P.2 Part b: Machine learning

In this part you will use Neural Network (NN) to modify the Explicit Algebraic Stress Model (EARSM). The EARSM is presented in Sections 11.11 and AF but you find a shorter description in [206] (you can download the paper at Canvas).

In 2D, the EARSM reads (see Eq. 11 in [206])

$$\begin{aligned}
a_{ij} &= \beta_1 \bar{s}^*_{ij} + \beta_2\left(\bar{s}^*_{ik}\bar{s}^*_{kj} - \frac{1}{3}\bar{s}^*_{mn}\bar{s}^*_{nm}\delta_{ij}\right) + \beta_4(\bar{s}^*_{ik}\bar{\Omega}^*_{kj} - \bar{\Omega}^*_{ik}\bar{s}^*_{kj})\\
\bar{s}^*_{ij} &= \frac{k}{\varepsilon}\hat{s}_{ij}, \quad \bar{\Omega}^*_{ij} = \frac{k}{\varepsilon}\hat{\Omega}_{ij}
\end{aligned} \tag{P.10}$$

where

$$\begin{aligned}
\beta_1 &= -\frac{A_1 N}{Q}, \quad \beta_2 = 2\frac{A_1 A_2}{Q}, \quad \beta_4 = -\frac{A_1}{Q}\\
Q &= N^2 - 2II_\Omega - \frac{2}{3}A_2^2 II_S
\end{aligned} \tag{P.11}$$

where $A_1 - A_4$ are constants(see Eq. 10 in [206]) and N is given by a cubic equation which can be solved analytically.

In this assignment, we will study fully-developed channel flow for which Eq. P.10 reads

$$\begin{aligned}
a_{11} &= \frac{1}{12}\left(\frac{\partial \bar{v}^*_1}{\partial x_2}\right)^2(\beta_2 - 6\beta_4), \quad a_{22} = \frac{1}{12}\left(\frac{\partial \bar{v}^*_1}{\partial x_2}\right)^2(\beta_2 + 6\beta_4)\\
a_{33} &= -\frac{2\beta_2}{12}\left(\frac{\partial \bar{v}^*_1}{\partial x_2}\right)^2, \quad a_{12} = \frac{\beta_1}{2}\frac{\partial \bar{v}^*_1}{\partial x_2}, \quad \frac{\partial \bar{v}^*_1}{\partial x_2} = \frac{k}{\varepsilon}\frac{\partial \bar{v}_1}{\partial x_2}
\end{aligned} \tag{P.12}$$

In standard EARSM, the β coefficients are obtained from Eq. P.12. But in this assignment, we will compute them using Neural Network (NN) in PyTorch. Below I give you some useful links for a crash course on NN and PyTorch:

- Celsius to Fahrenheit with pytorch nn
- Building a Regression Model in PyTorch
- Multi-Target Predictions with Multilinear Regression in PyTorch
- How to Create a Neural Network

In NN we have input variables, output variables (targets), hidden layers and neurons, see Fig. P.1. For EARSM and NN, the targets are the three β coefficients in Eq. P.12. From Eq. P.12 we get explicit expressions for the targets

$$\beta_1 = \frac{2a_{12}}{\frac{\partial \bar{v}^*_1}{\partial x_2}}, \quad \beta_2 = \frac{6(a_{11}+a_{22})}{\left(\frac{\partial \bar{v}^*_1}{\partial x_2}\right)^2}, \quad \beta_4 = \frac{a_{22}-a_{11}}{\left(\frac{\partial \bar{v}^*_1}{\partial x_2}\right)^2} \tag{P.13}$$

where a_{11}, a_{12}, a_{22} and $\frac{\partial \bar{v}^*_1}{\partial x_2}$ may be taken from DNS data. In [206] I mention four possible dimensionless input parameters. $II_S = \frac{1}{2}\left(\frac{k}{\varepsilon}\frac{\partial \bar{v}_1}{\partial x_2}\right)^2$, $\frac{\partial U^+}{\partial y^+}$, $N = P^k/\tilde{\varepsilon}$ and P^{k+}. I finally chose to use P^{k+} and y^+, see caption in Fig. P.1.

Download the data and script files from the Canvas course page or

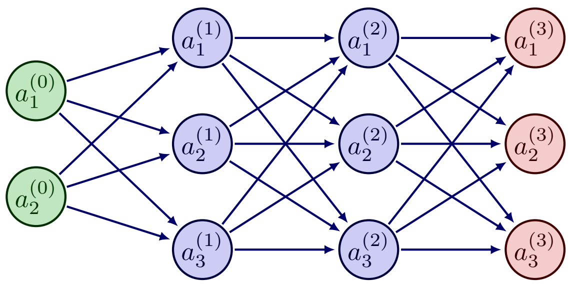

Figure P.1: The Neural Network with two inputs variables, $a_1^{(0)} = y^+$ and $a_2^{(0)} = P^+$ and three output variables, $a_1^{(3)} = \beta_1$, $a_2^{(3)} = \beta_2$ and $a_3^{(3)} = \beta_4$. There are three neurons and two hidden layers in this figure; in the simulations I use 50 neurons.

`http://www.tfd.chalmers.se/~lada/comp_turb_model`

The Python script `NN-earsm` uses PyTorch to train a EARSM-NN model on 80% of the DNS data (randomly chosen) and then test (predict) on the remaining 20%. Study `NN-earsm.py` carefully.

Assignment 1.9. In `NN-earsm.py` DNS data are used as input and target. Now you should take input from a $k-\omega$ simulation and target both from DNS and a $k-\omega$ simulation (as in Section 4.2 in [206]).

- 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 $\underbrace{\overline{v_1'v_2'}, k, \varepsilon}_{k-\omega}$.
- Data from a $k-\omega$ simulation are loaded at the end of `NN-earsm.py`.
- Plot $\overline{v_1'v_2'}$, $\overline{v_1'^2}$ and $\overline{v_2'^2}$. You should get similar results as in Fig. 6 in [206].

Now you have trained an EARSM-NN model using data of channel flow at $Re_\tau = 10\,000$. Next, you will use the model to predict the Reynolds stresses in another flow.

Assignment 1.10. Write a script which loads the EARSM-NN model you used in Assignment 1.9. You simply take out the training part in `NN-earsm.py` and load the EARSM-NN model as:

- torch.load('model-Re-10000.pth')
- load('scaler-yplus-Re-10000.bin')
- load('scaler-pk-Re-10000.bin')

- Predict and plot $\overline{v_1'v_2'}$, $\overline{v_1'^2}$ and $\overline{v_2'^2}$ of channel flow at $Re_\tau = 2\,000$.
- The plots should be similar to Fig. 8 in [206].