At the previous slide, the targets for the Neural Network training process are:
 - Real **photographs** of, e.g., traffic signs
 - **ECG signals** of healthy hearts
 - Normal **credit card** transactions
- In this work I will use **DNS databases** of channel flow and flat-plate boundary layer flow

TARGET DATABASES

- At the previous slide, the targets for the Neural Network training process are:
 - Real **photographs** of, e.g., traffic signs
 - **ECG signals** of healthy hearts
 - Normal **credit card** transactions
- In this work I will use **DNS databases** of channel flow and flat-plate boundary layer flow
- The objective is to improve the Explicit Algebraic Reynolds Stress Model (EARSM)

The Algebraic Stress Model (ASM) with the LRR pressure-strain models [5] reads

$$\begin{aligned} \left(c_1 - 1 + P^k / \varepsilon \right) a_{ij} = & -\frac{8}{15} \bar{s}_{ij} + \frac{7c_2 + 1}{11} (a_{ik} \bar{\Omega}_{kj} - \bar{\Omega}_{ik} a_{kj}) \\ & - \frac{5 - 9c_2}{11} \left(a_{ik} \bar{s}_{kj} + \bar{s}_{ik} a_{kj} - \frac{2}{3} a_{mn} \bar{s}_{nm} \delta_{ij} \right), \quad a_{ij} = \frac{\overline{v'_i v'_j}}{k} - \frac{2}{3} \delta_{ij} \end{aligned} \quad (1)$$

The Algebraic Stress Model (ASM) with the LRR pressure-strain models [5] reads

$$\begin{aligned} \left(c_1 - 1 + P^k / \varepsilon \right) a_{ij} = & -\frac{8}{15} \bar{s}_{ij} + \frac{7c_2 + 1}{11} (a_{ik} \bar{\Omega}_{kj} - \bar{\Omega}_{ik} a_{kj}) \\ & - \frac{5 - 9c_2}{11} \left(a_{ik} \bar{s}_{kj} + \bar{s}_{ik} a_{kj} - \frac{2}{3} a_{mn} \bar{s}_{nm} \delta_{ij} \right), \quad a_{ij} = \frac{\overline{v'_i v'_j}}{k} - \frac{2}{3} \delta_{ij} \end{aligned} \quad (1)$$

Wallin & Johansson [9, 10] and Girimaji [3, 4] derived an explicit form which in 2D which reads

$$\begin{aligned} a_{ij} = & \beta_1 \bar{s}_{ij}^* + \beta_2 \left(\bar{s}_{ik}^* \bar{s}_{kj}^* - \frac{1}{3} \bar{s}_{mn}^* \bar{s}_{nm}^* \delta_{ij} \right) + \beta_4 \bar{s}_{ik}^* \left(\bar{\Omega}_{kj}^* - \bar{\Omega}_{ik}^* \bar{s}_{kj}^* \right) \end{aligned} \quad (2)$$

$$\bar{s}_{ij}^* = \frac{k}{\varepsilon} \bar{s}_{ij}, \quad \bar{\Omega}_{ij}^* = \frac{k}{\varepsilon} \bar{\Omega}_{ij}$$

The Algebraic Stress Model (ASM) with the LRR pressure-strain models [5] reads

$$\begin{aligned} \left(c_1 - 1 + P^k / \varepsilon \right) a_{ij} = & -\frac{8}{15} \bar{s}_{ij} + \frac{7c_2 + 1}{11} (a_{ik} \bar{\Omega}_{kj} - \bar{\Omega}_{ik} a_{kj}) \\ & - \frac{5 - 9c_2}{11} \left(a_{ik} \bar{s}_{kj} + \bar{s}_{ik} a_{kj} - \frac{2}{3} a_{mn} \bar{s}_{nm} \delta_{ij} \right), \quad a_{ij} = \frac{\overline{v'_i v'_j}}{k} - \frac{2}{3} \delta_{ij} \end{aligned} \quad (1)$$

Wallin & Johansson [9, 10] and Girimaji [3, 4] derived an explicit form which in 2D which reads

$$a_{ij} = \beta_1 \bar{s}_{ij}^* + \beta_2 \left(\bar{s}_{ik}^* \bar{s}_{kj}^* - \frac{1}{3} \bar{s}_{mn}^* \bar{s}_{nm}^* \delta_{ij} \right) + \beta_4 \bar{s}_{ik}^* \left(\bar{\Omega}_{kj}^* - \bar{\Omega}_{ik}^* \bar{s}_{kj}^* \right) \quad (2)$$

$$\bar{s}_{ij}^* = \frac{k}{\varepsilon} \bar{s}_{ij}, \quad \bar{\Omega}_{ij}^* = \frac{k}{\varepsilon} \bar{\Omega}_{ij}$$

Note that Eq. 1 is simplified if c_2 is set to 5/9.

EARSM: ANALYTICAL SOLUTION

Wallin & Johansson [9, 10] and Girimaji [3, 4] derived an analytical solution of Eq. 2.

EARSM: ANALYTICAL SOLUTION

Wallin & Johansson [9, 10] and Girimaji [3, 4] derived an analytical solution of Eq. 2.

In [10] it reads

$$\beta_1 = -\frac{A_1 N}{Q}, \quad \beta_2 = 2\frac{A_1 A_2}{Q}, \quad \beta_4 = -\frac{A_1}{Q}, \quad Q = N^2 - 2I_{\Omega} - \frac{2}{3}A_2^2 I_S$$

where N is given by the cubic equation

$$N^3 - A_3 N^2 \left(\left(A_1 A_4 + \frac{2}{3} A_2^2 \right) I_S + 2I_{\Omega} \right) N + 2A_3 \left(\frac{1}{3} A_2^2 I_S + I_{\Omega} \right) = 0 \quad (3)$$

where $I_S = \bar{s}_{mn}^* \bar{s}_{nm}^*$ and $I_{\Omega} = \bar{\Omega}_{mn}^* \bar{\Omega}_{nm}^*$.

EARSM: ANALYTICAL SOLUTION

Wallin & Johansson [9, 10] and Girimaji [3, 4] derived an analytical solution of Eq. 2.

In [10] it reads

$$\beta_1 = -\frac{A_1 N}{Q}, \quad \beta_2 = 2\frac{A_1 A_2}{Q}, \quad \beta_4 = -\frac{A_1}{Q}, \quad Q = N^2 - 2I_{\Omega} - \frac{2}{3}A_2^2 I_S$$

where N is given by the cubic equation

$$N^3 - A_3 N^2 \left(\left(A_1 A_4 + \frac{2}{3} A_2^2 \right) I_S + 2I_{\Omega} \right) N + 2A_3 \left(\frac{1}{3} A_2^2 I_S + I_{\Omega} \right) = 0 \quad (3)$$

where $I_S = \bar{s}_{mn}^* \bar{s}_{nm}^*$ and $I_{\Omega} = \bar{\Omega}_{mn}^* \bar{\Omega}_{nm}^*$.

Equation 3 can be solved analytically.

EARSM: NEURAL NETWORK (NN)

Instead of using the expression on the previous slide for $\beta_1, \beta_2, \beta_4$, let's make them functions of something using NN.

$$a_{ij} = \beta_1 \bar{s}_{ij}^* + \beta_2 \left(\bar{s}_{ik}^* \bar{s}_{kj}^* - \frac{1}{3} \bar{s}_{mn}^* \bar{s}_{nm}^* \delta_{ij} \right) + \beta_4 \bar{s}_{ik}^* \left(\bar{\Omega}_{kj}^* - \bar{\Omega}_{ik}^* \bar{s}_{kj}^* \right), \quad a_{ij} = \frac{\overline{v_i' v_j'}}{k} - \frac{2}{3} \delta_{ij}$$

EARSM: NEURAL NETWORK (NN)

Instead of using the expression on the previous slide for $\beta_1, \beta_2, \beta_4$, let's make them functions of something using NN.

$$a_{ij} = \beta_1 \bar{s}_{ij}^* + \beta_2 \left(\bar{s}_{ik}^* \bar{s}_{kj}^* - \frac{1}{3} \bar{s}_{mn}^* \bar{s}_{nm}^* \delta_{ij} \right) + \beta_4 \bar{s}_{ik}^* \left(\bar{\Omega}_{kj}^* - \bar{\Omega}_{ik}^* \bar{s}_{kj}^* \right), \quad a_{ij} = \frac{\overline{v_i' v_j'}}{k} - \frac{2}{3} \delta_{ij}$$

- Train the NN model in fully-developed channel flow

EARSM: NEURAL NETWORK (NN)

Instead of using the expression on the previous slide for $\beta_1, \beta_2, \beta_4$, let's make them functions of something using NN.

$$a_{ij} = \beta_1 \bar{s}_{ij}^* + \beta_2 \left(\bar{s}_{ik}^* \bar{s}_{kj}^* - \frac{1}{3} \bar{s}_{mn}^* \bar{s}_{nm}^* \delta_{ij} \right) + \beta_4 \bar{s}_{ik}^* \left(\bar{\Omega}_{kj}^* - \bar{\Omega}_{ik}^* \bar{s}_{kj}^* \right), \quad a_{ij} = \frac{\overline{v_i' v_j'}}{k} - \frac{2}{3} \delta_{ij}$$

- Train the NN model in fully-developed channel flow
- Output variables: $\beta_1, \beta_2, \beta_4$

EARSM: NEURAL NETWORK (NN)

Instead of using the expression on the previous slide for $\beta_1, \beta_2, \beta_4$, let's make them functions of something using NN.

$$a_{ij} = \beta_1 \bar{s}_{ij}^* + \beta_2 \left(\bar{s}_{ik}^* \bar{s}_{kj}^* - \frac{1}{3} \bar{s}_{mn}^* \bar{s}_{nm}^* \delta_{ij} \right) + \beta_4 \bar{s}_{ik}^* \left(\bar{\Omega}_{kj}^* - \bar{\Omega}_{ik}^* \bar{s}_{kj}^* \right), \quad a_{ij} = \frac{\overline{v_i' v_j'}}{k} - \frac{2}{3} \delta_{ij}$$

- Train the NN model in fully-developed channel flow
- Output variables: $\beta_1, \beta_2, \beta_4$
- I will use NN in Python's `pytorch`

EARSM IN FULLY-DEVELOPED CHANNEL FLOW

- I will train NN in fully-developed channel flow

EARSM IN FULLY-DEVELOPED CHANNEL FLOW

- I will train NN in fully-developed channel flow
- Then the EARSM (Eq. 2) reads:

EARSM IN FULLY-DEVELOPED CHANNEL FLOW

- I will train NN in fully-developed channel flow
- Then the EARSM (Eq. 2) reads:

$$a_{11} = \frac{1}{12} \left(\frac{\partial \bar{v}_1^*}{\partial x_2} \right)^2 (\beta_2 - 6\beta_4), \quad a_{22} = \frac{1}{12} \left(\frac{\partial \bar{v}_1^*}{\partial x_2} \right)^2 (\beta_2 + 6\beta_4)$$
$$a_{33} = -\frac{2\beta_2}{12} \left(\frac{\partial \bar{v}_1^*}{\partial x_2} \right)^2, \quad a_{12} = \frac{\beta_1}{2} \frac{\partial \bar{v}_1^*}{\partial x_2}, \quad \frac{\partial \bar{v}_1^*}{\partial x_2} = \frac{k}{\epsilon} \frac{\partial \bar{v}_1}{\partial x_2}$$

EARSM IN FULLY-DEVELOPED CHANNEL FLOW

- I will train NN in fully-developed channel flow
- Then the EARSM (Eq. 2) reads:

$$a_{11} = \frac{1}{12} \left(\frac{\partial \bar{v}_1^*}{\partial x_2} \right)^2 (\beta_2 - 6\beta_4), \quad a_{22} = \frac{1}{12} \left(\frac{\partial \bar{v}_1^*}{\partial x_2} \right)^2 (\beta_2 + 6\beta_4)$$
$$a_{33} = -\frac{2\beta_2}{12} \left(\frac{\partial \bar{v}_1^*}{\partial x_2} \right)^2, \quad a_{12} = \frac{\beta_1}{2} \frac{\partial \bar{v}_1^*}{\partial x_2}, \quad \frac{\partial \bar{v}_1^*}{\partial x_2} = \frac{k}{\epsilon} \frac{\partial \bar{v}_1}{\partial x_2}$$

- Find $\beta_1, \beta_2, \beta_4$ (targets for NN and computed from DNS):

EARSM IN FULLY-DEVELOPED CHANNEL FLOW

- I will train NN in fully-developed channel flow
- Then the EARSM (Eq. 2) reads:

$$a_{11} = \frac{1}{12} \left(\frac{\partial \bar{v}_1^*}{\partial x_2} \right)^2 (\beta_2 - 6\beta_4), \quad a_{22} = \frac{1}{12} \left(\frac{\partial \bar{v}_1^*}{\partial x_2} \right)^2 (\beta_2 + 6\beta_4)$$
$$a_{33} = -\frac{2\beta_2}{12} \left(\frac{\partial \bar{v}_1^*}{\partial x_2} \right)^2, \quad a_{12} = \frac{\beta_1}{2} \frac{\partial \bar{v}_1^*}{\partial x_2}, \quad \frac{\partial \bar{v}_1^*}{\partial x_2} = \frac{k}{\epsilon} \frac{\partial \bar{v}_1}{\partial x_2}$$

- Find $\beta_1, \beta_2, \beta_4$ (targets for NN and computed from DNS):

$$\beta_1 = \frac{2a_{12}}{\frac{\partial \bar{v}_1^*}{\partial x_2}}, \quad \beta_2 = \frac{6(a_{11} + a_{22})}{\left(\frac{\partial \bar{v}_1^*}{\partial x_2} \right)^2}, \quad \beta_4 = \frac{a_{22} - a_{11}}{\left(\frac{\partial \bar{v}_1^*}{\partial x_2} \right)^2}$$

EARSM, CHANNEL FLOW, $Re_\tau = 2\,000$, $Re_\tau = 5\,200$, $Re_\tau = 10\,000$

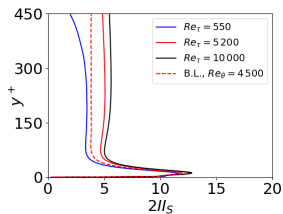
- Input variables?

EARSM, CHANNEL FLOW, $Re_\tau = 2\,000$, $Re_\tau = 5\,200$, $Re_\tau = 10\,000$

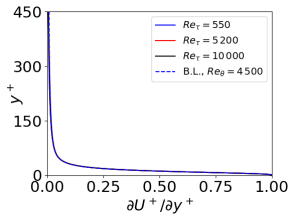
- Input variables? $II_S = \frac{1}{2} \left(\frac{k}{\varepsilon} \frac{\partial \bar{v}_1}{\partial x_2} \right)^2$, $N = P^k / \varepsilon$, $\frac{\partial U^+}{\partial y^+}$, P^{k+} and y^+

EARSM, CHANNEL FLOW, $Re_\tau = 2\,000$, $Re_\tau = 5\,200$, $Re_\tau = 10\,000$

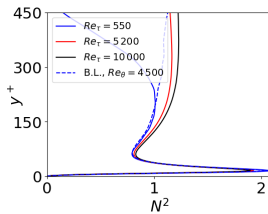
- Input variables? $II_S = \frac{1}{2} \left(\frac{k}{\varepsilon} \frac{\partial \bar{v}_1}{\partial x_2} \right)^2$, $N = P^k / \varepsilon$, $\frac{\partial U^+}{\partial y^+}$, P^{k+} and y^+



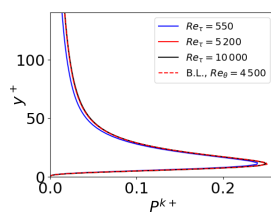
(A) Velocity gradient scaled with k and $\tilde{\varepsilon}$.



(B) Velocity gradient scaled with u_τ and ν .



(C) Ratio of P^k to $\tilde{\varepsilon}$.



(D) Production scaled with u_τ and ν .

FIGURE: DNS data. $\tilde{\varepsilon} = \varepsilon - \nu \partial^2 k / \partial y^2$.

EARSM, CHANNEL FLOW, $Re_\tau = 2\,000$, $Re_\tau = 5\,200$, $Re_\tau = 10\,000$

- I choose P^{k+} and y^+ as input parameters.

EARSM, CHANNEL FLOW, $Re_\tau = 2\,000$, $Re_\tau = 5\,200$, $Re_\tau = 10\,000$

- I choose P^{k+} and y^+ as input parameters.
- The velocity gradient, $\frac{\partial U^+}{\partial y^+}$, could be an option

EARSM, CHANNEL FLOW, $Re_\tau = 2\,000$, $Re_\tau = 5\,200$, $Re_\tau = 10\,000$

- I choose P^{k+} and y^+ as input parameters.
- The velocity gradient, $\frac{\partial U^+}{\partial y^+}$, could be an option
- But it was found that the NN training process does not converge. Probably because of the large gradients near the wall.

EARSM, CHANNEL FLOW, $Re_\tau = 2\,000$, $Re_\tau = 5\,200$, $Re_\tau = 10\,000$

- I choose P^{k+} and y^+ as input parameters.
- The velocity gradient, $\frac{\partial U^+}{\partial y^+}$, could be an option
- But it was found that the NN training process does not converge. Probably because of the large gradients near the wall.
- I scale the two input parameters using `MinMaxScaler()` so that they are in the range $[0, 1]$

NEURAL NETWORK (NN). PYTHON'S `PYTORCH`

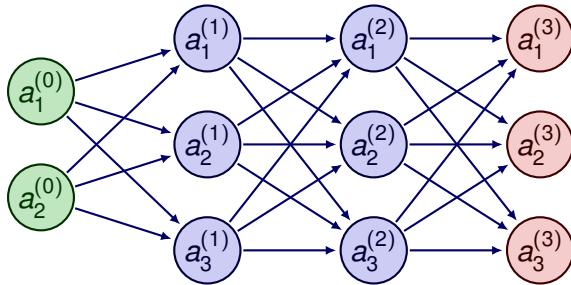
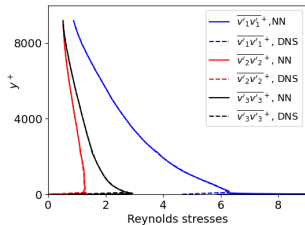


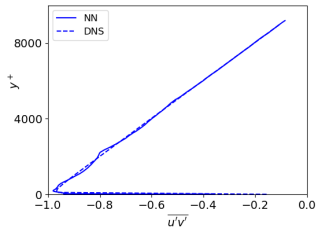
FIGURE: The Neural Network with two input variables, $a_1^{(0)} = y^+$ and $a_2^{(0)} = P^{k+}$ and three output variables, $a_1^{(3)} = \beta_1$, $a_2^{(3)} = \beta_2$ and $a_3^{(3)} = \beta_4$. There are three neurons in this figure; in the simulations I have 50.

PYTHON'S PYTORCH. TRAINING & PREDICTING

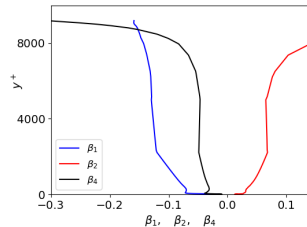
- I train the NN model using DNS data of channel flow at $Re_\tau = 10\,000$.
- I exclude data in the viscous sublayer ($y^+ \leq 5$) and near the center ($y^+ \geq 9\,800$) where $\frac{\partial \bar{v}_1^*}{\partial x_2}$ goes to zero.
- I train on 80% of the data (approx 800 randomly selected DNS data points) and test (predict) on the remaining 20%.



(A) Reynolds normal stresses.



(B) Reynolds shear stress.



(C) Predicted EARSM coefficients.

FIGURE: Predicted with NN. $Re_\tau = 10\,000$. Trained on $Re_\tau = 10\,000$.

- The **Python** finite volume code **pyCALC-RANS** [1] is used.

CFD SOLVER

- The **Python** finite volume code **pyCALC-RANS** [1] is used.
- Fully vectorized (i.e. no `for` loops).

- The **Python** finite volume code **pyCALC-RANS** [1] is used.
- Fully vectorized (i.e. no `for` loops).
- SIMPLEC and Wilcox $k - \omega$ model

- The **Python** finite volume code **pyCALC-RANS** [1] is used.
- Fully vectorized (i.e. no `for` loops).
- SIMPLEC and Wilcox $k - \omega$ model
- Discretization: Hybrid first-order upwind/second-order central differencing

- The **Python** finite volume code **pyCALC-RANS** [1] is used.
- Fully vectorized (i.e. no `for` loops).
- SIMPLEC and Wilcox $k - \omega$ model
- Discretization: Hybrid first-order upwind/second-order central differencing
- The discretized equations are solved with **Python sparse matrix solvers**.

STANDARD $k - \omega$ MODEL

$$\begin{aligned}\bar{\nu}_j \frac{\partial k}{\partial x_j} &= P^k + \frac{\partial}{\partial x_j} \left[\left(\nu + \frac{\nu_t}{\sigma_k} \right) \frac{\partial k}{\partial x_j} \right] - C_\mu k \omega \\ \bar{\nu}_j \frac{\partial \omega}{\partial x_j} &= \alpha \frac{\omega}{k} P^k + \frac{\partial}{\partial x_j} \left[\left(\nu + \frac{\nu_t}{\sigma_\omega} \right) \frac{\partial \omega}{\partial x_j} \right] - \beta \omega^2 \\ \nu_t &= \frac{k}{\omega}\end{aligned}\tag{4}$$

INCLUDING NN PREDICTION IN THE CFD SOLVER

- 1 Load the NN model into **pyCALC-RANS**

INCLUDING NN PREDICTION IN THE CFD SOLVER

- ① Load the NN model into **pyCALC-RANS**
- ② Solve U , V and PP equations. The Reynolds stresses $\overline{v_1'^2}$, $\overline{v_2'^2}$, $\overline{v_1'v_2'}$ in the U and V equations are taken from previous iteration

INCLUDING NN PREDICTION IN THE CFD SOLVER

- ① Load the NN model into **pyCALC-RANS**
- ② Solve U , V and PP equations. The Reynolds stresses $\overline{v_1'^2}$, $\overline{v_2'^2}$, $\overline{v_1'v_2'}$ in the U and V equations are taken from previous iteration
- ③ Compute β_1 , β_2 , β_4 using NN. Use limits from training data.

INCLUDING NN PREDICTION IN THE CFD SOLVER

- ① Load the NN model into **pyCALC-RANS**
- ② Solve U , V and PP equations. The Reynolds stresses $\overline{v_1'^2}$, $\overline{v_2'^2}$, $\overline{v_1'v_2'}$ in the U and V equations are taken from previous iteration
- ③ Compute β_1 , β_2 , β_4 using NN. Use limits from training data.
- ④ Compute the anisotropic Reynolds stresses (a_{11} , a_{22} , a_{12}) using β_1 , β_2 , β_4

INCLUDING NN PREDICTION IN THE CFD SOLVER

- 1 Load the NN model into **pyCALC-RANS**
- 2 Solve U , V and PP equations. The Reynolds stresses $\overline{v_1'^2}$, $\overline{v_2'^2}$, $\overline{v_1'v_2'}$ in the U and V equations are taken from previous iteration
- 3 Compute β_1 , β_2 , β_4 using NN. Use limits from training data.
- 4 Compute the anisotropic Reynolds stresses (a_{11} , a_{22} , a_{12}) using β_1 , β_2 , β_4

5

$$\overline{v_1'^2} = ka_{11} + \frac{2}{3}k, \quad \overline{v_2'^2} = ka_{22} + \frac{2}{3}k, \quad \overline{v_1'v_2'} = ka_{12}$$

INCLUDING NN PREDICTION IN THE CFD SOLVER

- 1 Load the NN model into **pyCALC-RANS**
- 2 Solve U , V and PP equations. The Reynolds stresses $\overline{v_1'^2}$, $\overline{v_2'^2}$, $\overline{v_1'v_2'}$ in the U and V equations are taken from previous iteration
- 3 Compute β_1 , β_2 , β_4 using NN. Use limits from training data.
- 4 Compute the anisotropic Reynolds stresses (a_{11} , a_{22} , a_{12}) using β_1 , β_2 , β_4

5

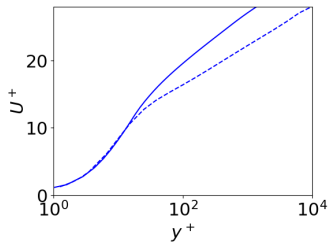
$$\overline{v_1'^2} = ka_{11} + \frac{2}{3}k, \quad \overline{v_2'^2} = ka_{22} + \frac{2}{3}k, \quad \overline{v_1'v_2'} = ka_{12}$$

- 6 Solve k and ω equations. The Reynolds stresses $\overline{v_1'^2}$, $\overline{v_2'^2}$, $\overline{v_1'v_2'}$ are used in the production terms

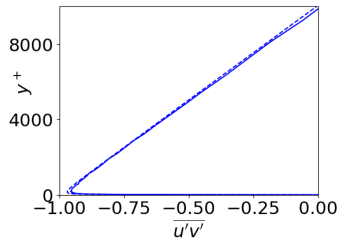
INCLUDING NN PREDICTION IN THE CFD SOLVER

- 1 Load the NN model into **pyCALC-RANS**
- 2 Solve U , V and PP equations. The Reynolds stresses $\overline{v_1'^2}$, $\overline{v_2'^2}$, $\overline{v_1'v_2'}$ in the U and V equations are taken from previous iteration
- 3 Compute β_1 , β_2 , β_4 using NN. Use limits from training data.
- 4 Compute the anisotropic Reynolds stresses (a_{11} , a_{22} , a_{12}) using β_1 , β_2 , β_4
- 5
$$\overline{v_1'^2} = ka_{11} + \frac{2}{3}k, \quad \overline{v_2'^2} = ka_{22} + \frac{2}{3}k, \quad \overline{v_1'v_2'} = ka_{12}$$
- 6 Solve k and ω equations. The Reynolds stresses $\overline{v_1'^2}$, $\overline{v_2'^2}$, $\overline{v_1'v_2'}$ are used in the production terms
- 7 End of global iteration. Repeat from Step 2 until convergence (1000s of iterations)

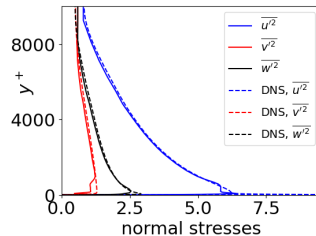
CFD & NN. $Re_\tau = 10\,000$. NN TRAINED ON $Re_\tau = 10\,000$



(A) Velocity.

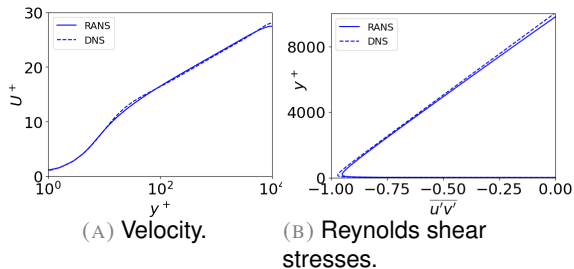


(B) Reynolds shear stresses.



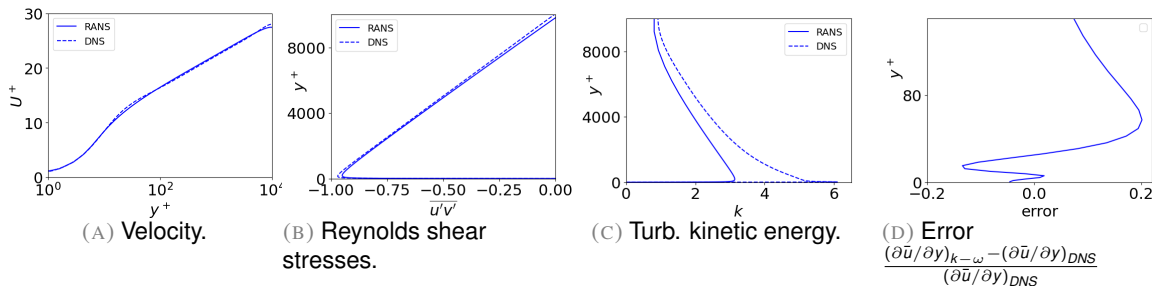
(C) Reynolds normal stresses.

WHY SUCH POOR PREDICTIONS? LET'S LOOK AT $k - \omega$ PREDICTIONS



- Velocity and shear stress are (or seems to be) well predicted

WHY SUCH POOR PREDICTIONS? LET'S LOOK AT $k - \omega$ PREDICITONS



- Velocity and shear stress are (or seems to be) well predicted
- Both k and the velocity gradient are poorly predicted

LET'S TRAIN NN WITH $k - \omega$ DATA AND DNS AT $Re_\tau = 10\,000$

- I use NN with
 - Input: P^k and y^+ from $k - \omega$ prediction
 - Target: β_1, β_2 and β_4 from $\underbrace{\overline{v_1'^2}, \overline{v_2'^2}}_{DNS}$ and $\underbrace{\overline{v_1' v_2'}}_{k-\omega}, k, \varepsilon$
- Note that $k_{DNS} = 0.5 \left(\overline{v_1'^2}_{DNS} + \overline{v_2'^2}_{DNS} + \overline{v_3'^2}_{DNS} \right) \neq k_{k-\omega}$

LET'S TRAIN NN WITH $k - \omega$ DATA AND DNS AT $Re_\tau = 10\,000$

- I use NN with
 - Input: P^k and y^+ from $k - \omega$ prediction
 - Target: β_1, β_2 and β_4 from $\underbrace{\overline{v_1'^2}, \overline{v_2'^2}}_{DNS}$ and $\underbrace{\overline{v_1' v_2'}}_{k-\omega}, k, \varepsilon$
- Note that $k_{DNS} = 0.5 \left(\overline{v_1'^2}_{DNS} + \overline{v_2'^2}_{DNS} + \overline{v_3'^2}_{DNS} \right) \neq k_{k-\omega}$
- $\overline{v_3'^2}$ predicted by NN will give $k_{k-\omega} = 0.5 \left(\overline{v_1'^2}_{DNS} + \overline{v_2'^2}_{DNS} + \overline{v_3'^2}_{DNS} \right)$

LET'S TRAIN NN WITH $k - \omega$ DATA AND DNS AT $Re_\tau = 10\,000$

- I use NN with
 - Input: P^k and y^+ from $k - \omega$ prediction
 - Target: β_1, β_2 and β_4 from $\underbrace{\overline{v_1'^2}, \overline{v_2'^2}}_{DNS}$ and $\underbrace{\overline{v_1' v_2'}}_{k-\omega}, k, \varepsilon$
- Note that $k_{DNS} = 0.5 \left(\overline{v_1'^2}_{DNS} + \overline{v_2'^2}_{DNS} + \overline{v_3'^2}_{DNS} \right) \neq k_{k-\omega}$
- $\overline{v_3'^2}$ predicted by NN will give $k_{k-\omega} = 0.5 \left(\overline{v_1'^2}_{DNS} + \overline{v_2'^2}_{DNS} + \overline{v_3'^2}_{DNS} \right)$
- \Rightarrow wrong $\overline{v_3'^2}$ (even negative near the wall)

LET'S TRAIN NN WITH $k - \omega$ DATA AND DNS AT $Re_\tau = 10\,000$

- I use NN with
 - Input: P^k and y^+ from $k - \omega$ prediction
 - Target: β_1, β_2 and β_4 from $\underbrace{\overline{v_1'^2}, \overline{v_2'^2}}_{DNS}$ and $\underbrace{\overline{v_1' v_2'}}_{k-\omega}, k, \varepsilon$
- Note that $k_{DNS} = 0.5 \left(\overline{v_1'^2}_{DNS} + \overline{v_2'^2}_{DNS} + \overline{v_3'^2}_{DNS} \right) \neq k_{k-\omega}$
- $\overline{v_3'^2}$ predicted by NN will give $k_{k-\omega} = 0.5 \left(\overline{v_1'^2}_{DNS} + \overline{v_2'^2}_{DNS} + \overline{v_3'^2}_{DNS} \right)$
- \Rightarrow wrong $\overline{v_3'^2}$ (even negative near the wall)
- but in 2D $\overline{v_3'^2}$ is not used

LET'S TRAIN NN WITH $k - \omega$ DATA AND DNS AT $Re_\tau = 10\,000$

- I use NN with
 - Input: P^k and y^+ from $k - \omega$ prediction
 - Target: β_1, β_2 and β_4 from $\underbrace{\overline{v_1'^2}, \overline{v_2'^2}}_{DNS}$ and $\underbrace{\overline{v_1' v_2'}}_{k-\omega}, k, \varepsilon$
- Note that $k_{DNS} = 0.5 \left(\overline{v_1'^2}_{DNS} + \overline{v_2'^2}_{DNS} + \overline{v_3'^2}_{DNS} \right) \neq k_{k-\omega}$
- $\overline{v_3'^2}$ predicted by NN will give $k_{k-\omega} = 0.5 \left(\overline{v_1'^2}_{DNS} + \overline{v_2'^2}_{DNS} + \overline{v_3'^2}_{DNS} \right)$
- \Rightarrow wrong $\overline{v_3'^2}$ (even negative near the wall)
- but in 2D $\overline{v_3'^2}$ is not used
- In order to make this approach applicable in 3D:

LET'S TRAIN NN WITH $k - \omega$ DATA AND DNS AT $Re_\tau = 10\,000$

- I use NN with
 - Input: P^k and y^+ from $k - \omega$ prediction
 - Target: β_1, β_2 and β_4 from $\underbrace{\overline{v_1'^2}, \overline{v_2'^2}}_{DNS}$ and $\underbrace{\overline{v_1' v_2'}, k, \varepsilon}_{k-\omega}$
- Note that $k_{DNS} = 0.5 \left(\overline{v_1'^2}_{DNS} + \overline{v_2'^2}_{DNS} + \overline{v_3'^2}_{DNS} \right) \neq k_{k-\omega}$
- $\overline{v_3'^2}$ predicted by NN will give $k_{k-\omega} = 0.5 \left(\overline{v_1'^2}_{DNS} + \overline{v_2'^2}_{DNS} + \overline{v_3'^2}_{DNS} \right)$
- \Rightarrow wrong $\overline{v_3'^2}$ (even negative near the wall)
- but in 2D $\overline{v_3'^2}$ is not used
- In order to make this approach applicable in 3D:
 - develop a new $k - \omega$ model that accurately predicts k

RESULTS

- I will compare two models.
 - NN EARSM with $k - \omega$ trained in channel flow at $Re_\tau = 10\,000$
 - Standard EARSM with $k - \omega$
- Four flows
 - Channel flow at $Re_\tau = 10\,000$, $Re_\tau = 5\,200$, and $Re_\tau = 2\,000$
 - Flat-plate boundary layer, $Re_\theta = 5\,500$.

CHANNEL FLOW, $Re_\tau = 10\,000$

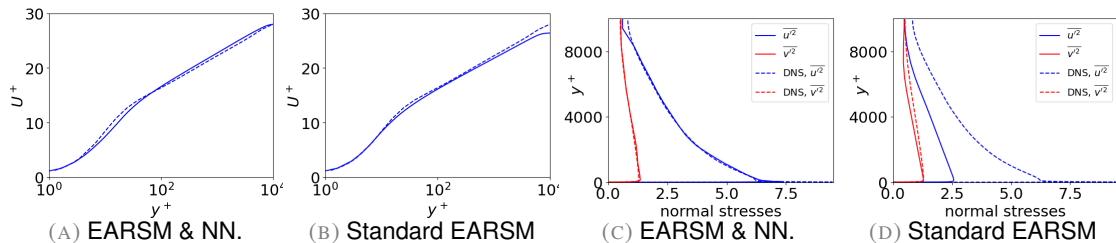


FIGURE: CFD predictions with NN and standard EARSM at $Re_\tau = 10\,000$.

CHANNEL FLOW, $Re_\tau = 5\,200$

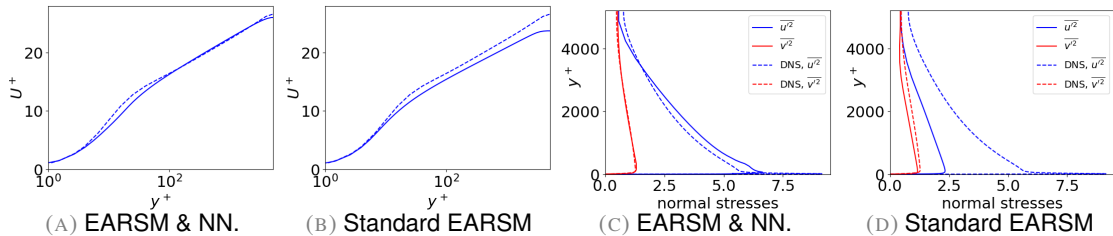


FIGURE: CFD predictions with NN and standard EARSM at $Re_\tau = 5\,200$.

CHANNEL FLOW, $Re_\tau = 2\,000$

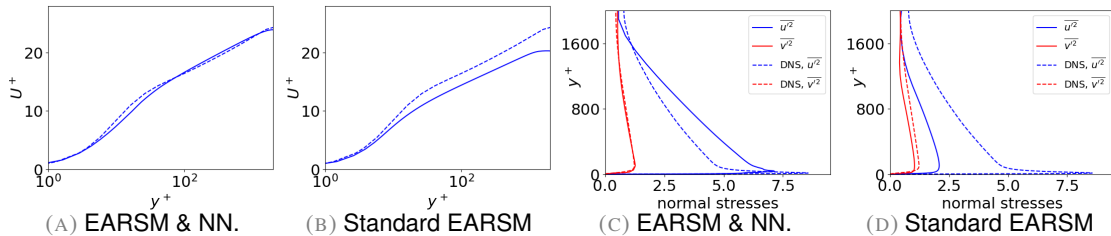
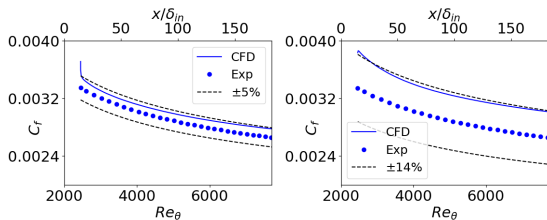


FIGURE: CFD predictions with NN and standard EARSM at $Re_\tau = 2\,000$.

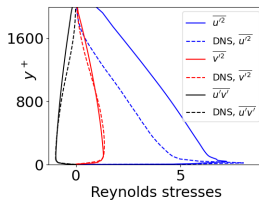
FLAT-PLATE BOUNDARY LAYER, $Re_\theta = 5\,500$.

- Inlet b.c. taken from a pre-cursor $k - \omega$ simulation at $Re_\theta \simeq 2\,500$

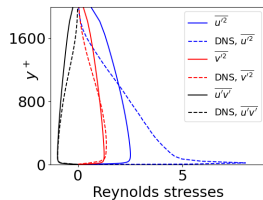


(A) EARSIM & NN.

(B) Standard EARSIM



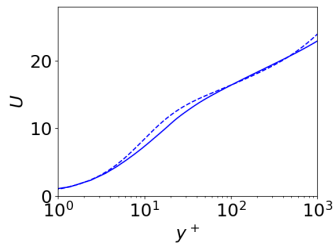
(C) EARSIM & NN.



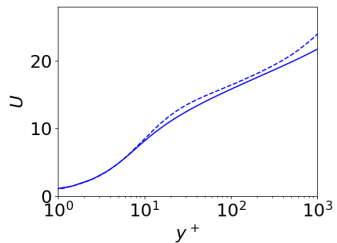
(D) Standard EARSIM

FIGURE: CFD predictions with NN at $Re_\theta = 5\,500$.

FLAT-PLATE BOUNDARY LAYER, $Re_\theta = 5\,500$.



(A) EARSM & NN.



(B) Standard EARSM

FIGURE: CFD predictions with NN at $Re_\theta = 5\,500$.

CONCLUSIONS

- A new NN EARSIM model has been presented

CONCLUSIONS

- A new NN EARSIM model has been presented
- It is incorporated into a CFD solver and coupled to the Wilcox $k - \omega$ model

CONCLUSIONS

- A new NN EARSIM model has been presented
- It is incorporated into a CFD solver and coupled to the Wilcox $k - \omega$ model
- It is found that the NN model cannot be trained with only DNS data

CONCLUSIONS

- A new NN EARSM model has been presented
- It is incorporated into a CFD solver and coupled to the Wilcox $k - \omega$ model
- It is found that the NN model cannot be trained with only DNS data
- Input, channel flow. $Re_\tau = 10\,000$: P^k and y^+ from $k - \omega$ prediction

CONCLUSIONS

- A new NN EARSIM model has been presented
- It is incorporated into a CFD solver and coupled to the Wilcox $k - \omega$ model
- It is found that the NN model cannot be trained with only DNS data
- Input, channel flow. $Re_\tau = 10\,000$: P^k and y^+ from $k - \omega$ prediction
- Target, channel flow. $Re_\tau = 10\,000$: $\underbrace{\overline{v_1'^2}, \overline{v_2'^2}}_{DNS}$ and $\underbrace{\overline{v_1' v_2'}, k, \varepsilon}_{k-\omega}$

CONCLUSIONS

- A new NN EARSM model has been presented
- It is incorporated into a CFD solver and coupled to the Wilcox $k - \omega$ model
- It is found that the NN model cannot be trained with only DNS data
- Input, channel flow. $Re_\tau = 10\,000$: P^k and y^+ from $k - \omega$ prediction
- Target, channel flow. $Re_\tau = 10\,000$: $\underbrace{\overline{v_1'^2}, \overline{v_2'^2}}_{DNS}$ and $\underbrace{\overline{v_1' v_2'}, k, \varepsilon}_{k-\omega}$
- Good results are obtained for channel flow at $Re_\tau = 2\,000, 5\,200, 10\,000$ and flat-plate boundary layer (better than the standard EARSM)


CONCLUSIONS

- A new NN EARSM model has been presented
- It is incorporated into a CFD solver and coupled to the Wilcox $k - \omega$ model
- It is found that the NN model cannot be trained with only DNS data
- Input, channel flow. $Re_\tau = 10\,000$: P^k and y^+ from $k - \omega$ prediction
- Target, channel flow. $Re_\tau = 10\,000$: $\underbrace{\overline{v_1'^2}, \overline{v_2'^2}}_{DNS}$ and $\underbrace{\overline{v_1' v_2'}, k, \varepsilon}_{k-\omega}$
- Good results are obtained for channel flow at $Re_\tau = 2\,000, 5\,200, 10\,000$ and flat-plate boundary layer (better than the standard EARSM)
- Python scripts can be downloaded [here](#)

CONCLUSIONS

- A new NN EARSM model has been presented
- It is incorporated into a CFD solver and coupled to the Wilcox $k - \omega$ model
- It is found that the NN model cannot be trained with only DNS data
- Input, channel flow. $Re_\tau = 10\,000$: P^k and y^+ from $k - \omega$ prediction
- Target, channel flow. $Re_\tau = 10\,000$: $\underbrace{\overline{v_1'^2}, \overline{v_2'^2}}_{DNS}$ and $\underbrace{\overline{v_1' v_2'}, k, \varepsilon}_{k-\omega}$
- Good results are obtained for channel flow at $Re_\tau = 2\,000, 5\,200, 10\,000$ and flat-plate boundary layer (better than the standard EARSM)
- Python scripts can be downloaded [here](#)
- Assignment on NN for improving EARSM can be found here [here](#)

REFERENCES

- [1] L. Davidson. pyCALC-RANS: a 2D Python code for RANS . Division of Fluid Dynamics, Dept. of Mechanics and Maritime Sciences, Chalmers University of Technology, Gothenburg
Download the code [here](#), 2021.
- [2] L. Davidson. Using neural network for improving an explicit algebraic stress model in 2D flow (to appear). In J. C. Tyacke and N. R. Vadlamani, editors, *CUSF 2024, Proceedings of the Cambridge Unsteady Flow Symposium 2024*. Springer Nature Switzerland AG, 2024.
- [3] S. S. Girimaji. Fully-explicit and self-consistent algebraic Reynolds stress model. *Theoretical and Computational Fluid Dynamics*, 8:387–402, 1996.
- [4] S.S. Girimaji. Fully-explicit and self-consistent algebraic Reynolds stress model. ICASE Rep. 95-82, Institute for Computer Applications in Science and Engineering NASA Langley Research Center, Hampton, VA, USA, 1995.

REFERENCES

- [5] B. E. Launder, G. J. Reece, and W. Rodi. Progress in the development of a Reynolds-stress turbulence closure. *Journal of Fluid Mechanics*, 68(3):537–566, 1975.
- [6] Andreas Lindholm, Niklas Wahlström, Fredrik Lindsten, and Thomas Schön. *Machine Learning: A First Course for Engineers and Scientists*. Cambridge University Press, 2022.
- [7] 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.
- [8] 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 – ICICNIS*, 2021.

REFERENCES

- [9] S. Wallin and A.V. Johansson. A new explicit algebraic Reynolds stress turbulence model including an improved near-wall treatment. In C.-J. Chen, C. Shih, J. Lienau, and R. J. Kung, editors, *Proc. Flow Modeling and Turbulence Measurements VI, Tallahassee F. L.*, pages 399–406. Balkema, 1996.
- [10] S. Wallin and A.V. Johansson. A new explicit algebraic Reynolds stress model for incompressible and compressible turbulent flows. *Journal of Fluid Mechanics*, 403:89–132, 2000.