Using Neural Network for Improving an Explicit Algebraic Stress Model in 2D Flow

Lars Davidson

Abstract Neural Network (NN) is used to improve an Explicit Algebraic Reynolds Stress Model (EARSM). The turbulent kinetic energy and its dissipation are predicted using the standard $k-\omega$ model. The NN model is trained in channel flow of $Re_\tau = 10{,}000$. The NN model is stored to disk and subsequently loaded into the CFD code. The NN model is called every iteration to compute the β coefficients in the EARSM, i.e. the CFD solver and the NN model are fully coupled. The Reynolds stresses are used in the momentum equations and the production term in the k and ω equations. It is found that when training the NN model, the target data cannot only be taken from DNS. The reason is that the stress-strain relation and the turbulent kinetic energy of the DNS data are different from those of the $k-\omega$ model. Hence, the target data are taken both from DNS and a $k-\omega$ simulation. The new EARSM-NN model is used for predicting channel flow at $Re_\tau = 2000$, 5200 and 10,000 and flat-plate boundary layer at $2500 \leq Re_\theta \leq 8000$. The EARSM-NN model gives much better results than the standard EARSM.

1 Introduction

The relation between strain rate and Reynolds stresses in two-equation eddy-viscosity turbulence models is usually linear (the Boussinesq assumption). Non-linear turbulence models were developed in the 1990s, see e.g. [1]. These non-linear models are based on a subset of the generalized nonlinear form given by Pope [2] which is expressed in ten tensors, T_{ij}^n and five invariants, i.e.

$$a_{ij} = \sum_{n=1}^{10} G^{(n)} T_{ij}^n, \quad a_{ij} = \overline{v_i' v_j'} - \frac{2}{3} k \delta_{ij} \tag{1}$$

CUFS-2024-PAPER ID, Cambridge, United Kingdom, 4-5 March 2024.

L. Davidson (✉)
Division of Fluid Dynamics, Department of Mechanics and Maritime Sciences,
Chalmers University of Technology, 412 96 Gothenburg, Sweden
e-mail: lada@chalmers.se

J. C. Tyacke and N. R. Vadlamani (eds.), *Proceedings of the Cambridge Unsteady Flow Symposium 2024*, https://doi.org/10.1007/978-3-031-69035-8_2

where a_{ij} is the anisotropy tensor. Ling *et al.* [3] seem to be the first who used neural networks (NN) and the generalized nonlinear for predicting the turbulent Reynolds stress. They use a general non-linear Reynolds stress model in which the stresses are expressed by Eq. (1). They trained their neural network using the five invariants as input and the 10 tensors as output. Six cases were used for training, namely duct flow, channel flow, a jet in cross-flow, an inclined jet in cross-flow, flow around a square cylinder and flow through a converging-diverging channel. However, when they include their NN model into their CFD solver, they apply the NN model only once and inject the NN stresses and then keep them frozen (sometimes called the frozen substitution [4]).

The objective of [5] was also to develop a model for the turbulent stress tensor, $\overline{v_i' v_j'}$. Contrary to the work by [3], they include more influence parameters. They include also the gradients of pressure and turbulent kinetic energy, i.e.

$$b_{ij} = f\left(|\bar{s}|, |\bar{\Omega}|, \partial \bar{p}/\partial x_i, \partial k/\partial x_i\right)$$

where $|\bar{s}| = (\bar{s}_{ij}\bar{s}_{ij})^{1/2}$ and $|\bar{\Omega}| = (\bar{\Omega}_{ij}\bar{\Omega}_{ij})^{1/2}$. They propose another three influence variables to account for viscous (low-Re number) effects near walls; they use local Reynolds number, $k^{1/2}y/\nu$ (y denotes wall-distance), turbulent kinetic energy and the ratio of turbulent to mean flow time scales. They use a machine learning method based on random forest regression.

Duraisamy *et al.* [6] gives a comprehensive review on Machine Learning and turbulence modeling. They survey recent developments in bounding uncertainties in RANS models using physical constraints and they present methods how to use machine learning to improve turbulence models.

Weatheritt and Sandberg [7] employed Genetic Expression Programming (GEP) to train an EARSM. The improvement of the Reynolds stress is chosen as the target application. They train the model in backward facing step and periodic hill flow. Then they do *à posteriori* simulations of the hill flow. However, when applying the GPA model they use a frozen velocity field predicted by a SST $k-\omega$ model.

Akoleka *et al.* [8] use the methodology developed in [7]. They train an EARSM using data of a DNS of a low-pressure turbine blade. Then they used the EARSM-GEP model to carry out 2D URANS simulations of the same turbine blade. The underlying turbulence model was the $\underline{k}-\overline{v'^2}-\omega$ transition model. However, they used the frozen concept, solving only $\overline{v'^2}$ and ω and taking the remaining variables from DNS.

Neural Network (NN) is used in the present work to improve an EARSM. The turbulent kinetic energy and its dissipation are predicted by the standard $k-\omega$ model. The NN model is trained in channel flow of $Re_\tau = 10{,}000$. The NN model is stored to disk and subsequently loaded into the CFD code. The NN model is called every iteration to compute the β coefficients in the EARSM. Then the Reynolds stresses are computed which are used in the momentum equations and the production term in the k and ω equation.

It is found that when training the NN model, the target data cannot only be taken from DNS. The reason is that stress-strain relation and the turbulent kinetic energy of the DNS data are different from those of the $k-\omega$ model. Hence, the target data are taken both from DNS and a $k-\omega$ simulation.

2 Flow Equations

The momentum equations read

$$\frac{\partial \bar{v}_j \bar{v}_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial \bar{p}}{\partial x_i} + \frac{\partial}{\partial x_j}\left[\left(\nu + \nu_{eff}\right)\left(\frac{\partial \bar{v}_i}{\partial x_j} + \frac{\partial \bar{v}_j}{\partial x_i}\right)\right] - \frac{\partial(\overline{v_i' v_j'})_r}{\partial x_j} \tag{2}$$

When EARSM is used, ν_{eff} is the effective viscosity including the EARSM coefficient β_1, see Eq. (16). The last term includes $(\overline{v_i' v_j'})_r$ which is the residual stress tensor in EARSM. The total stress tensor is

$$\overline{v_i' v_j'} = -\nu_{eff}\left(\frac{\partial \bar{v}_i}{\partial x_j} + \frac{\partial \bar{v}_j}{\partial x_i}\right) + (\overline{v_i' v_j'})_r \tag{3}$$

When the standard $k-\omega$ is used without EARSM the residual stress vanishes and $\nu_{eff} = \nu_t$.

2.1 The Numerical Solver

The **pyCALC-RANS** code is used [9]. It is an incompressible, finite volume code written in Python. It is fully vectorized (i.e. no for loops). The convective terms in all equations are discretized using the Hybrid central/upwind scheme. The numerical procedure is based on the pressure-correction method, SIMPLEC, and a collocated grid arrangement using Rhie-Chow interpolation [10].

2.2 The $k-\omega$ Model

The Wilcox $k-\omega$ turbulence model reads [11]

$$\begin{aligned}\frac{\partial \bar{v}_j k}{\partial x_j} &= P^k + \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\nu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right] - C_\mu k\omega \\ \frac{\partial \bar{v}_j \omega}{\partial x_j} &= C_{\omega_1}\frac{\omega}{k}P^k + \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\nu_t}{\sigma_\omega}\right)\frac{\partial \omega}{\partial x_j}\right] - C_{\omega 2}\omega^2 \\ \nu_t &= \frac{k}{\omega}\end{aligned} \tag{4}$$

The standard coefficients are used, i.e. $C_{\omega 1} = 5/9$, $C_{\omega 2} = 3/40$, $\sigma_k = \sigma_\omega = 2$ and $C_\mu = 0.09$. When EARSM is used, the production term is computed as

$$P^k = -\overline{v_i' v_j'}\frac{\partial \bar{v}_i}{\partial x_j} \tag{5}$$

and the dissipation in the EARSM reads

$$\varepsilon = C_\mu k\omega \tag{6}$$

In the $k-\omega$ model (without EARSM), the production term is computed as

$$P^k = \nu_t\left(\frac{\partial \bar{v}_i}{\partial x_j} + \frac{\partial \bar{v}_j}{\partial x_i}\right)\frac{\partial \bar{v}_i}{\partial x_j} \tag{7}$$

3 The EARSM

The Algebraic Stress Model (ASM) [12] with the LRR pressure-strain model [13] reads [14]

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

Note that the last term in Eq. (8) is zero if c_2 is set to 5/9 [15]. Equation (8) can be written as [16]

$$\begin{aligned}N a_{ij} &= -A_1\bar{s}^*_{ij} + (a_{ik}\bar{\Omega}^*_{kj} - \bar{\Omega}^*_{ik}a_{kj}) - A_2\left(\bar{s}^*_{ik}a_{kj} + a_{ik}\bar{s}^*_{kj} - \frac{2}{3}\delta_{ij}\bar{s}^*_{mn}a_{nm}\right) \\ \bar{s}^*_{ij} &= \frac{k}{\varepsilon}\bar{s}_{ij}, \quad \bar{\Omega}^*_{ij} = \frac{k}{\varepsilon}\bar{\Omega}_{ij}\end{aligned} \tag{9}$$

where

$$A_1 = 1.54, \quad A_2 = 0.37, \quad A_3 = 1.45, \quad A_4 = 2.89 \tag{10}$$

In order to get an explicit form of Eq. 9, Girimaji [17, 18] and Wallin & Johansson [16, 19], formulated a_{ij} in terms of the strain-rate tensor ($\bar{s}_{ij}$) and the vorticity tensor ($\bar{\Omega}_{ij}$). In 2D, it reads [2]

$$a_{ij} = \beta_1 \bar{s}^*_{ij} + \beta_2 \left(\bar{s}^*_{ik}\bar{s}^*_{kj} - \frac{1}{3}\bar{s}^*_{mn}\bar{s}^*_{nm}\delta_{ij} \right) + \beta_4 (\bar{s}^*_{ik}\bar{\Omega}^*_{kj} - \bar{\Omega}^*_{ik}\bar{s}^*_{kj}) \tag{11}$$

By inserting Eq. (11) in Eq. (9), Girimaji [17, 18] and Wallin & Johansson [16] derived an explicit form which in 2D reads [16] (a detailed derivation is given in [14])

$$\beta_1 = -\frac{A_1 N}{Q}, \quad \beta_2 = 2\frac{A_1 A_2}{Q}, \quad \beta_4 = -\frac{A_1}{Q}, \quad Q = N^2 - 2II_\Omega - \frac{2}{3}A_2^2 II_S \tag{12}$$

where N is given by the cubic equation

$$\begin{aligned} &N^3 - A_3 N^2 - \left(\left(A_1 A_4 + \frac{2}{3}A_2^2 \right) II_S + 2II_\Omega \right) N + 2A_3 \left(\frac{1}{3}A_2^2 II_S + II_\Omega \right) = 0 \\ &II_S = \bar{s}^*_{mn}\bar{s}^*_{nm}, \quad II_\Omega = \bar{\Omega}^*_{mn}\bar{\Omega}^*_{nm}. \end{aligned} \tag{13}$$

Equation (13) can be solved analytically. The analytical solution for the positive root reads [16]

$$N = \begin{cases} \frac{A_3}{3} + \left(P_1 + \sqrt{P_2} \right)^{1/3} + \text{sign}\left(P_1 - \sqrt{P_2} \right) \left| P_1 - \sqrt{P_2} \right|^{1/3}, & P_2 \geq 0 \\ \frac{A_3}{3} + 2\left(P_1^2 - P_2 \right)^{1/6} \cos\left[\frac{1}{3} \arccos\left(\frac{P_1}{\sqrt{P_1^2 - P_2}} \right) \right], & P_2 < 0 \end{cases} \tag{14}$$

where

$$\begin{aligned} P_1 &= \left(\frac{A_3}{27} + \left(\frac{A_1 A_4}{6} - \frac{2}{9}A_2^2 \right) II_S - \frac{2}{3} II_\Omega \right) A_3 \\ P_2 &= P_1^2 - \left(\frac{A_3}{9} + \left(\frac{A_1 A_4}{3} + \frac{2}{9}A_2^2 \right) II_S + \frac{2}{3} II_\Omega \right)^3 \end{aligned}$$

The Reynolds stress tensor including only the first term in Eq. (11) reads (see Eq. 8)

$$\overline{v_i' v_j'} = \beta_1 k \bar{s}^*_{ij} + \frac{2}{3}k\delta_{ij} = \beta_1 \frac{k^2}{\varepsilon}\bar{s}_{ij} + \frac{2}{3}k\delta_{ij} = -\nu_{eff}\bar{s}_{ij} + \frac{2}{3}k\delta_{ij} \tag{15}$$

where

$$\nu_{eff} = -0.5\beta_1 \frac{k^2}{\varepsilon} \tag{16}$$

is the effective viscosity and $\varepsilon = C_\mu k\omega$, see Eq. (6). Equation (15) corresponds to the Boussinesq assumption with $\beta_1 = -2C_\mu$. The discretized momentum equation on matrix form reads

$$AW = b \tag{17}$$

where W is $\bar{v}_1$ or $\bar{v}_2$. The term including the effective viscosity, ν_{eff}, in Eq. (15) is included in A which greatly improves the numerical stability of the CFD code.

3.1 The Neural Network Model

Instead of computing β_1, β_2 and β_4 from Eqs. (12) and (13), I will in the present work make them functions of some input parameter(s) (to be determined) using Neural Network (NN). The process can be depicted as:

1. Choose input parameter(s) involving e.g. the velocity gradient, the shear stress, the dissipation, the wall distance which should all be non-dimensional.
2. The output (target) parameters are β_1, β_2, β_4.
3. Train the NN model in fully-developed channel flow.
4. Use the NN model to compute β_1, β_2, β_4 in the EARSM (k and ω predicted with the $k-\omega$ model) in the **pyCALC-RANS** CFD code.

In fully-developed channel flow the EARSM (Eq. 11) reads:

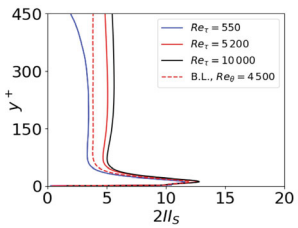
$$\begin{aligned} a_{11} &= \frac{1}{12}\left(\frac{\partial \bar{v}_1^*}{\partial y}\right)^2(\beta_2 - 6\beta_4), \quad a_{22} = \frac{1}{12}\left(\frac{\partial \bar{v}_1^*}{\partial y}\right)^2(\beta_2 + 6\beta_4) \\ a_{33} &= -\frac{2\beta_2}{12}\left(\frac{\partial \bar{v}_1^*}{\partial y}\right)^2, \quad a_{12} = \frac{\beta_1}{2}\frac{\partial \bar{v}_1^*}{\partial y}, \quad \frac{\partial \bar{v}_1^*}{\partial y} = \frac{k}{\varepsilon}\frac{\partial \bar{v}_1}{\partial y} \end{aligned} \tag{18}$$

From the relations above, I get the targets for the NN model

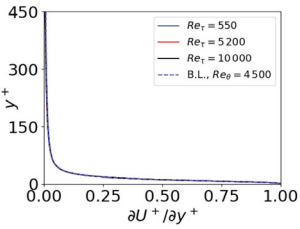
$$\beta_1 = \frac{2a_{12}}{\frac{\partial \bar{v}_1^*}{\partial y}}, \quad \beta_2 = \frac{6(a_{11}+a_{22})}{\left(\frac{\partial \bar{v}_1^*}{\partial y}\right)^2}, \quad \beta_4 = \frac{a_{22}-a_{11}}{\left(\frac{\partial \bar{v}_1^*}{\partial y}\right)^2} \tag{19}$$

a_{ij}, $k/\varepsilon \equiv (\omega C_\mu)^{-1}$ and $\frac{\partial \bar{v}_1}{\partial y}$ are computed from DNS data of channel flow. Note that a_{33} is defined by $a_{ii} = 0$.

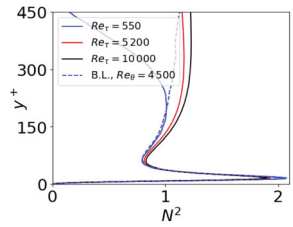
The output parameters of the NN model are β_1, β_2 and β_4. What input parameters should be used? The NN model should be applicable at difference Reynolds numbers so it should be a good idea to choose input parameters which also are Reynolds number independent. The NN model will be used in the CFD code and validated against DNS data in channel flow at $Re_\tau = 2000$ [20], $Re_\tau = 5200$ [21] $Re_\tau = 10{,}000$ [22] and flat-plate boundary-layer flow $Re_\tau = 5\,500$ [23].



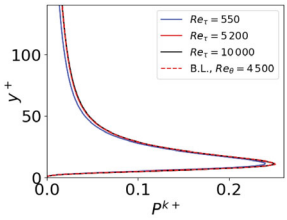
(a) Velocity gradient scaled with k and $\tilde{\varepsilon}$.



(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.

Fig. 1 DNS data $\tilde{\varepsilon} = \varepsilon - \nu\partial^2 k/\partial y^2$

Figure 1 presents four possible dimensionless input parameters. $II_S = \frac{1}{2}\left(\frac{k}{\tilde{\varepsilon}}\frac{\partial \bar{v}_1}{\partial y}\right)^2$, $\frac{\partial U^+}{\partial y^+}$, $N = P^k/\tilde{\varepsilon}$ and P^{k+}. Note that $\tilde{\varepsilon} = \varepsilon - \nu\partial^2 k/\partial y^2$ is used because its near-wall behaviour is similar to that of the dissipation tern in the $k-\omega$ model. The velocity gradient seems to be best (i.e. least Re number dependent), but it turns out that it is very difficult to get the NN model to converge with this input variable. The reason is probably its large gradient near the wall, see Fig. 1b. I choose the second best, i.e. the production term together with y^+ as input parameters. I scale the two input parameters using `MinMaxScaler()` so that they are in the range [0, 1]

I use the NN in Python's `pytorch`. Figure 2 shows the NN model schematically. The optimizer is set as

```
optimizer = torch.optim.SGD(neural_net.parameters(), lr=l_rate)
```

with learning rate `l_rate = 0.07`; the number of epochs is 5000. When predicting the β coefficientsusing the 20% of DNS data the maximum error, e_i, defined as ($i = 1, 2,$ or 4)

$$e_i = \max\left(\frac{|\beta_i - \beta_{i,DNS}|}{\beta_{i,DNS}}\right)$$

is less than 2.5%. More detail on the use of `pytorch` can be found in [24].

Since I use y^+ as input variable, I must train the NN model at the largest Reynolds number (which has the largest y^+ value), i.e. channel flow at $Re_\tau = 10{,}000$. An option could be to train the model at all Reynolds numbers. I exclude data in the viscous sublayer ($y^+ \leq 5$) because the gradient of β_2 and β_4 (see Eq. 19) are very large near the wall. I also exclude data near the center ($y^+ > 9\,800$) where $\frac{\partial \bar{v}_1^*}{\partial y}$ is very

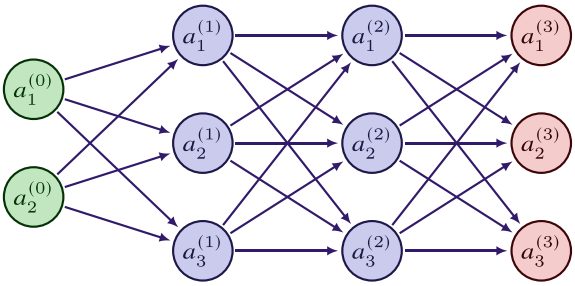
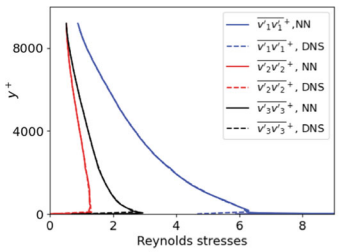
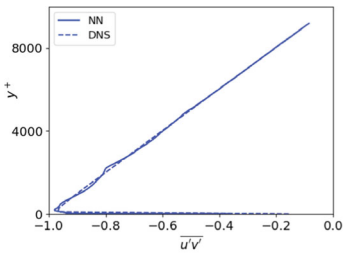


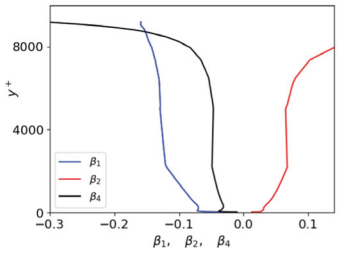
Fig. 2 The Neural Network with two inputs variables, $a_1^{(0)} = y^+$ and $a_2^{(0)} = P^+$ and three output variables, $a_1^{(3)} = \beta_1$, $a_2^{(3)} = \beta_2$ and $a_3^{(3)} = \beta_4$. There are three neurons and two hidden layers in this figure; in the simulations I use 50 neurons



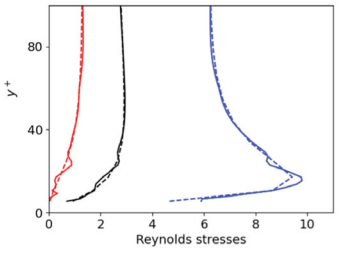
(a) Reynolds normal stresses.



(b) Reynolds shear stress.



(c) Predicted EARSM coefficients.



(d) Reynolds normal stresses near the wall.

Fig. 3 Predicted with NN model (no CFD). $Re_\tau = 10{,}000$. Markers: DNS data [22]

small. The turbulence is negligible in both these regions. I train on 80% of the data (approximately 800 randomly chosen data points) and test on (predict) the remaining 20%.

Figure 3 presents the predicted stresses. The agreement is—as it should—almost perfect. However, Fig. 3d shows that there are some small discrepancies near the wall. Figure 3c presents the EARSM coefficients predicted by the NN model.

4 The NN Model Incorporated in the CFD Solver

The prediction of Reynolds stresses presented in the previous section is *à priori* study. In the literature, these studies are very common. But they are of little use unless the NN model is coupled to a CFD code because in channel flow the velocity profile is entirely determined by the turbulent shear stress (see discussion in the next section).

Hence, in the next step I save the NN model to disk and then a load it into the CFD code. I include the NN model in the CFD code as follows:

1. Load the NN model
2. Solve $\bar{v}_1$, $\bar{v}_2$ and P' equations. The Reynolds stresses $\overline{v_1'^2}$, $\overline{v_2'^2}$, $\overline{v_1' v_2'}$ in the $\bar{v}_1$ and $\bar{v}_2$ equations (see Eq. 2) are taken from the previous iteration.
3. Compute β_1, β_2, β_4 using the NN model. Limits are set on both input and output parameters corresponding to min and max values during the training process.
4. Compute the anisotropic Reynolds stresses (a_{11}, a_{22}, a_{12}) using the β coefficients, see Eq. (18).
5. Compute the Reynolds stresses $\overline{v_1'^2} = ka_{11} + \frac{2}{3}k$, $\overline{v_2'^2} = ka_{22} + \frac{2}{3}k$, $\overline{v_1' v_2'} = ka_{12}$.
6. Solve the k and ω equations. The Reynolds stresses are used in the production term, see Eq. (5). In fully-developed channel flow $\overline{v_1'^2}$ and $\overline{v_2'^2}$ have no effect since $\partial \bar{v}_1/\partial x_1 = \partial \bar{v}_2/\partial x_2 = 0$, but in flat-plate boundary layer flow they have a small effect.
7. End of iteration. Repeat from Item 2 until convergence (1000 s of iterations).

4.1 Channel Flow with the NN Model Trained on DNS Data

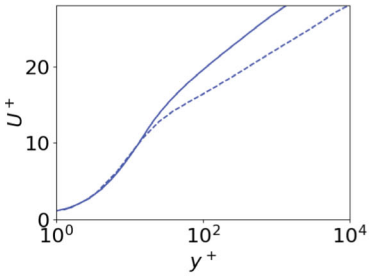
Channel flow simulations are made. The grid has 110 cells in the y direction and is stretched by 10% from the walls. The first cell center is at $y^+ \simeq 0.5$. Figure 4 presents the predicted mean flow and the Reynolds stresses. The agreement of the predicted mean flow with the DNS data is poor. The shear stress agrees well DNS data but that does not signify anything. When using a finite volume CFD method, the shear stress will always agree with DNS data. Integrating the momentum equation the total shear stress, $\tau_{12,tot}$ (viscous plus turbulent), reads [14]

$$\tau_{12,tot} = \tau_w + \frac{1}{\rho}\frac{\partial \bar{p}}{\partial x_1} x_2 = \tau_w \left(1 - \frac{x_2}{\delta}\right) \tag{20}$$

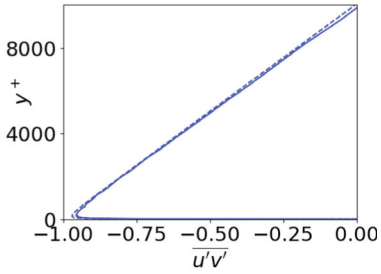
where δ is the half-channel width and the total shear stress is defined as

$$\tau_{12,tot} = \nu \frac{\partial \bar{v}_1}{\partial x_2} - \overline{v_1' v_2'}. \tag{21}$$

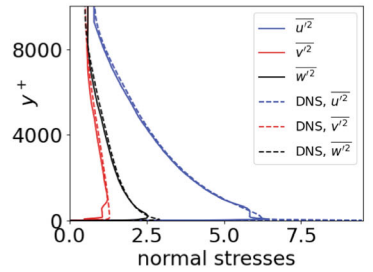
At the last step in Eq. (20), I used the fact that the pressure gradient balances the wall shear stress. The velocity gradient will adapt so that Eq. (20) is satisfied.



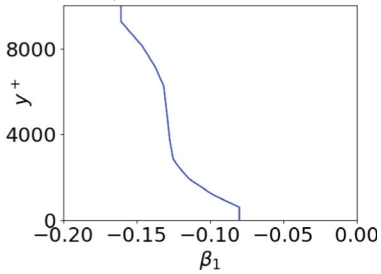
(a) Velocity.



(b) Reynolds shear stresses.



(c) Reynolds normal stresses.



(d) β_1 coefficient.

Fig. 4 The NN model incorporated in the CFD code. $Re_\tau = 10{,}000$. Dashed lines: DNS data [22]

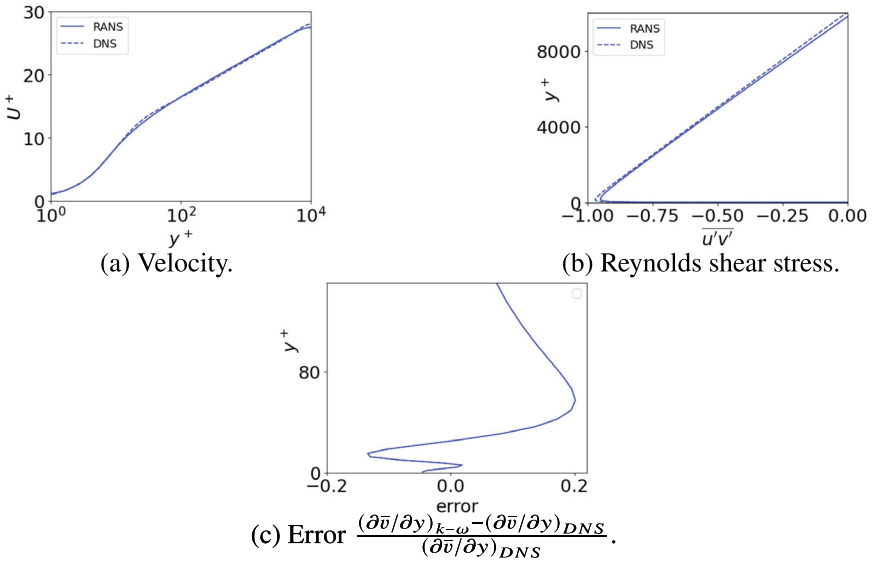
Why is the velocity so poorly predicted? To find out, I make a CFD simulation using the $k-\omega$ model (no EARSM, no NN model). Figure 5a, b show the predicted velocity and shear stress and the agreement with DNS data is (or seems to be) excellent. The relation between the velocity gradient and the shear stress in the $k-\omega$ model is given by the Boussinesq assumption, see Eq. (15)

$$\overline{v_1' v_2'} = -2\nu_t \bar{s}_{12} = -\frac{2k}{\omega}\bar{s}_{12} = -\frac{2C_\mu k^2}{\varepsilon}\bar{s}_{12} \tag{22}$$

using $\varepsilon = C_\mu k\omega$, see Eq. (6). Comparing Eqs. (22) and (15) gives that

$$\beta_1 = -2C_\mu \tag{23}$$

The poor agreement in Fig. 4a can now be understood. The $k-\omega$ models predicts an (seemingly) excellent velocity profile (Fig. 5a) where the relation in Eq. (22) is used with $C_\mu = 0.09$. If the CFD and NN should give the same results, then the relation in Eq. (23) must be satisfied; Fig. 4d shows that this is not the case. The reason is that although the total shear stresses in the $k-\omega$ predictions and the DNS data both follow the linear law in Eq. (20), the velocity gradients are different. Figure 5c shows that the discrepancy is more than 20%.



(a) Velocity. (b) Reynolds shear stress.

(c) Error $\frac{(\partial\bar{v}/\partial y)_{k-\omega}-(\partial\bar{v}/\partial y)_{DNS}}{(\partial\bar{v}/\partial y)_{DNS}}$.

Fig. 5 CFD using the $k-\omega$ model. Channel flow, $Re_\tau = 10{,}000$. Markers: DNS data [22]

4.2 Channel Flow with the NN Model Trained on $k-\omega$ and DNS Data

Instead of training the NN model on DNS data as in Sect. 4.1, I will train on data taken both from DNS and the $k-\omega$ simulation shown in Fig. 5. The fact that this may be necessary was noted in [25]. I will use the following data:

- Input: P^k and y^+ from $k-\omega$ prediction
- Target: β_1, β_2 and β_4 computed from $\underbrace{\overline{v_1'^2}, \overline{v_2'^2}}_{DNS}$ and $\underbrace{\overline{v_1'v_2'}, k, \varepsilon}_{k-\omega}$

This will ensure that the relation between the shear stress and the velocity gradient as well as the turbulent kinetic energy are the same in the training process as in the CFD simulation. Note that $k_{DNS} = 0.5(\overline{v_1'^2}_{DNS} + \overline{v_2'^2}_{DNS} + \overline{v_3'^2}_{DNS})$ is not equal to the turbulent kinetic energy, $k_{k-\omega}$, predicted by the $k-\omega$ model. The spanwise normal stress, $\overline{v_3'^2}$, predicted by the NN model will adapt in order to satisfy $k_{k-\omega} = 0.5(\overline{v_1'^2}_{DNS} + \overline{v_2'^2}_{DNS} + \overline{v_3'^2}_{DNS})$. Hence, $\overline{v_3'^2}$ will be incorrectly predicted by the NN model (it even goes negative near the wall). Thus, the proposed EARSM-NN model is applicable only in two-dimension flows. In order to make the model applicable in three dimension, a new $k-\omega$ (or $k-\varepsilon$) model must be developed which satisfies $k_{DNS} = k_{k-\omega}$. This is out of the scope of the present work.

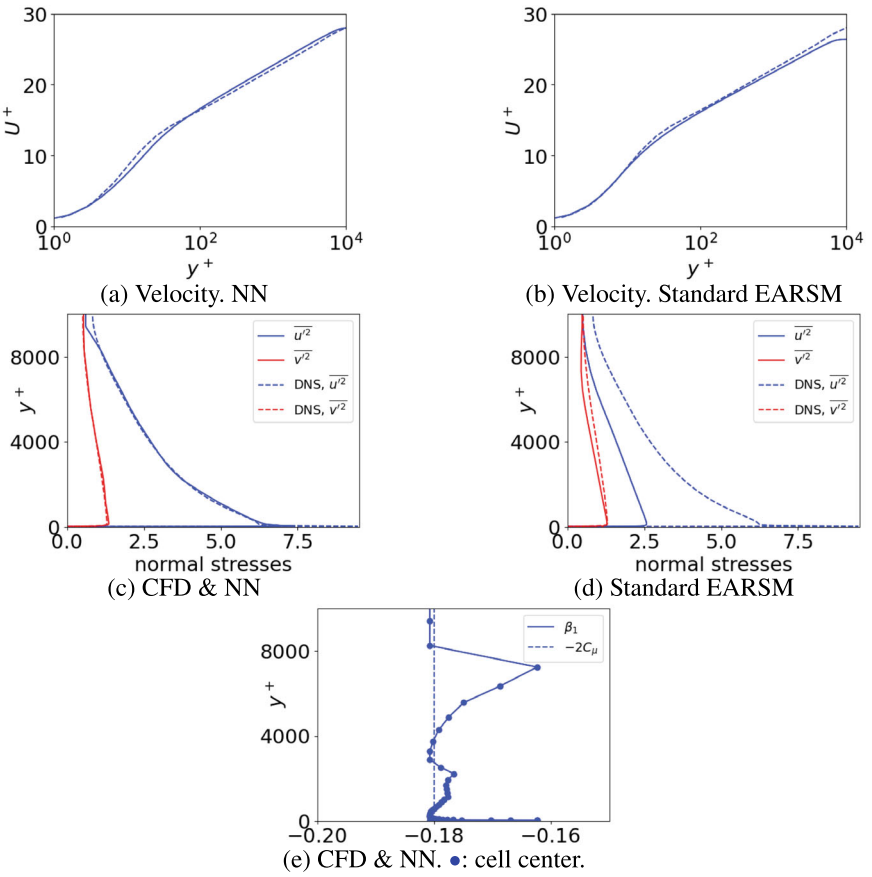
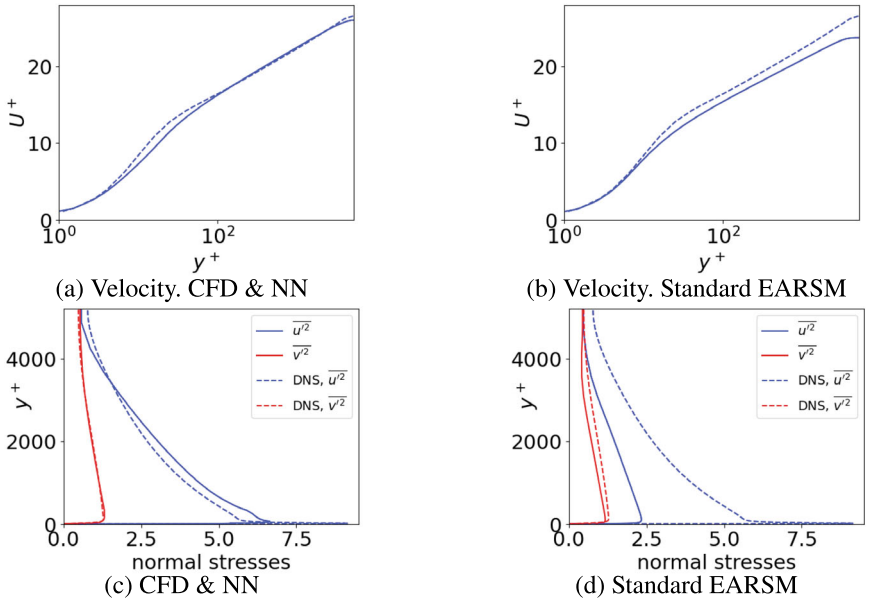


Fig. 6 CFD predictions with EARSM-NN model and standard EARSM. The NN model is trained using DNS and $k-\omega$ data. Channel flow, $Re_\tau = 10{,}000$. Dashed lines: DNS

Figure 6 presents CFD prediction using the EARSM-NN and the standard EARSM (see Eqs. (10), (12) and (14)). Both models predict the velocity well although the standard EARSM gives a slight overprediction near the center and the EARSM-NN profile exhibits somewhat too low values in the buffer layer. However, the EARSM-NN predicts much better normal stresses than the standard EARSM. Figure 6e presents the β_1 coefficient and its values are close to $-2C_\mu$ as it should. Setting $\beta_1 = -2C_\mu$ in EARSM-NN gives very similar results (not shown) as in Fig. 8a, c.

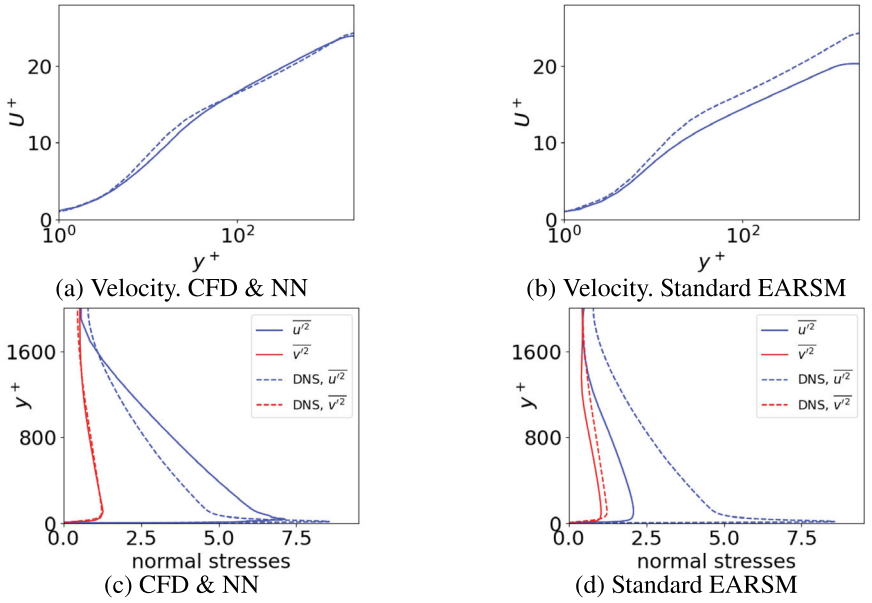
At $Re_\tau = 5200$ (Fig. 7) and $Re_\tau = 2000$ (Fig. 8) the EARSM-NN model predicts much better velocity profiles and normal Reynolds stresses than the standard EARSM. At the lower Reynolds number, The EARSM-NN model (Fig. 8c) overpredicts the streamwise normal stress by some 15%, whereas the standard EARSM (Fig. 8d) model predicts up to 50% too small values.



(a) Velocity. CFD & NN (b) Velocity. Standard EARSM

(c) CFD & NN (d) Standard EARSM

Fig. 7 CFD predictions with EARSM-NN model and standard EARSM. The NN model is trained using DNS and $k-\omega$ data. $Re_\tau = 5200$. Dashed lines: DNS



(a) Velocity. CFD & NN (b) Velocity. Standard EARSM

(c) CFD & NN (d) Standard EARSM

Fig. 8 CFD predictions with EARSM-NN model and standard EARSM. The NN model is trained using DNS and $k-\omega$ data. Channel flow, $Re_\tau = 2000$. Dashed lines: DNS

4.3 Flat-Plate Boundary-Layer Flow

In this section I present simulations of a flat-plate boundary layer. The grid has 300×90 (x, y) cells and is stretched by 10% from the wall but limiting the cell size to $\Delta y_{max} = 0.5$. The first cell center is at $y^+ \simeq 0.5$. The streamwise grid spacing is constant, $\Delta x = 0.5$. The inlet boundary-layer thickness is $\delta_{in} \simeq 0.8$. A refined mesh (600×180) gives identical results as the coarse mesh.

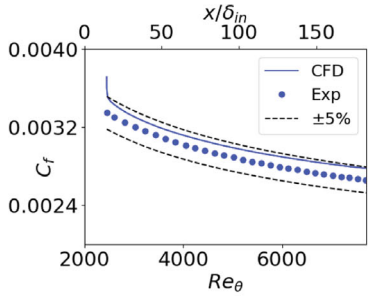
A pre-cursor $k-\omega$ simulation of a flat-plate boundary layer is carried out and $\bar{v}_1$, $\bar{v}_1, k$ and ω are stored at $Re_\theta = 2\,500$ where θ denotes the boundary-layer momentum thickness. These stored data are used as inlet boundary condition in the subsequent flat-plate boundary-layer simulations using the two EARSM models.

Figure 9 presents the predicted skin friction along with velocity and normal stresses at $Re_\theta = 2\,500$. Again, the EARSM-NN model performs much better than the standard EARSM although the normal stresses are less well predicted than for the channel flow. But the EARSM-NN model predicts the skin friction within 5% of experimental data whereas the standard EARSM gives some 14% too large values.

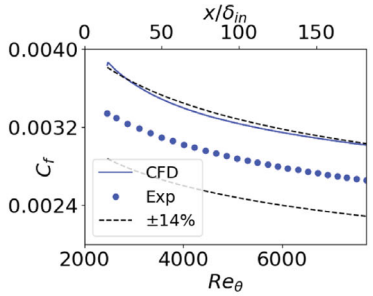
The channel flow simulations at $Re_\tau = 10{,}000$ were also carried out setting $\beta_1 = -2C_\mu$ and it was noted that the results were similar to those in Figs. 8a, c. This was also found for the other two Reynolds numbers, $Re_\tau = 5\,200$ and $Re_\tau = 2\,000$ (not shown). However, in the boundary layer flow the results are somewhat worse setting $\beta_1 = -2C_\mu$; the skin friction is then over-predicted by 6%, see Fig. 10a. The skin friction predicted with the Wilcox $k-\omega$ model is also 6% too large (see Fig. 10b), but the trend is much worse. The β_1 coefficient predicted with EARSM-NN is shown in Fig. 10c and it can be seen that near the wall it is larger than $-2C_\mu$.

5 Conclusions

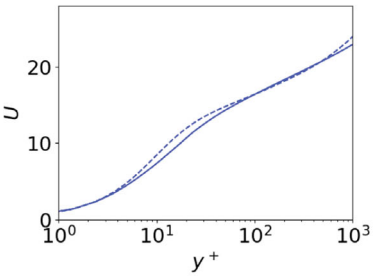
An Explicit Algebraic Reynolds Stress Model (together with Wilcox $k-\omega$ model) has been improved using Neural Network (NN). The NN model is trained in channel flow at $Re_\tau = 10{,}000$. It is found that target data cannot be taken only from DNS because the stress-strain relation and the turbulent kinetic energy are not the same in DNS and $k-\omega$ predictions. Hence the target data are taken both from DNS ($\overline{v_1'^2}$ and $\overline{v_2'^2}$) and a $k-\omega$ simulation ($\frac{\partial \bar{v}_1}{\partial y}$, $\overline{v_1' v_2'}$, k, $\varepsilon = C_\mu k\omega$). In this way the strain-stress relation and the turbulent kinetic energy are the same in the training process as in the CFD-NN predictions. Since k in the training process is taken from the $k-\omega$ results it means that $k \neq 0.5(\overline{v_1'^2}_{DNS} + \overline{v_2'^2}_{DNS} + \overline{v_3'^2}_{DNS})$. In the NN model, the spanwise Reynolds stress adapts to satisfy $a_{ii} = 0$ which means that $\overline{v_3'^2}$ is not correctly predicted (it even goes negative in the near-wall region). Hence, the EARSM-NN model is applicable only to two-dimensional flow where $\overline{v_3'^2}$ is not used. One way to make the model applicable in three-dimensional flow is to develop a $k-\omega$ (or $k-\varepsilon$ model) which accurately predicts the turbulent kinetic energy.



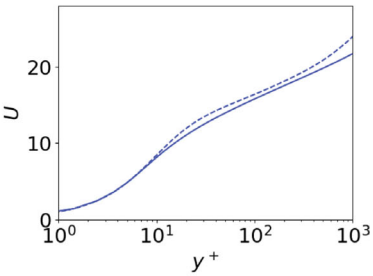
(a) Skin friction. CFD & NN.



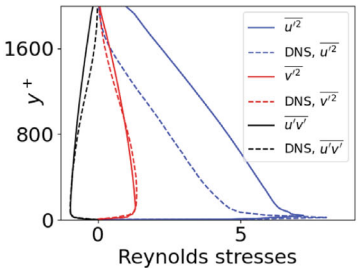
(b) Skin friction. Standard EARSM



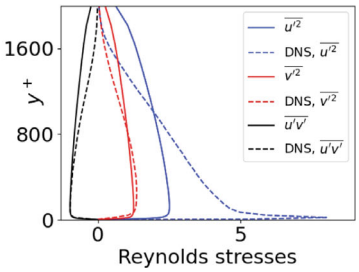
(c) EARSM & NN.



(d) Standard EARSM

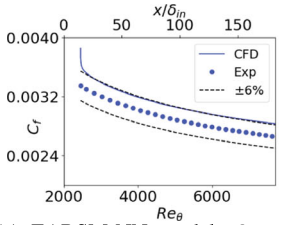


(e) CFD & NN

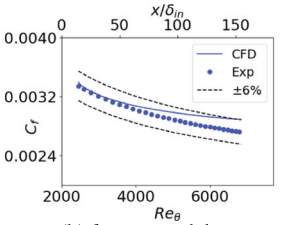


(f) Standard EARSM

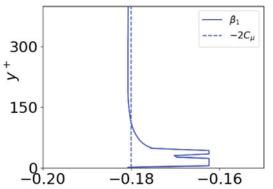
Fig. 9 CFD predictions with EARSM-NN and standard EARSM. The NN model is trained using DNS and $k-\omega$ data. Flat-plate boundary layer, $Re_\theta = 5\,500$. Dashed lines: DNS



(a) EARSM-NN model. $\beta_1 = -2C_\mu$.



(b) $k-\omega$ model.



(c) EARSM-NN model. $Re_\theta = 5\,500$.

Fig. 10 CFD predictions. Flat-plate boundary layer

Acknowledgement This study was partly financed by the *Strategic research project on Chalmers on hydro- and aerodynamics* and *Chalmers Transport Area of Advance*, Grant No. C 2023-0125-19.

The computations were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS), partially funded by the Swedish Research Council through grant agreement no. 2022-06725.

References

1. Craft, T.J., Launder, B.E., Suga, K.: Prediction of turbulent transitional phenomena with a nonlinear eddy-viscosity model. Int. J. Heat Fluid Flow **18**, 15–28 (1997)
2. Pope, S.B.: A more general effective-viscosity hypothesis. J. Fluid Mech. **472**, 331–340 (1975)
3. Ling, J., Kurzawski, A., Templeton, J.: Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. J. Fluid Mech. **807**, 155–166 (2016). https://doi.org/10.1017/jfm.2016.615
4. Yin, Y., Shen, Z., Zhang, Y., Chen, H., Fu, S.: An iterative data-driven turbulence modeling framework based on reynolds stress representation. Theor. Appl. Mech. Lett. **12**(5), 100381 (2022). https://doi.org/10.1016/j.taml.2022.100381
5. Wu, J.-L., Xiao, H., Paterson, E.: Physics-informed machine learning approach for augmenting turbulence models: a comprehensive framework. Phys. Rev. Fluids **3**, 074602 (2018). https://doi.org/10.1103/PhysRevFluids.3.074602
6. Duraisamy, K., Iaccarino, G., Xiao, H.: Turbulence modeling in the age of data. Annu. Rev. Fluid Mech. **51**(1), 357–377 (2019). https://doi.org/10.1146/annurev-fluid-010518-040547
7. Weatheritt, J., Sandberg, R.: A novel evolutionary algorithm applied to algebraic modifications of the rans stress-strain relationship. J. Comput. Phys. **325**, 22–37 (2016). https://doi.org/10.1016/j.jcp.2016.08.015
8. Akolekar, H.D., Weatheritt, J., Hutchins, N., Sandberg, R.D., Laskowski, G., Michelassi, V.: Development and use of machine-learnt algebraic reynolds stress models for enhanced prediction of wake mixing in low-pressure turbines. J. Turbomach. **141**(4), 041010 (2019). https://doi.org/10.1115/1.4041753
9. Davidson, L.: pyCALC-RANS: a 2D python code for RANS, Division of Fluid Dynamics, Dept. of Mechanics and Maritime Sciences, Chalmers University of Technology, Gothenburg (2021)
10. Rhie, C.M., Chow, W.L.: Numerical study of the turbulent flow past an airfoil with trailing edge separation. AIAA J. **21**, 1525–1532 (1983)
11. Wilcox, D.C.: Reassessment of the scale-determining equation. AIAA J. **26**(11), 1299–1310 (1988)
12. Rodi, W.: A new algebraic relation for calculating the Reynolds stresses. ZAMM **56**, T219–T221 (1976)
13. Launder, B.E., Reece, G.J., Rodi, W.: Progress in the development of a Reynolds-stress turbulence closure. J. Fluid Mech. **68**(3), 537–566 (1975)
14. Davidson, L.: Fluid mechanics, turbulent flow and turbulence modeling, eBook, Division of Fluid Dynamics. Chalmers University of Technology, Gothenburg, Dept. of Mechanics and Maritime Sciences (2021)
15. Taulbee, D.B.: An improved algebraic Reynolds stress model and corresponding nonlinear stress model. Phys. Fluids A **4**, 2555–2561 (1992)
16. Wallin, S., Johansson, A.V.: A new explicit algebraic Reynolds stress model for incompressible and compressible turbulent flows. J. Fluid Mech. **403**, 89–132 (2000)
17. Girimaji, S.S.: Fully-explicit and self-consistent algebraic Reynolds stress model. Theoret. Comput. Fluid Dyn. **8**, 387–402 (1996)

18. Girimaji, S.: 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)
19. Wallin, S., Johansson, A.: A new explicit algebraic Reynolds stress turbulence model including an improved near-wall treatment, in: C.-J. Chen, C. Shih, J. Lienau, R. J. Kung (Eds.), Proc. Flow Modeling and Turbulence Measurements VI, Tallahassee F. L., Balkema, pp. 399–406 (1996)
20. Hoyas, S., Jimenez, J.: Reynolds number effects on the reynolds-stress budgets in turbulent channels, Physics of Fluids A 20 (101511) (2008)
21. Lee, M., Moser, R.D.: Direct numerical simulation of turbulent channel flow up to $Re_\tau \approx 5200$. J. Fluid Mech. **774**, 395–415 (2015). https://doi.org/10.1017/jfm.2015.268
22. Hoyas, S., Oberlack, M., Alcántara-Ávila, F., Kraheberger, S.V., Laux, J.: Wall turbulence at high friction Reynolds numbers. Phys. Rev. Fluids **7**, 014602 (2022). https://doi.org/10.1103/PhysRevFluids.7.014602
23. Sillero, J., Jimenez, J., Moser, R.: One-point statistics for turbulent wall-bounded flows at Reynolds numbers up to $\delta^+ \simeq 2000$, Phys. Fluids **25** (105102) (2014)
24. Davidson, L.: Using machine learning for improving a non-linear $k - \varepsilon$ model: a first attempt, Tech. rep., Division of Fluid Dynamics, Dept. of Mechanics and Maritime Sciences, Chalmers University of Technology, Gothenburg (2023)
25. Duraisamy, K.: Perspectives on machine learning-augmented reynolds-averaged and large eddy simulation models of turbulence. Phys. Rev. Fluids (2021). https://doi.org/10.1103/PhysRevFluids.6.050504