Using Machine Learning for formulating new wall functions for Large Eddy Simulation: A First Attempt

L. Davidson
Div. of Fluid Dynamics
Mechanics and Maritime Sciences (M2)
Chalmers University of Technology, Gothenburg, Sweden

Sketch of support vector regression (SVR). Data point inside (●) and outside (■) the tube (gray area). The gray dashed line is the hyperplane (regression plane) predicted by \text{svrLINEAR}.

Colored surface is the hyperplane. I get non-physical (negative) friction velocities, $u_\tau$, predicted by \text{svrLINEAR} because I use too small “slack” (i.e. too large $C$). Thick blue line: $u_\tau = 0$. 
Abstract

Machine learning is used for developing wall functions for Large Eddy Simulations (LES). I use Direct Numerical Simulation (DNS) of fully-developed channel flow at frictional Reynolds number of 800 to create a database. This database is used as a training set for the machine learning method (support vector regression). The input data (i.e. the influence parameters) are the local Reynolds number, the non-dimensional velocity gradient and the time-averaged $y^+$ value. The machine learning method is trained to predict the wall shear stress.

The support vector regression methods in Python are used. The trained machine learning model is saved to disk and it is subsequently uploaded into the Python CFD code pyCALC-LES (Davidson, 2021). LES is carried out on coarse – and semi-course – near-wall meshes and the wall-shear stress is predicted using the developed machine learning models.
## Contents

1 Introduction 4

2 Direct Numerical Simulations (DNS) 5

3 Large Eddy Simulation (LES) 7

4 Standard wall functions 8

5 The machine learning methods 9
   5.1 Python code .............................................. 12

6 Results 14
   6.1 $Re_T = 2000$ .................................................. 14
   6.2 $Re_T = 5200$ .................................................. 17
   6.3 $Re_T = 5200$ with a stretching of $1.08$ .................. 19
   6.4 $Re_T = 5200$ with a stretching of $1.11$ .................. 19

7 Conclusions 22
1. Introduction

Machine learning is a method where known data are used for teaching the algorithm to classify a set of data. The data may be photographs where the machine learning algorithm should recognize, for example, traffic lights or traffic signs (Rao and Desai, 2021). Another example may be ECG signals where the machine learning algorithm should recognize certain unhealthy conditions of the heart (Lindholm et al., 2022). A third example is detecting fraud for credit card payments (Rachana et al., 2021). Machine learning methods such as Support Vector Machines (SVM) and neural networks are often used for solving this type of problems.

The examples above are classification problems using supervised learning (i.e. learning to recognize a traffic light, an unhealthy heart, learn what a customers usual credit card payment looks like). However, in the present work input and output are numerical values. In this case, machine learning in the form of regression methods should be used (Lindholm et al., 2022); I will use support vector regression (SVR) methods available in Python.

In SVR a regression mult-dimensional “surface” is created which has as many dimensions as number of influence parameters (in the present work I use one, two
2. Direct Numerical Simulations (DNS)

To create a database which can be used for training the SVR I will carry out a DNS of fully-developed channel flow. The Navier-Stokes equations read

$$\frac{\partial v_i}{\partial x_i} = 0$$  \hspace{1cm} (1)

$$\frac{\partial v_i}{\partial t} + \frac{\partial}{\partial x_j} (v_i v_j) = -\frac{\partial p}{\partial x_i} + \nu \frac{\partial^2 v_i}{\partial x_j \partial x_j}$$  \hspace{1cm} (2)

or three influence parameters). Let’s make a simple example. In Fig. 1 there is one influence parameter, $x$, and one parameter to predict, $y$. Two main input parameters may be given to the SVR methods. The first is $\epsilon$ which determines the width of the tube around the hyperplane. Points that lie inside this tube are considered as correct predictions and are not penalized by the algorithm. The support vectors are the points that lie outside the tube. The second parameter given to SVR models is the $C$ value. It controls the “slack” ($\xi$), see Fig. 1, which is the distance to points outside the tube. If $C$ is increased the size of the tube is increased so that some or all of the data points are located inside the tube. It will be shown in Section 5 that the parameter $C$ may have a large influence on the form of the hyperplane.

There are not many studies in the literature on machine-learning for improving wall functions. In (Tieghi et al., 2020) they use a time-averaged high-fidelity LES simulation which is used to train a neural network for improving the predicted modeled turbulent kinetic used in wall functions in RANS. Ling et al. (2017) use neural network to improve the predicted wall pressure to be used in fluid-structure interactions. They target it the wall pressure spectrum and the input parameters are the pressure power spectra above the wall. In (Dominique et al., 2022) they use neural network to predict the wall pressure spectra. Their input data include are boundary-layer thicknesses (physical, displacement and momentum), streamwise pressure gradient and wall shear stress which are taken from experiments and high-fidelity DNS/LES in the literature.

1A hyperplane is a plane whose number of dimension is the same the number of influence parameters. For example, a two-dimensional hyperplane has two influence parameters.
2. Direct Numerical Simulations (DNS)

(a) Velocity.

(b) $\langle u' u' \rangle$.

(c) $\langle v' v' \rangle$.

(d) $\langle w' w' \rangle$.

Figure 4: Channel flow at $Re_\tau = 800$. : Present DNS; : DNS data (Tanahashi et al., 2004).
3. Large Eddy Simulation (LES)

### Table 1: DNS database for high \( y^+ \) wall functions.

<table>
<thead>
<tr>
<th>Location</th>
<th>( \langle \Delta y^+ \rangle )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location 1</td>
<td>5</td>
</tr>
<tr>
<td>Location 2</td>
<td>13</td>
</tr>
<tr>
<td>Location 3</td>
<td>24</td>
</tr>
<tr>
<td>Location 4</td>
<td>39</td>
</tr>
<tr>
<td>Location 5</td>
<td>58</td>
</tr>
<tr>
<td>Location 6</td>
<td>85</td>
</tr>
</tbody>
</table>

\( \langle \cdot \rangle \) denotes average in time, \( x_1 \) and \( x_3 \)).

An incompressible, finite volume code is used (Davidson and Peng, 2003; Davidson, 2018) to solve Eqs. 1 and 2. The convective terms in the momentum equations are discretized using central differencing. The Crank-Nicolson scheme is used for time discretization of all equations. The numerical procedure is based on an implicit, fractional step technique with a multigrid pressure Poisson solver (Emvin, 1997) and a non-staggered grid arrangement.

The size of the channel is \( x_{\text{max}} = 2\pi \) (streamwise, \( x \) or \( x_1 \)), \( y_{\text{max}} = 2 \) (wall normal, \( y \) or \( x_2 \)) and \( z_{\text{max}} = 1.6 \) (spanwise, \( z \) or \( x_3 \)). The mesh has \( 386 \times 258 \times 386 \) and the Reynolds number is 800 based on the friction velocity, \( \langle u_r \rangle \) (\( \langle \cdot \rangle \) denotes average in time, \( x_1 \) and \( x_3 \)), and the half-channel width, \( \delta \).

Figure 4 presents comparison of the predicted velocity field and RSM fluctuations with DNS by Tanahashi et al. (2004) and – as can be seen – the agreement is very good.

When creating the instantaneous velocity, \( \bar{U} \), for the database, I integrate over a distance in \( y \) direction (see Fig. 3). The streamwise velocities for the 1\(^{st} \), 2\(^{nd} \) and 3\(^{rd} \) cells are computed as

\[
\begin{align*}
\bar{U}_{1st} &= \frac{1}{2\Delta y} \int_{0}^{2\Delta y} u dy, \\
\bar{U}_{2nd} &= \frac{1}{2\Delta y} \int_{2\Delta y}^{4\Delta y} u dy, \\
\bar{U}_{3rd} &= \frac{1}{2\Delta y} \int_{4\Delta y}^{6\Delta y} u dy
\end{align*}
\]

where \( \Delta y \) is representative of a typical cell size in a LES simulation using wall functions. The six locations of the first cell are given in Table 1. The locations of the second and third cells are shown in Fig. 3.

### 3 Large Eddy Simulation (LES)

The LES equations read

\[
\frac{\partial \bar{v}_i}{\partial x_i} = 0
\]

\[
\frac{\partial \bar{v}_i}{\partial t} + \frac{\partial}{\partial x_j} (\bar{v}_i \bar{v}_j) = -\frac{\partial \bar{p}}{\partial x_i} + \frac{\partial}{\partial x_j} \left[ \nu + \nu_{\text{sgs}} \frac{\partial \bar{v}_i}{\partial x_j} \right]
\]

\( \nu_{\text{sgs}} \) denotes the subgrid scale viscosity.
4. Standard wall functions

The machine-learning wall functions will be compared to the standard wall functions which are based on log-law, i.e.

\[ \frac{\bar{v}_1}{u_\tau} = \frac{1}{\kappa} \ln \left( \frac{E u_\tau \Delta y}{\nu} \right) \equiv \frac{1}{\kappa} \ln \left( E y^+ \right), \quad E = 9.0 \tag{4} \]

where \( \Delta y \) is given by Fig. 3 and subscript \( P \) denotes wall-adjacent cell. I compute the friction velocity from Equation 4 as

\[ u_\tau = \frac{\kappa \bar{v}_{1,P}}{\ln(E u_\tau \Delta y / \nu)} \tag{5} \]
which is solved by iterating a couple of times. If \( y^+ < 11.63 \), the linear law is used

\[
\frac{\overline{v_1}}{\nu \left( \frac{\partial \overline{\tilde{v}_1}}{\partial x_2} \bigg|_{y=0} \right)^{1/2}}
\]

(6)

5 The machine learning methods

As indicated in the introduction, I will use SVR (Support Vector Regression). Two different packages are used, \texttt{svr} and \texttt{svrLINEAR}, both available in Python. The former method is non-linear and the latter is linear. The machine learning method consists of a learning part and a testing part. In the learning part, the machine learning method is trained and in the testing part it is tested.

First, I need to determine which input variable (influence parameters) that should be used. In standard wall functions, the input parameters are wall-parallel velocity, \( \overline{v}_1 \), and the non-dimensional wall distance, \( y^+ \) (which includes the friction velocity, \( u_\tau \)); the output is the friction velocity. Hence, the friction is both input and output. In a machine learning method, it is probably not a good idea to let a parameter be included in both an input and output parameter. In order to make the machine learning method as general as possible, the input variables should be non-dimensional. The following influence (i.e. input) parameters are evaluated:

- the local Reynolds number, \( \frac{\overline{U}_{1st} \Delta y}{\nu} \)
- the velocity gradient, \( \frac{\partial U}{\partial y} = \frac{\overline{U}_{2nd} - \overline{U}_{1st}}{2 \Delta y} \); it is made non-dimensional by \( \nu \) and \( \overline{U}_{1st} \) as
  \[
  \left( \frac{\partial U}{\partial y} \right)_{non} = \frac{\partial \overline{U}}{\partial y} \frac{\nu}{\overline{U}_{2nd}^2}
  \]
  (7)
- the averaged non-dimensional wall distance, \( \frac{\langle u_\tau \rangle \Delta y}{\nu} \)

where \( \overline{U}_{1st} \) and \( \overline{U}_{2nd} \) are computed from Eq. 3. It may be noted that the velocity at the third point, see Fig. 3, is not used when computing these three input parameters.

Next, I will train the machine learning method using the DNS data created in Section 2. I start with \texttt{svrLINEAR}. 100 000 independent samples of \( \overline{U}_{1st} \) and \( \overline{U}_{2nd} \) are used at each of the six locations, see Table 1 and Fig. 3 together with the output parameter, \( u_\tau \). I have also evaluated to use 10 000 samples and I get identical results. I could probably use even fewer samples. I pick 80% of the data randomly and define that as the training set. The remaining 20% is then used for testing, i.e. predicting.

Figure 5 shows predicted friction friction velocities and the hyperplanes for \texttt{svrLINEAR} using the velocity gradient and local Reynolds number as influence (i.e. input) parameters for different \( C \) and \( \epsilon \) (these two parameters determines the
5. The machine learning methods

(a) $\epsilon = 0.01, C = 0.01$.
(b) $\epsilon = 0.001, C = 1$.
(c) $\epsilon = 0.001, C = 0.01$.

Figure 5: Predicted friction velocity with svrLINEAR using high $y^+$ data. •: predicted friction velocity; 3D surface: hyperplane colored with $u_\tau$. 
5. The machine learning methods

Figure 6: Predicted friction velocity with $\text{svr}$ using low $y^+$ data, see Table 1. 3D view. •: Predicted friction velocity; 3D surface: hyperplane colored with $u_T$.

(a) $\epsilon = 0.1$, $C = 0.1$. Thick blue line: $u_T = 0$.

(b) $\epsilon = 0.1$, $C = 0.01$.

Figure 7: Predicted friction velocity with $\text{svr}$ using low $y^+$ data, see Table 2. 2D view. •: Predicted friction velocity; colored surface: $u_T < 0$.

(a) $\epsilon = 0.1$, $C = 0.1$.

(b) $\epsilon = 0.1$, $C = 0.01$. 


width of the tube around hyperplane and the “slack”, respectively, see Section 1). Please recall that svrLINEAR is a linear method which gives a linear hyperplane. As can be seen in Fig.5, the predictions are not sensitive to $C$ and $\epsilon$. The RMS error, $e$, between the predicted friction velocity and the DNS database

$$ e = \frac{\text{std}(u_{\tau,pred} - u_{\tau,DNS})}{\sqrt[2]{\text{mean}(u_{\tau,pred}^2)} \text{mean}(u_{\tau,DNS}^2)}^{1/2} $$

(8)

is slightly below 20% for all svr and svrLINEAR cases (also for the standard wall functions). This error is rather high and it is discussed in Section 7. For svrLINEAR, I choose to use $C = 1$ and $\epsilon = 0.001$.

Figure 6 presents the friction velocities and the hyperplanes for two sets of $C$ and $\epsilon$, namely $\epsilon = 0.1$ and $C = 0.1$ (Fig. 7a) and $\epsilon = 0.1$ and $C = 0.01$ (Fig. 7b). First, we note that the hyperplanes are no longer linear since svr is a non-linear method. Second, the predicted friction velocities are strongly negative for $C = 0.1$ which is clearly unphysical (the smallest instantaneous friction velocity in the DNS database is $u_{\tau,min} = 0.11$). Decreasing $C$ by a factor of 10 gives a much more reasonable hyperplane. As mentioned above, the RMS error (Eq. 8) is slightly below 20% also for this case which at first is somewhat surprising. However, when looking at the data in a 2D view we get the explanation: the area spanned by $(Re$ and $(\partial \bar{U}/\partial y)_{non})$ in which the hyperplane for $u_{\tau}$ goes negative does not include any DNS values, see Fig. 7a. Figure 7b does not show any negative values of the hyperplane which we already observed in Fig. 7b. I choose to use $C = 0.1$ and $\epsilon = 0.01$.

It may be noted that Fig. 7 indicates a strong correlation between $Re$ and $(\partial \bar{U}/\partial y)_{non}$. This is indeed the case. It turns out that the correlation coefficient (Eq. 10) is $-0.45$; this is further discussed in Section 7.

### 5.1 Python code

Here I present some of the Python commands. First, I scale the input data

```python
scaler_dudy=StandardScaler()
scaler_re=StandardScaler()
scaler_yplus=StandardScaler()
dudy_in=scaler_dudy.fit_transform(dudy_in)
re_in=scaler_re.fit_transform(re_in)
yplus_in=scaler_yplus.fit_transform(yplus_in)
```

Then I put the input and output data on generic form

```python
y=ustar_out # output
X=np.zeros((n_svr,3)) # output
X[:,0]=re_in[:,0] # 1st indata
X[:,1]=dudy_in[:,0] # 2nd indata
X[:,2]=yplus_in[:,0] # 3rd indata
```
5.1. Python code

Then I choose the model

```python
c=1
eps=0.001
model = LinearSVR(max_iter=10000, epsilon=eps, C=C)
```

I train the model

```python
svr = model.fit(X, y.flatten())
```

Now comes the testing part. I scale my test data

```python
# Use MinMax scaling
dudy_in_test=dudy_in_test.reshape(-1, 1)
re_in_test=re_in_test.reshape(-1, 1)
yplus_in_test=yplus_in_test.reshape(-1, 1)
dudy_in_test=scaler_dudy.transform(dudy_in_test)
re_in_test=scaler_re.transform(re_in_test)
yplus_in_test=scaler_yplus.transform(yplus_in_test)
```

Then I predict friction velocities

```python
y_svr = model.predict(X_test)
```

Now I want to find how accurate my predictions are. I compare with DNS data

```python
# find difference
scale=np.mean(y_svr)*np.mean(ustar_out_test)
ustar_rms=np.std(y_svr-ustar_out_test)/scale
print('ustar_rms',ustar_rms)
```

Finally I store the model on disk.

```python
# save the model to disk
dump(model, 'model-svrLINEAR.bin')
dump(scaler_re,'model-svrLINEAR_scaler-u.bin')
dump(scaler_dudy,'model-svrLINEAR_scaler-dudy.bin')
dump(scaler_yplus,'model-svrLINEAR_scaler-yplus.bin')
np.savetxt('min-max-model-svrLINEAR.txt', 
[dudy_min,dudy_max,re_max,re_min,yplus_max,yplus_min])
```

When I do the LES simulations with **pyCALC-LES** I load the model. Then when I predict the friction velocities, I do as in the testing phase above, except that

```python
dudy_in_test
re_in_test
yplus_in_test
```
6. Results

Here I evaluate \texttt{svr} and \texttt{svrLINEAR} in fully-developed channel flow at two different Reynolds numbers, namely $Re_\tau = 2000$ and $Re_\tau = 5200$. For the lower Reynolds number I use a proper wall functions grid (i.e. wall-adjacent cells at $\langle y^+ \rangle = 55$) whereas I for the higher Reynolds also use grids i.e. which are not really suitable for wall functions (wall-adjacent cells at $\langle y^+ \rangle = 15$ and $\langle y^+ \rangle = 8$).

The extent of the domain in the $x$ and $z$ direction is $3.2$ and $1.6$, respectively, covered by $32$ cells in each direction. The Python code \texttt{pyCALC-LES} is used for all simulations together with the Smagorinsky model, see Section 3.

6.1 $Re_\tau = 2000$

36 cells ($N_y = 36$, constant grid spacing) are used in the wall-normal direction which gives $\langle y^+ \rangle = 55$ for the wall-adjacent cells. The DNS database defined in Table 1 is used. Figure 8 presents the predicted velocities. The \texttt{svrLINEAR} – using one, two or three influence parameters (see p. 9), \texttt{svr} and the standard wall functions are compared with Reichardt’s law

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

(9)

The agreement is excellent for all cases except for \texttt{svrLINEAR} when only one influence parameter ($Re$) is used. The resolved stresses are compared with DNS in Fig. 9 and the agreement is reasonable.

The conclusion for this test case is that both \texttt{svr} and \texttt{svrLINEAR} (and the standard wall functions) give excellent agreement with benchmark data. However, the required CPU time for \texttt{svr} and \texttt{svrLINEAR} differ greatly. Table 3 presents the elapsed time. As can be seen, the required time for the \texttt{svr} is five orders of

<table>
<thead>
<tr>
<th></th>
<th>LES with \texttt{svr}</th>
<th>LES with \texttt{svrLINEAR}</th>
</tr>
</thead>
<tbody>
<tr>
<td>one time step</td>
<td>1.64</td>
<td>0.045</td>
</tr>
<tr>
<td>\texttt{svr} or \texttt{svrLINEAR} prediction</td>
<td>1.6</td>
<td>$1.2 \cdot 10^{-4}$</td>
</tr>
</tbody>
</table>

Table 3: Elapsed time in seconds.

at both walls are given by \texttt{pyCALC-LES}.

As you see above, I store $\min$ and $\max$ of all indata parameters. As a precaution, I limit the indata given by \texttt{pyCALC-LES} so that they stay within the limits of the DNS database, e.g.

\begin{verbatim}
dudy=np.minimum(dudy, dudy_max)
dudy=np.maximum(dudy, dudy_min)
\end{verbatim}
6.1. $Re_\tau = 2000$

(a) Standard wall functions.

(b) svrLINEAR. With $\langle y^+ \rangle$, with $\left( \frac{\partial \bar{u}}{\partial y} \right)_{non}$

(c) svrLINEAR. With $\langle y^+ \rangle$, no $\left( \frac{\partial \bar{u}}{\partial y} \right)_{non}$

(d) svrLINEAR. No $\langle y^+ \rangle$, with $\left( \frac{\partial \bar{u}}{\partial y} \right)_{non}$

(e) svrLINEAR. No $\langle y^+ \rangle$, no $\left( \frac{\partial \bar{u}}{\partial y} \right)_{non}$

(f) svr. No $\langle y^+ \rangle$, with $\left( \frac{\partial \bar{u}}{\partial y} \right)_{non}$

Figure 8: Channel flow. $Re_\tau = 2000$. Velocity. $N_y = 36$. $Re$ is used as influence parameters for all svr and svrLINEAR cases. \textbullet{}: Reichardt’s law, Eq. 9.
6.1. $Re_\tau = 2000$

Figure 9: Channel flow. $Re_\tau = 2000$. $N_y = 36$. Reynolds stresses. $Re$, $\langle y^+ \rangle$ and $(\frac{\partial U}{\partial y})_{non}$ are used as influence parameters for svrLINEAR.
6.2. $Re_\tau = 5200$

(a) Standard wall functions.

(b) svrLINEAR.

Figure 10: Channel flow. $Re_\tau = 5,200$. Velocity. $N_y = 70$. All three influence parameters are used for svrLINEAR. •: Reichardt’s law, Eq. 9

Magnitude larger than svrLINEAR. As I mentioned in Section 5, I use 10 000 samples in each of the six locations (i.e. 10 000). This number could most likely be reduced. But let’s wait until we have looked at the results of the other channel flow simulations below.

6.2 $Re_\tau = 5200$

70 cells ($N_y = 70$, constant grid spacing) are used in the wall-normal direction which gives $\langle y^+ \rangle = 74$ for the wall-adjacent cells. The DNS database defined in Table 1 is used. Figure 10 presents the predicted velocity profile and the agreement with Reichardt’s laws is very good (slightly better for svr than the standard wall function). The Reynolds stresses are shown in Fig. 11 and both methods give fairly good agreement with DNS.
6.2. $Re_\tau = 5200$

(a) Standard wall functions.

(b) svrLINEAR with $\langle y^+ \rangle$, with $\left( \frac{\partial U}{\partial y} \right)_{non}$.

(c) svrLINEAR with $\langle y^+ \rangle$, no $\left( \frac{\partial U}{\partial y} \right)_{non}$.

(d) svr with $\langle y^+ \rangle$, no $\left( \frac{\partial U}{\partial y} \right)_{non}$.

Figure 12: Channel flow. $Re_\tau = 5200$. $N_y = 70$. Velocity. Stretching in $y$ direction: 1.08. $Re$ is used as influence parameters for all svr and svrLINEAR cases. ■: Reichardt’s law, Eq. 9.
6.3. \( Re = 5,200 \) with a stretching of 1.08.

Here I will present LES simulation at \( Re = 5,200 \) with \( N_y = 70 \) and a stretching of the grid in the wall-normal direction of 1.08. This gives \( \langle y^+ \rangle = 15 \) at the wall-adjacent cells. The DNS database defined in Table 2 is used.

Figure 12 show predicted velocities with the standard wall functions, \texttt{svr} and \texttt{svrLINEAR}. \texttt{svrLINEAR} including \( \left( \frac{\partial \bar{U}}{\partial y} \right)_{\text{non}} \) (Fig. 12b) gives poorest agreement with Reichardt’s law. \texttt{svrLINEAR} with \( Re \) and \( \langle y^+ \rangle \) shows best performance (Fig. 12c), slightly better than the standard wall functions. Figure 13 compares the normal stresses with the DNS stresses. The stresses obtained with both methods gives too large streamwise stresses and too small wall-normal and spanwise stresses.

6.4. \( Re = 5,200 \) with a stretching of 1.11.

Here I present the last LES simulations; the Reynolds number is \( Re = 5,200 \) with \( N_y = 70 \) and a stretching of the grid in the wall-normal direction of 1.11. This moves the wall-adjacent cells even closer to the wall so that \( \langle y^+ \rangle = 8 \). The DNS database defined in Table 2 is used. The predicted velocities are shown in Fig. 14; The best agreement is shown by \texttt{svrLINEAR} with \( Re \) and \( \langle y^+ \rangle \) as influence parameters (Fig. 14c); standard wall functions give almost as good agreement (Fig. 14a). The non-linear \texttt{svr} gives poor agreement. The Reynolds stresses (Fig. 15) predicted show the same pattern as in Fig. 13: too large streamwise stresses (slightly larger for \texttt{svrLINEAR}) and \( \langle \bar{v}^' \bar{v}^' \rangle \) and \( \langle \bar{w}^' \bar{w}^' \rangle \) too small.
6.4. $Re_\tau = 5200$ with a stretching of 1.11.

(a) Standard wall functions.
(b) svrLINEAR with $(y^+)$, with $\left(\frac{\partial U}{\partial y}\right)_{non}$.
(c) svrLINEAR with $(y^+)$, no $\left(\frac{\partial U}{\partial y}\right)_{non}$.
(d) svr. With $(y^+)$, no $\left(\frac{\partial U}{\partial y}\right)_{non}$.

Figure 14: Channel flow. $Re_\tau = 5200$. $N_y = 70$. Velocity. Stretching in y direction: 1.11. $Re$ is used as influence parameters for all svr and svrLINEAR cases. •: Reichardt’s law, Eq. 9.

(a) Standard wall functions.
(b) svrLINEAR with $(y^+)$, no $\left(\frac{\partial U}{\partial y}\right)_{non}$.

Figure 15: Channel flow. $Re_\tau = 5200$. $N_y = 70$. Stretching in y direction: 1.11. Reynolds stresses. •: DNS data (Lee and Moser, 2015)
6.4. \( \text{Re}_\tau = 5\,200 \) with a stretching of 1.11.

\[ \tau = 5\,200 \]

with a stretching of 1.11.

(a) \( \text{svr} \) with \( \text{Re} \) as influence parameter. Location 6

(b) \( \text{svr} \) with \( \text{Re} \) and \( (\partial \bar{U}/\partial y)_{\text{non}} \) as influence parameter. Location 6

(c) \( \text{svrLINEAR} \) with \( \text{Re} \) and \( (\partial \bar{U}/\partial y)_{\text{non}} \) as influence parameter. Location 6

(d) \( \text{svrLINEAR} \) with \( \text{Re} \) and \( (\partial \bar{U}/\partial y)_{\text{non}} \) as influence parameter. Location 1

Figure 16: Testing of \( \text{svrLINEAR} \) and \( \text{svr} \) at Location 1 and 6, see Table 1.

- \( u_\tau \) from \( \text{svrLINEAR} \) or \( \text{svr} \);
- \( u_\tau \) from the standard wall functions;
- DNS database;
- \( \bar{U}_{1st} \);
- \( (\partial \bar{U}_{1st}/\partial y) \).

Figure 17: Averaged streamwise velocity from the DNS database and predicted by \( \text{svr} \).
7 Conclusions

Machine learning methods have been used to develop new wall functions. Support Vector Regression methods are employed, both non-linear, svr, and linear, svrLINEAR, which are available in Python. A DNS database created from fully-developed channel flow has been used for training and testing the svr and the svrLINEAR. When the wall-adjacent cells are located in the log-law region, both svr and svrLINEAR give perfect agreement with the Reichardt’s law (as does the standard wall functions). When the wall-adjacent cells are located in the buffer layer, svrLINEAR gives better agreement with the Reichardt’s law than the standard wall functions, but the agreement is not perfect. It is found that svrLINEAR gives better agreement with the Reichardt’s law than svr, in some cases much better. Furthermore, svr requires computer times that is orders of magnitude larger than svrLINEAR; it is found that the svr may dominate the CPU/GPU time for one time step, see Table 3. This problem could probably be reduced by using a smaller training set, but since svrLINEAR anyway is more accurate than svr, no such attempt has been made.

I find in Section 5 that the error (Eq. 8) of all svr and svrLINEAR methods – and the standard wall functions – is close to 20%. That is rather high. The predicted friction velocity for 20 time steps using svrLINEAR and svr are compared with the DNS database at Location 6, see Table 1. I also show $u_t$ obtained from the standard wall functions. Start by looking at Fig. 16a. We find that $u_t$ obtained from the wall functions closely follows the variation of $\bar{U}_{1,st}$. This applies also for the svr predicted friction velocity which is not surprising since $Re$ (which is linearly related to $\bar{U}_{1,st}$) is used as influence parameter. Next, we turn to Fig. 16b where the influence parameter $\left(\frac{\partial C}{\partial y}\right)_{non}$ is also used. It is somewhat surprising that the prediction by svr is virtually identical to that in Fig. 16a (there is a small difference at time step 16). Closer inspection of Fig. 16b reveals that there is a negative correlation between $\bar{U}_{1,st}$ and $\partial \bar{U}_{1,st}/\partial y$. Computing the correlation coefficient, $e_c$,

$$
e_c \left(\bar{U}_{1,st}, \partial \bar{U}_{1,st}/\partial y\right) = \frac{\text{dot} \left(\bar{U}_{1,st}', \partial \bar{U}_{1,st}'/\partial y\right)}{\text{mean} \left(\bar{U}_{1,st}'\right)^{1/2} \text{mean} \left(\left(\partial \bar{U}_{1,st}'/\partial y\right)^2\right)^{1/2}} \quad (10)$$

(dot is Python’s element-wise multiplication command) gives $e_c = -0.45$. The corresponding correlation coefficient between $\bar{U}_{1,st}$ and the $u_t$ predicted by svr is 0.41. Hence, it can be concluded that since $\bar{U}_{1,st}$ and $\partial \bar{U}_{1,st}/\partial y$ are fairly strongly correlated it does not make sense to use both as influence parameters. Now, let’s look at the third figure (Fig. 16c) we find that the friction velocity by svrLINEAR is very similar to svr (Fig. 16b). A general observation in Fig. 16 is that the agreement between DNS data and svr (and svrLINEAR) in Fig. 16 is poor. Why? We can see that the time scales of the friction velocity from the DNS database is much smaller than in those by svr and svrLINEAR. The reason is that the friction velocity in the DNS database is computed over only one DNS cell for which
\[ \Delta x_{DNS}^+ = 13 \] and \[ \Delta z_{DNS}^+ = 6.5 \] which should be compared with \[ 2\Delta y = 170 \] at Location 6 (see Fig. 3 and Table 1). However, the predicted friction velocity with \texttt{svrLINEAR} at Location 1 (Fig. 16d) for which \[ 2\Delta y = 10 \] (which is close to \[ \Delta x_{DNS}^+ \] and \[ \Delta z_{DNS}^+ \]) agrees well with the DNS database. This is a strong indication that the friction velocity in the DNS database should be integrated over \[ \Delta x \approx 2\Delta y \] and \[ \Delta z \approx 2\Delta y \]. This means that the friction velocity in the DNS database will be different for the six different locations, see Tables 1 and 2.

The averaged filtered velocities (see Eqs. 3 and 7) are shown in Fig. 17. It is seen that \[ \bar{U}_{1st} \] does not agree with the DNS velocity profile whereas the agreement for \[ \bar{U}_{2nd} \] and \[ \bar{U}_{3rd} \] is perfect. The reason is that the gradient of the DNS data near the wall is very large and since \[ \bar{U}_{1st} \] is computed by integrating from the wall to \[ 2\Delta y \] the integrated value differs much from the local (i.e. DNS) value. Maybe it is best to avoid using the first cell, \[ \bar{U}_{1st} \], and instead rely on \[ \bar{U}_{2nd} \] and \[ \bar{U}_{3rd} \] as proposed by Kawai1 and Larsson (2012); Mukha et al. (2021).

I have here presented – as indicated in the title – my first attempt. In my next attempt I will create the DNS database by integrating the friction velocities over suitable \[ \Delta x \] and \[ \Delta z \] (I’ve already started). It would also be interesting to develop machine-learning wall functions for heat transfer. To extend the method to recirculating flow is the next – but very challenging – step.

Acknowledgments

This study was partly financed by Strategic research project on Chalmers on hydro- and aerodynamics.

The computations were enabled by resources provided by the Swedish National Infrastructure for Computing (SNIC) at NSC partially funded by the Swedish Research Council through grant agreement no. 2018-05973.

References


7. Conclusions


M. Tanahashi, S.-J. Kang, T. Miyamoto, S. Shiokawa, and T. Miyauchi. Scaling law of fine scale eddies in turbulent channel flows up to \( re_\tau = \)

---

Note: The text is truncated at the end and seems incomplete. The full text continues with more references and details beyond the visible part.