Machine learning to model closure terms of laminar to turbulent transitional boundary layer milestone

Shaun Harris

1 Introduction

Boundary layer flows are common in nature and often pose a significant problem in efficiently calculating the fluid flow for a given geometry. In simulating these flows, we often have access to the steady state base flow velocity profile and use models for calculating the closure terms stresses. Modeling these terms is a matter of research and this project attempts to use machine learning to model these quantities.

We will examine incompressible, 2D, steady state flow over a flat plate. It is desired to obtain a model that takes inputs of the averaged velocity state and output for each location in a flow the nonlinear averaged fluctuation terms. Various models were tested and the results for the model are shown in this milestone. PyTorch was used as the main machine learning library with mean squared error as the loss function and Adam as the optimization method.

2 Models

Various models were implemented to model the averaged nonlinear fluctuation terms. The resulting study shows the results and qualitative error associated with each of the two models.

2.1 Linear regression

As a first attempt, a linear regression was used where we had a dimension d of 16 values at each spatial location

$$x^{(i)} = \left[\overline{u_1}, \overline{u_2}, \overline{P}, \frac{\partial \overline{u_1}}{\partial x_1}, \frac{\partial \overline{u_2}}{\partial x_1}, \frac{\partial \overline{P}}{\partial x_1}, \frac{\partial \overline{u_1}}{\partial x_2}, \frac{\partial \overline{u_2}}{\partial x_2}, \frac{\partial \overline{P}}{\partial x_2}, \frac{\partial^2 \overline{u_1}}{\partial x_1^2}, \frac{\partial^2 \overline{u_2}}{\partial x_1^2}, \frac{\partial^2 \overline{P}}{\partial x_1^2}, \frac{\partial^2 \overline{u_1}}{\partial x_2^2}, \frac{\partial^2 \overline{u_2}}{\partial x_2^2}, \frac{\partial^2 \overline{P}}{\partial x_2^2}, \nu\right]^T. \tag{1}$$

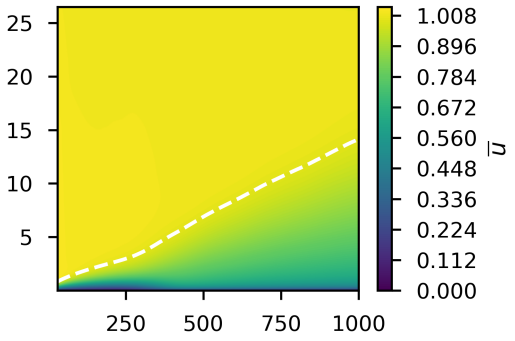
The overbar indicates an averaged steady state base flow value. Each spatial location was treated as a new data point. Data was extracted from John Hopkins Turbulence Database (JHTDB) at `http://turbulence.pha.jhu.edu` for the transitional flat plate boundary layer that matched inputs of the T3A boundary layer transition case. This flow is used as it contains both laminar and turbulent data and the associated transition process.

The labels for each of the data points are the six quantities as

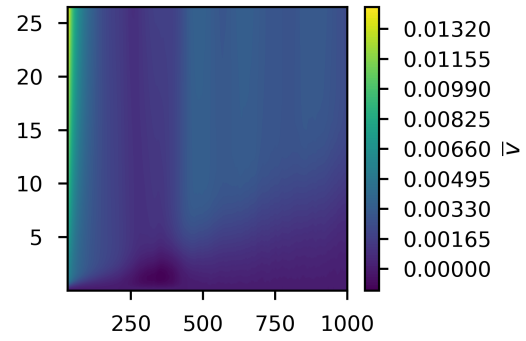
$$y^{(i)} = \left[\overline{u_1' u_1'}, \overline{u_1' u_2'}, \overline{u_1' u_3'}, \overline{u_2' u_2'}, \overline{u_2' u_3'}, \overline{u_3' u_3'}\right]^T \tag{2}$$

The prime indicates a fluctuation quantity around the averaged flow. Thus, the decomposition of the true velocity at any point in time would be $u(t, x, y, z) = \overline{u}(x, y) + u'(t, x, y, z)$ or alternatively $u(t, x_1, x_2, x_3) = \overline{u}(x_1, x_2) + u'(t, x_1, x_2, x_3)$ where x or x_1 denotes the streamwise dimension, y or x_2 denotes the wall normal dimension, and z or x_3 denotes the spanwise dimension. We note here that we have averaged the quantities in time and in the spanwise dimensions. The steady state boundary layer base flow velocity and pressure is shown in Fig. 1. Note here that $\overline{u_1} = \overline{u}$ and $\overline{u_2} = \overline{v}$

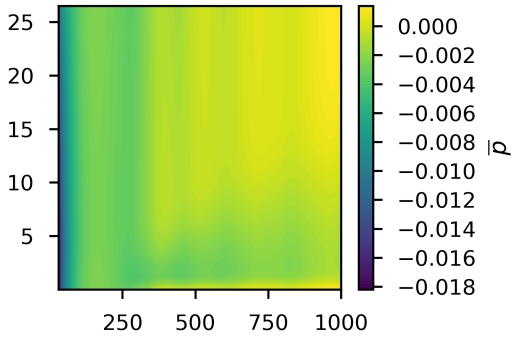
Using PyTorch, a linear regression was conducted with these sixteen inputs using the first of the output labels. The training data was split into training and validation data using a random sampling. The resulting model for the whole domain is shown in Fig. 2. We see that the prediction is not as accurate as desired. Further analysis and reporting of errors will be conducted here for the final project.



(a) Streamwise velocity

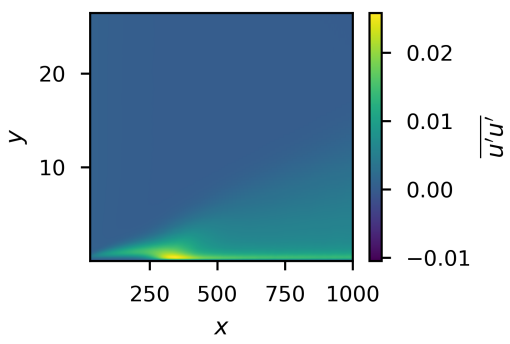


(b) Wall normal velocity

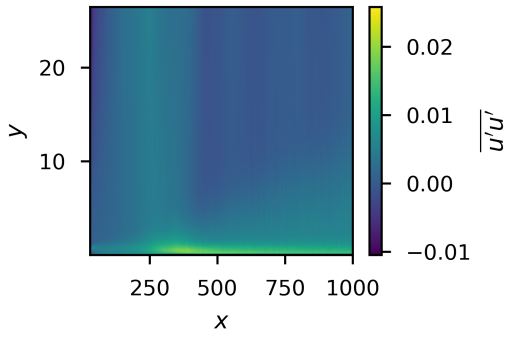


(c) Pressure

Figure 1: Steady state and averaged base flow of transitional boundary layer from JHTDB. The white dashed line indicted the edge of the boundary layer based on a streamwise velocity that is 99% of the freestream velocity.

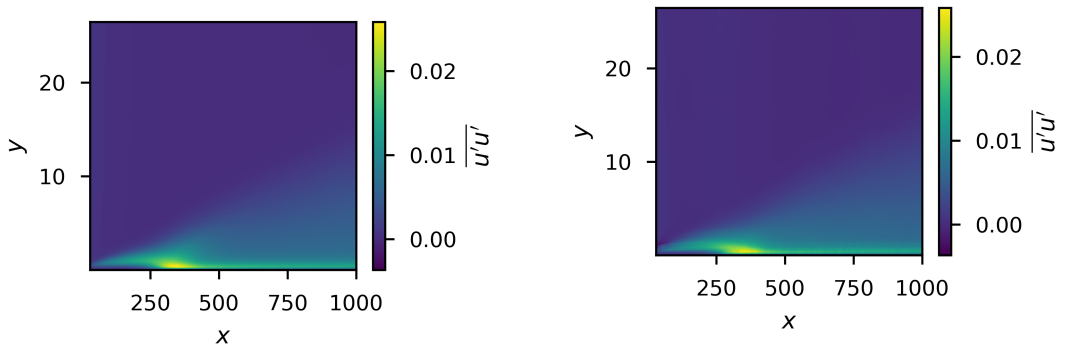


(a) JHTDB data



(b) Linear regression model that was trained on 70% of the JHTDB data

Figure 2: Streamwise component of the averaged fluctuation quantities modeled by linear regression and compared with the actual quantities



(a) JHTDB data

(b) Neural network model that was trained on 70% of the JHTDB data

Figure 3: Streamwise component of the averaged fluctuation quantities predicted by the neural network model that was trained on 70% of the JHTDB data. We see here near identical results compared to Fig. **??** where large deviations were visually observed.

2.2 Neural network

A small neural network was used here as well using the same inputs and outputs as the previous linear regression shown in Section 2.1

The neural network is defined with one hidden layer of 64 nodes and an activation function of ReLU. This led to a much lower mean squared error loss on the training data, suggesting that the more complicated network had reduced some bias. The preliminary results for trained neural network model is shown in Fig. 3. These results show near identical results for the model. This is promising and further error analysis will be demonstrated.

Further investigation into the neural network and checking other models is a matter of future work.

3 Future outlook for project

Further error analysis of the results shown will be presented in the project. Incorporating more training sets than the one simulation presented here will be conducted. Simulations run at the Center for Turbulence Research and other simulations from other repositories such as ERCOFTAC (`http://cfd.mace.manchester.ac.uk/ercoftac/classif.html#s-cf`) database will be incorporated. Additionally, other models and quantities will be analyzed in a systematic manner to come up with the best model for these simulations. If time allows, further optimization on inputs and outputs will be conducted. For example, it may be useful to output labels that contain the gradients of the averaged fluctuation quantities, thus further increasing the output labels may help with the model prediction.