T MTF271, Assignment 2a: DES, DDES and SAS

In this exercise you will evaluate PANS (Section 23), DES (Section 20), DDES (Section 20.3) and SAS (Section 22). You will use data from the IDD-PANS model [175] (see also Section 23.4). The flow is fully developed channel flow, see Fig. R.1. The Re number based on the friction velocity and the half channel width is $Re_\tau = u_\tau h/\nu = 5\ 200$. The matching line may be defined by when f_k in Eq. 47 in [175] (see also Eq. 23.79) falls down to, say , 0.4. It turns out that it occurs at gridline number 28 at which $x_2^+ \simeq 270$, $x_2/\delta = 0.052$ (δ denotes half channel width).

A $32 \times 96 \times 32$ mesh has been used. The cell size in x_1 and x_3 directions are $\Delta x_1 = 0.1$ and $\Delta x_3 = 0.05$. Periodic boundary conditions were applied in the x_1 and x_3 direction (homogeneous directions). All data have been made non-dimensional by u_τ and ρ.

At the course www page you find data files with instantaneous flow fields (statistically independent). The data files include the instantaneous variables u (v_1), v (v_2), w (v_3), k and ε (made non-dimensional by u_τ and ρ).

You can use Matlab, Octave or Python. Both Octave and Python are open-source software. Octave is a Matlab clone. Many large Swedish industries prefer engineers to use Python instead of Matlab due to Matlab's high license fees.

Use one of the programs Python/Matlab/Octave to analyze the data. You find a Python/Matlab/Octave program at the www page which reads the data and computes the mean velocity. The data files are Matlab binary files.

You will also find a file with time history of u.

T.1 Time history

At the www page you find the files `u-time-history-i70.dat` and `w-time-history-i70.dat` with the time history of $\overline{v}_1$ and $\overline{v}_3$. Each file has six columns of $\overline{v}_1$ and $\overline{v}_3$ at five nodes (and time): $x_2/\delta = 0.0028$ (node 10) , $x_2/\delta = 0.0203$, (node 22) $x_2/\delta = 0.0364$, (node 26) $x_2/\delta = 0.0645$ (node 30) and $x_2/\delta = 0.20$ (node 38) . Hence, three nodes are located in the URANS region, and two nodes in the LES region. With $u_\tau = 1$ and $\nu = 1/Re_\tau = 1/5200$, this correspond to $x_2^+ = 15$, $x_2^+ = 105$, $x_2^+ = 189$, $x_2^+ = 336$ and $x_2^+ = 1040$, respectively. The sampling time step is $1.25E-3$ (every time step). Use the Python/Matlab program `pl_time_IDD_DES` to load and plot the time history of $\overline{v}_1$.

Recall that the velocities have been scaled with the friction velocity u_τ, and thus what you see is really $\overline{v}_1/u_\tau$. The time history of $\overline{v}_1$ at $x_2/\delta = 0.0203$ and $x_2/\delta = 0.0645$ are shown. To study the profiles in closer detail, use the `plt.axis` command (Python) or `axis` command (Matlab/Octave) to zoom-in.

Plot $\overline{v}_1$ for all five nodes. How does the time variation of $\overline{v}_1$ differ for different positions? Recall that the three points closest the wall are located in the URANS region and the other two are located in the LES region. In the URANS region the turbulent viscosity is much larger than in the LES region. How do you expect that the difference in ν_t affects the time history of $\overline{v}_1$?[10]. You should also keep in my mind that turbulent fluctuations are created by the mean flow gradient (i.e. the production). Small mean flow gradients means little or no creation of turbulence. Does the time history of $\overline{v}_1$ behave as you expect? What about $\overline{v}_3$?

[10] Hint: small scales destroys/kills the large scales

Compute the autocorrelation (see Section 10.2) of the five points. In Python, you do

```
u1_fluct=u1-np.mean(u1)
two=np.correlate(u1_fluct,u1_fluct,'full')
# find max
nmax=np.argmax(two)
# and its value
two_max=np.max(two)
# Pick the right half and normalize
two_sym_norm=two[nmax:]/two_max
```

and in Matlab/Octave

```
imax=500;
two_uu_1_mat=autocorr(u1,imax);
```

Plot the autocorrelation in Python

```
fig1 = plt.figure("Figure 1")
imax=500;
plt.plot(t[0:imax],two_sym_norm[0:imax],'b-')
plt.xlabel('t')
plt.ylabel('$B_{uu}$')
```

or in Matlab/Octave

```
plot(t(1:imax),two_uu_1_mat(1:imax),'linew',2)
xlabel('t')
ylabel('B_{uu}')
handle=gca
set(handle,'fontsi',[20])
```

Above we set the maximum separation in time to 500 samples. For large time separation you find that the autocorrelation oscillates around zero. This is numerical noise. You should carry out the integration in Eq. 10.11 only up to the point when the autocorrelation goes negative. If is does not go negative, terminate the integration at the time separation when the autocorrelation starts to increase.

Above I told you to use built-in functions (`correlate` or `autocorr`) to compute the auto correlation; the reason is that they are very fast. If you really want to understand what's going on you should use `for` loops. I have made an example in Section X.10.

Compute the integral timescale in Pythons

```
dt=t[1];
int_T_1=np.trapz(two_sym_norm)*dt;
```

or in Matlab/Octave

```
dt=t(1);
int_T_1=trapz(two_uu_1_mat)*dt;
```

Compute the autocorrelation and integral timescale also for the other four points. Do you see any difference between the points in the URANS region and the LES region? What did you expect?
Hint: small-scale turbulence breaks up/destroys large eddies.

T.2 Mean velocity profile

After having performed a DES, we want to look at the time-averaged results. Use the file `pl_uvw_IDD_PANS` to look at the mean velocity profiles. `pl_uvw_IDD_PANS` reads the instantaneous $\bar{v}_1$ field and performs an averaging in the homogeneous directions x_1 and x_3. The time averaged velocity profile is compared with the log profile (markers). There are 16 files with instantaneous values of $\bar{v}_1$ $\bar{v}_2$ and $\bar{v}_3$ which in the beginning of `pl_uvw_IDD_PANS` is merged into three 3D arrays, `u3d`, `v3d` and `w3d` of size $16 * (ni, nj, nk)$ (`nfiles=16`). If you want to speed-up the simulations when debugging/testing, you can use fewer files (e.g. `nfiles=4`).

T.3 Resolved stresses

We want to find out how much of the turbulence that has been resolved and how much that has been modelled. Compute first `vmean` (this quantity should be very small, but if you had infinite number of samples it would be zero). Now compute $\langle v_1' v_2' \rangle$. Here's an example how to do in Python:

```
uv=np.zeros(nj)
for k in range (0,nk):
for j in range (0,nj):
for i in range (0,ni):
   ufluct=u3d[i,j,k]-umean[j]
   vfluct=v3d[i,j,k]-vmean[j]
   wfluct=w3d[i,j,k]-wmean[j]
   uv[j]=uv[j]+ufluct*vfluct

uv=uv/ni/nk
```

and in Matlab

```
uv=zeros(nj,1);
for k=1:nk
for j=1:nj
for i=1:ni
   ufluct=u3d(i,j,k)-umean(j);
   vfluct=v3d(i,j,k)-vmean(j);
   uv(j)=uv(j)+ufluct*vfluct;
end
end
end
uv=uv/ni/nk;
```

A much faster way is to use Python's built-in function

```
uvmean1=np.mean((u3d-umean[None,:,None])*(v3d-vmean[None,:,None]), axis=(0,2))
```

or Matlab's built-in function

```
uv=mean(mean(((u3d-umean).*(v3d-vmean)),3),1);
```

Plot it in a new figure. Note that if you compare with DNS, it is the modeled plus resolved shear stress which should match the DNS data. Maybe you can do that comparison in Section T.5.

T.4 Turbulent kinetic energy

The resolved turbulent kinetic energy is defined as

$$k_{res} = 0.5\left(\langle v_1'^2\rangle + \langle v_2'^2\rangle + \langle v_3'^2\rangle\right)$$

Plot and compare the resolved and modelled turbulent kinetic energies. Note that the modelled turbulent kinetic energy, k (`te1_IDD_PANS.mat`, `te2_IDD_PANS.mat`, ...), can be downloaded from the www page and are loaded at the beginning of `pl_uvw_IDD_PANS`. Which is largest (modeled or resolved)? Which is largest in the URANS region and in the LES region, respectively? What about the sum? The magnitude of resolved and modelled turbulent kinetic energies is discussed on p. 115 in [167] in relation to Fig. 6 in [167].

T.5 The modelled turbulent shear stress

We have computed the resolved shear stress. Let's find the modelled shear stress. The modelled turbulent kinetic energy, k (file `te1_IDD_PANS.mat`, ...), and the dissipation, ε (file `eps1_IDD_PANS.mat`, ...) will be used. Recall that $\nu = 1/5200$. Compute the turbulent viscosity as $C_\mu f_\mu k^2/\varepsilon$ (AKN/PANS model), see Eq. 23.19.

Compute the modelled shear stress from the Boussinesq assumption

$$\tau_{12} = -2\nu_t \bar{s}_{12} = -\nu_t\left(\frac{\partial \overline{v}_1}{\partial x_2} + \frac{\partial \overline{v}_2}{\partial x_1}\right)$$

To compute the velocity gradient, use the Python command

```
dudx, dudy, dudz=np.gradient(u3d,dx,y,dz)
```

or the Matlab command

```
[dudy dudx dudz]=gradient(u3d,y,x,z);
```

Plot $\langle\tau_{12}\rangle$ and compare with the resolved shear stress (see Section T.3). Are they smooth across the interface? Is the resolved shear stress large in the URANS region? Should it be large? Why/why not?

T.6 Location of interface in DES and DDES

As mentioned above, the interface in the present simulations may be defined where $f_k \simeq 0.4$. Let's compare that with SA-DES and DDES.

In SA-DES, the interface is defined as the location where the wall distance is equal to $C_{DES}\Delta$ where $\Delta = \max\{\Delta_x, \Delta_y, \Delta_z\}$, see Eq. 20.3.

In SST-DES, the location of the interface is computed using k and ε. The switch of this model takes place according to Eq. 20.8. How does the location compare with SA-DES?

In DDES, the boundary layer is shielded with a damping function. In SST-DES, the shielding function (see Eq. 20.11) may be one of the blending functions, F_1 or F_2 (see Eq. 20.5). Let's use F_2 as the shielding function as in [79]. Does DDES work, i.e. does it make the model to be in RANS mode in the entire boundary layer?

T.7 SAS turbulent length scales

The SAS model is described in Section 22.

Compute the 1D von Kármán length scale defined as

$$L_{vK,1D} = \kappa \left| \frac{\partial \langle \overline{v}_1 \rangle / \partial x_2}{\partial^2 \langle \overline{v}_1 \rangle / \partial x_2^2} \right| \tag{T.1}$$

where $\kappa = 0.41$ is the von Kármán constant. Note that you should take the derivatives of the *averaged* $\overline{v}_1$ velocity, i.e. of $\langle \overline{v}_1 \rangle$. How does it behave, i.e. what is n in $\mathcal{O}(x_2^n)$ in the fully turbulent region? What should n be?[11]

When we're doing real 3D simulations, the first and second derivative must be defined in 3D. One way of defining the von Kármán length scale in 3D is [161, 162]

$$\begin{aligned} L_{vK,3D,inst} &= \kappa \left| \frac{S}{U''} \right| \\ S &= (2 \overline{s}_{ij} \overline{s}_{ij})^{0.5} \\ U'' &= \left(\frac{\partial^2 \overline{v}_i}{\partial x_j \partial x_j} \frac{\partial^2 \overline{v}_i}{\partial x_k \partial x_k} \right)^{0.5} \end{aligned} \tag{T.2}$$

Plot the von Kármán length scale using Eqs. T.2 and T.2. Compare them with Eq. T.1. What's the difference? What effect do the different length scales give for P_{SAS} (i.e. T_1 in Eq. 22.5?

Another way to compute the second derivative is [213]

$$U'' = \left(\frac{\partial^2 \overline{v}_i}{\partial x_j \partial x_k} \frac{\partial^2 \overline{v}_i}{\partial x_j \partial x_k} \right)^{0.5} \tag{T.3}$$

Plot and compare the von Kármán length scales using the second derivatives defined in Eqs. T.2 and T.3.

T.8 Anisotropic errors (optional)

Recently, Toosi & Johan Larsson [214] proposed a method for evaluating resolution. They propose to use a directionally small-scale (i.e., a directionally high-pass test filtered) field, $\overline{v}_i^{*(\mathbf{n})}$ defined as

$$\overline{v}_i^{*(\mathbf{n})} \equiv \overline{v} - \hat{\overline{v}}_i^{*(\mathbf{n})} \simeq \frac{-\Delta_{\mathbf{n}}^2}{4} \mathbf{n}^T \left(\nabla \nabla^T \overline{v}_i \right) \mathbf{n} \tag{T.4}$$

where $\mathbf{n}$ defines the direction along which the small-scale velocity is computed. The anisotropic error is then defined as

$$\mathcal{A}(\mathbf{n}) = \left(\left\langle \overline{v}_i^{*(\mathbf{n})} \overline{v}_i^{*(\mathbf{n})} \right\rangle \right)^{1/2} \tag{T.5}$$

In [214] they used the quantity as an indicator where the grid needed to be refined and in which direction. But it can also be used as a post-processing tool to evaluate if the grid is equally good (or bad) everywhere. Hence we want the error in Eq. T.5 to be as constant as possible in all grid direction and throughout the domain. Since our flow

[11] you will find a large peak at $y \simeq 2000$ because $\partial^2 U_{mean} / \partial y^2$ behave strangely due to too few samples

is homogeneous in x_1 and x_3 we need only to evaluate how inhomogeneous the error is in the x_2 direction. Furthermore, on structured grids we can compute Eq. T.4 using second velocity derivatives, i.e.

$$\bar{v}_i^{*(\mathbf{n})} \simeq -\frac{\Delta_{\mathbf{n}}^2}{4}\frac{\partial \bar{v}_i}{\partial x_n \partial x_n}, \text{ no summation over } n \tag{T.6}$$

where $\Delta_{\mathbf{n}}$ is equal to Δx, Δy or Δz. In order to take the anisotropic cells into account, the error should be multiplied by the cell volume [215]

$$\mathcal{A}(\mathbf{n})_V = \Delta x \Delta y \Delta z \left(\left\langle \bar{v}_i^{*(\mathbf{n})} \bar{v}_i^{*(\mathbf{n})} \right\rangle \right)^{1/2} \tag{T.7}$$

Plot $\mathcal{A}(\mathbf{n})_V$ and find out how constant and isotropic it is. Where does the mesh needs to be refined? In which direction(s)?

U MTF271, Assignment 2b, recirculating flow: PANS, DES, DDES and SAS

In this exercise you will use data from PANS and Zonal PANS SAS for for the flow over a hump. In Section T, a similar analysis is carried for channel flow. It may be interesting to do that assignment before this one.

The results of the hump simulations are presented in detail in [167, 178, 212] [12]

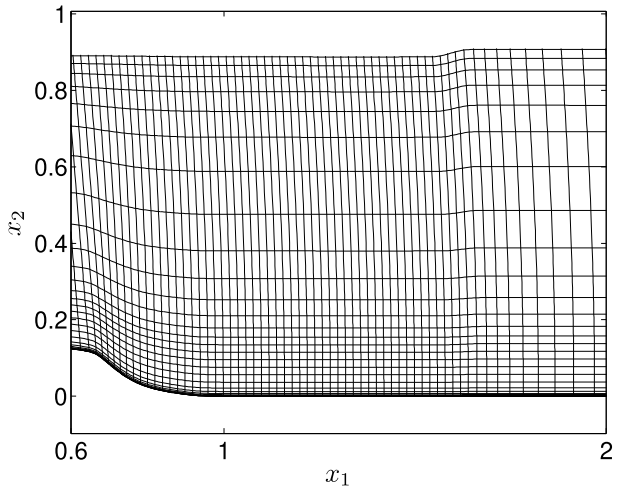

Figure U.1: Hump flow. Grid. Every 4^{th} grid line is shown.

The Reynolds number of the hump flow is $Re_c = 936\,000$, based on the hump length, c, and the inlet mean velocity at the centerline, $U_{in,c}$. The grid is shown in Fig. U.1. Experiments were conducted by Greenblatt [216,217]; they are also described in detail at http://cfdval2004.larc.nasa.gov/case3.html. The maximum height of the hump, h, and the channel height, H, are given by $H/c = 0.91$ and $h/c = 0.128$, respectively. The baseline mesh has $312 \times 120 \times 64$ cells with $Z_{max} = 0.2$.

There are side-wall effects (3D flow) near the side plates in the experiment. Hence, in order to compensate for the blockage effect of the side plates in the computation, the surface shape of the upper wall (above the hump) is modified and the upper wall is moved slightly indented, see Fig. U.1. The ratio of the local cross-sectional area of the side plates (facing the flow) to the cross sectional area of the tunnel enclosed by the side plates was computed. This ratio was used to scale the local height of the channel and thus modifying the contour shape of the upper wall.

Neumann conditions are used at the outflow section located at $x_1 = 4.2$. Slip conditions are used at the upper wall and symmetric boundary conditions are used on the spanwise boundaries. Inflow boundary (at $x_1 = 0.6$) conditions are taken from 2D RANS SST $k - \omega$ simulations carried out by the group of Prof. Strelets in St. Petersburg. The distributions of V_1 and V_2 at $x_1 = 0.6$ from the RANS simulation are used together with $V_3 = 0$ as mean inlet velocities to which the fluctuating velocity $\mathcal{V}_1'$, $\mathcal{V}_2'$ and $\mathcal{V}_2'$ are superimposed. The computed integral length scale for the synthetic inlet fluctuations is $\mathcal{L} \simeq 0.040$ and the integral time scale $\mathcal{T} \simeq 0.038$. It is noted that the

[12] these papers can be downloaded from http://www.tfd.chalmers.se/~lada/allpaper.html

prescribed inlet integral length scale is rather large [178] (approximately equal to the inflow boundary layer thickness). The reason is that synthetic fluctuations with a large integral length scale are efficient in generating resolved turbulent fluctuations [192].

At the course www page you find data files with time and spanwise averaged flow fields. You can use Matlab, Octave or Python. Both Octave and Python are open-source software. Octave is a Matlab clone. Many large Swedish industries prefer engineers to use Python instead of Matlab due to Matlab's high license fees.

Use one of the programs Python/Matlab/Octave to analyze the data. You find a Python/Matlab/Octave program at the www page which reads the data and computes the mean velocity. The data files are Matlab binary files.

Start by downloading `pl_vect_hump.py` or `pl_vect_hump.m`, `vectz_aiaa_paper.dat`, `xy_hump.dat` and `x065_off.dat` ...`x130_off.dat`. The py and m-files read the data files. The velocities, the square of the velocities and a number of other quantities are read. The resolved stresses are computed as in Eq. R.6. Furthermore, the py and m files plot contours of $\langle v_1' v_1' \rangle$, the grid, a vector plot as well as $\langle \bar{v}_1 \rangle$ and $\langle v_2' v_2' \rangle$ at $x = 0.65$. The experimental values are also included in the plots at $x = 0.65$.

U.1 Discretization schemes

You will analyze results from two publications [178, 212]. You can also find slides for the two papers at here (titles: *Embedded LES using PANS, Hawaii 2011* and *Embedded LES using PANS (CDS)*). The main difference is that in [212] pure central differencing was used whereas in [178] a blend of 95% central differencing and 5% bounded second-order upwind scheme (van Leer scheme [218]) was employed. In both works, f_k was set to 0.4 in the entire region, which means that the PANS model was operating in LES mode everywhere.

Consider the plot of $\langle \bar{v}_1 \rangle$ (Figure 1) and $\langle v_2' v_2' \rangle$ (Figure 2) at $x = 0.65$. The velocity profile shows that the boundary layer extends up to $x_2 \simeq 0.17$. However, the $\langle v_2' v_2' \rangle$ exhibits large values outside the boundary layer; the peak value at $x_2 \simeq 0.24$ is almost 5 times larger than in the boundary layer. These fluctuations are nonphysical and stem from the use of central differencing.

Now plot $\langle v_2' v_2' \rangle$ at locations further downstream (choose the positions where experimental data exist, i.e. $x = 0.65$, $x = 0.80$, $x = 0.90$, $x = 1.0$, $x = 1.1$, $x = 1.2$ and $x = 1.3$); There are nonphysical fluctuations at the inlet. Do they disappear further downstream? Compare with $\langle v_1' v_2' \rangle$ in Fig. 12 in [212]. The nonphysical fluctuations appear in the outer region where no physical resolved turbulence is present. Pure central differences sometimes give problems in flow regions like this. It is tempting to draw the conclusion that these nonphysical fluctuations are caused by the inlet synthetic fluctuations. However, when increasing the magnitude of the inlet synthetic fluctuations the magnitude of the nonphysical fluctuation diminishes, see Fig. 13a in [212].

Similar (much worse) problems with central differences were encountered in LES of the flow around an airfoil [115, 219]. Large nonphysical fluctuations were found in regions far from the airfoil; in these studies the problems were solved by using 100% van Leer scheme in the far-field regions.

In [178] the same simulations were carried out but now using a blend of central differencing (95%) and van Leer scheme (5%). Analyze these results by loading file `vectz_aiaa_journal.dat` in `pl_vect_hump.py` or `pl_vect_hump.m`.

Consider the resolved fluctuations: note how strongly they increase when going from the attached boundary layer ($x_1 = 0.65$) to the recirculating region, $0.8 \lesssim x \lesssim 1.2$. The peak of $\langle v_2' v_2' \rangle$, for example, increases from 0.0017 ($x_1 = 0.65$) to 0.046

($x_1 = 0.8$). This is the main reason why the nonphysical fluctuations at $x_1 = 0.65$ vanish quickly further downstream.

U.2 The modeled turbulent shear stress

As usual, when analyzing results of DES, PANS etc, we want to find out how much of the turbulence is modeled and how much is resolved. This gives an indication how well the turbulence has been resolved; if much turbulence is resolved it may be either good or bad. In recirculating regions it is usually good but it may be disastrous in boundary layer regions close to the wall if the mesh is not fine enough (this is the very reason why DDES was proposed, see Eq. 20.11). A suitable quantity to look at is the shear stress which is much more relevant than the turbulent kinetic energy; the shear stress is the largest diffusion term in the momentum equations, much larger than the normal stresses (at least in boundary layer flows).

Now plot the resolved (`uv_2d`) and the modelled shear stress (`uv_model_2d`) in the boundary layer region (e.g. $x = 0.65$) and the recirculating region (e.g. $x = 1$). Where is the modeled shear stress large and where is it small? Look at the other locations with experiments at some locations ($0.65 \ldots 1.30$).

When you compare turbulent quantities here and below, note that it is interesting to compare turbulent quantities **only inside the turbulent boundary layer**. The flow is laminar outside the turbulent boundary. To find the thickness of the turbulent, find where ν_t/ν falls below one.

U.3 The turbulent viscosity

It may be interesting to look at the turbulent viscosity. Usually it is plotted scaled with ν, i.e. ν_t/ν. Plot the viscosity, ν_t/ν (`(vis_2d-viscos)/viscos`), `vis_2d` is the total viscosity, $\nu_{tot} = \nu + \nu_t$) in the boundary layer and in the recirculating region. You find that the turbulent viscosity is large in the recirculation region, right? That might lead you to conclude that the turbulence is better resolved in the boundary layer (where ν_t is small) than in the recirculating region. But that would be an incorrect conclusion. When estimating the influence of the modeled turbulence, we should not look at the turbulent viscosity because it does not appear as a term in the momentum equations; we should look at the ratio of modeled stresses to the resolved one, and, as mentioned in Section U.2, preferably the ratio of the modeled to the resolved shear stress [128, 129]. It would actually be even more relevant to look at the divergence of these terms, i.e.

$$\frac{\partial}{\partial x_j}\left(\left\langle \nu_t \frac{\partial \bar{v}_i}{\partial x_j}\right\rangle\right) \quad \text{and} \quad -\frac{\partial \langle v_i' v_j' \rangle}{\partial x_j} \tag{U.1}$$

because these are the terms that indeed appear in the momentum equations. Plot and compare the two terms in Eq. U.1.

You have noted that the turbulent viscosities are rather large; in the recirculation region ν_t/ν is larger than 100. Go back and load the results of [212] (`vectz_aiaa_paper.dat`). Plot the turbulent viscosity for the two data sets. You find that the turbulent viscosity is much larger than for the data of [178]. It is true that the discretization schemes used in the two papers are slightly different, but that is not the reason. The difference stems from the fact that the turbulent Prandtl numbers in the k and ε equations were in the two papers set as

$$\sigma_k = 1.4, \quad \sigma_\varepsilon = 1.4 \quad [212] \tag{U.2}$$

$$\sigma_k = 1.4 f_k^2, \quad \sigma_\varepsilon = 1.4 f_k^2 \quad [178] \tag{U.3}$$

Equation U.3 corresponds to the original PANS model. Recall that the turbulent diffusion in, for example, the k equation reads

$$\frac{\partial}{\partial x_j}\left(\frac{\nu_t}{\sigma_k}\frac{\partial k}{\partial x_j}\right) \tag{U.4}$$

Since $f_k = 0.4$, it means that the turbulent diffusion in the k and ε equations are $1/0.4^2 \simeq 6$ times larger in [178] than in [212]. The consequence is that peaks in k and ε (and also ν_t) are reduced in the former case compared to the latter (this is the physical role played by diffusion: it transports k from regions of high k to regions of low k). This explains why the peaks of k are much larger in [212] compared to in [178].

Hence, in the original PANS model (Eq U.3), the RANS turbulent viscosity appears in the turbulent diffusion of k (and ε), because the turbulent diffusion term reads (recall that $f_k = k/k_{total} = k/k_{RANS}$ where k_{RANS} denotes the turbulent kinetic energy in a RANS simulation)

$$\begin{aligned} \frac{\partial}{\partial x_j}\left(\frac{\nu_t}{f_k^2\sigma_k}\frac{\partial k}{\partial x_j}\right) &= \frac{\partial}{\partial x_j}\left(\frac{c_\mu k^2}{\varepsilon f_k^2\sigma_k}\frac{\partial k}{\partial x_j}\right) \\ = \frac{\partial}{\partial x_j}\left(\frac{c_\mu k_{RANS}^2}{\varepsilon\sigma_k}\frac{\partial k}{\partial x_j}\right) &= \frac{\partial}{\partial x_j}\left(\frac{\nu_{t,RANS}}{\sigma_k}\frac{\partial k}{\partial x_j}\right) \end{aligned} \tag{U.5}$$

cf. Eqs. 18 and 19 in [145]. Thus the *total* (i.e. RANS) viscosity is responsible for the transport of the *modeled* turbulent kinetic energy.

U.4 Location of interface

The results analyzed above were from LES simulations [178, 212] (i.e. the PANS model was used in LES mode). Now we will analyze results from a modified IDDES. In [220] ψ – which is the ratio of the RANS to the LES length scale – is computed from Eq. 23.51. Note that for the SST-DES model we used the notation F_{DES} (Eq. 20.8) instead of ψ. We will use data obtained from [220] but on a finer mesh, $(ni, nj, nk) = (723 \times 128 \times 32)$ (this mesh has been refined in the outlet region compared to [220]).

Run the file `pl_vect_hump_IDDES.py` (in this part of the Assignment you must use Python) which loads the files

- `x2d_hump_IDDES.dat`
- `y2d_hump_IDDES.dat`
- `z_hump_IDD.dat`
- `itstep.npy`
- `p_averaged.npy`
- `u_averaged.npy`
- `v_averaged.npy`
- `w_averaged.npy`
- `k_averaged.npy`

- `vis_averaged.npy`
- `uu_stress.npy`
- `vv_stress.npy`
- `ww_stress.npy`
- `uv_stress.npy`
- `fk_averaged.npy`
- `eps_averaged.npy`
- `gen_averaged.npy`
- `x065_off.dat`.

Start by finding where the IDDES model switches from RANS to LES, i.e where ψ exceeds one. Plot location of the switch versus x.

The IDDES model is presented in Section 23.4.2. It has many blending functions and it is not straight-forward to understand. The main blending function is $\tilde{f}_d$, see Eq. 23.54. The author's experience is that the impact of the f_e function is negligible (it can be set to $f_e = 0$); this simplifies the IDDES model a lot since $\tilde{f}_d$ then only includes f_{dt} and f_B. The latter function is used to create a DES length scale (see Eq. 8 in [165] where it is denoted l_{WMLES})

$$L_{DES} = f_B L_{RANS} + (1 - f_B) L_{LES}$$

(I've set $f_e = 0$). Next, a DDES shielding function is introduced

$$L_{DDES} = L_{RANS} - f_d \max(0, L_{RANS} - L_{LES}) \tag{U.6}$$

where $f_d = 0$ gives RANS. These two expressions are then combined so that (cf. Eq. 23.53)

$$L_{IDDES} = \tilde{f}_d L_{RANS} + \left(1 - \tilde{f}_d\right) L_{LES}$$

where $\tilde{f}_d = 1$ gives RANS (since $1 - f_d$ is used in Eq. 23.54).

Plot $\tilde{f}_d$ along y at the usual locations $x = 0.65 \ldots 1.3$. The $\tilde{f}_d$ should fall below one at the same location as ψ exceeds one.

U.5 Location of interface in DES and DDES

Let's compare IDDES with DES and DDES. Recall that $\Delta z = 0.2/32$. In SA-DES, the interface is defined as the location where the wall distance is equal to $C_{DES}\Delta$ where $\Delta = \max\{\Delta_x, \Delta_y, \Delta_z\}$, see Eq. 20.3. How does this compare with switching locating defined by IDDES?

In SST-DES, the location of the interface is computed using k and ω. Compute ω from $\varepsilon/(\beta^* k)$ and compute the location using Eq. 20.8. How does the location compare with IDDES and SA-DES?

In DDES, the boundary layer is shielded with a blending function. As shown above IDDES is actually a combination of wall-modelled LES (i.e.. hybrid LES-RANS) and DDES (Eq. U.6) where the shielding function is given by Eq. 23.56.

Use the shielding function to find where the DDES switch takes place. When computing the shielding function you need $\bar{s}$ in Eq. 23.58. This quantity is loaded in `pl_vect_hump_IDDES.py`.

Does DDES work, i.e. does it make the model to be in RANS mode in the entire boundary layer? What about the separation region?

U.6 The SAS model

Learn about the SAS model in Section 22. The SAS model is based on the von Kármán lengthscale, Download the Python script `pl_sas_IDDES.py` which reads data files with instantaneous results. The files are

- `u3d_saved_0.npy`, `v3d_saved_0.npy`, `w3d_saved_0.npy`
- `u3d_saved_100.npy`, `v3d_saved_100.npy`, `w3d_saved_100.npy`
- `u3d_saved_200.npy`, `v3d_saved_200.npy`, `w3d_saved_200.npy`

⋮

-
- `u3d_saved_2200.npy`, `v3d_saved_2200.npy`, `w3d_saved_2200.npy`

which include 23 independent instantaneous velocity fields. Only the first three fields (`nfiles=2`) are loaded in `pl_sas_IDDES.py`. The more instantaneous velocity fields that are used, the better statistics and the smoother the plotted lines. When everything works, I recommend that you use more files (maybe all 23; set `nfiles=22`).

In the beginning of `pl_sas_IDDES.py`, all first and second velocity derivatives are computed. Compute the instantaneous von Kármán lengthscale $L_{vK,3D}$ using Eq. 22.3. Spanwise average and plot it at a couple of positions ($0.65 \ldots 1.30$).

Recall that the idea of the SAS model is that the von Kármán lengthscale is smaller for an unsteady flow field than for a steady one. Compute the von Kármán lengthscale using the time-averaged velocity field (use the file `vectz_fine.dat`); note that $\langle \bar{v}_3 \rangle$ is zero since the time-averaged flow field is two dimensional). Compare it with $L_{vK,3D}$. Compare also with $C_{DES}\Delta$. Try also the alternative formulation of U'' in Eq. T.3.

The von Kármán lengthscale using the time-averaged velocity field is a RANS lengthscale. Therefor it is interesting to compare it with the RANS lengthscale $(k_{res} + k)^{3/2}/\varepsilon + k$ where k_{res} is the resolved turbulent kinetic energy and ε is the modelled dissipation (from the PANS $k - \varepsilon$ model). Note that what we denote as a 1D von Kármán lengtscale in Section 22 here corresponds to

$$L_{vK,steady} = \kappa \frac{(2\langle \bar{s}_{ij} \rangle \langle \bar{s}_{ij} \rangle)^{1/2}}{U''_{mean}} \tag{U.7}$$

where

$$U''_{mean} = \left(\frac{\partial^2 \langle \bar{v}_i \rangle}{\partial x_j \partial x_j} \frac{\partial^2 \langle \bar{v}_i \rangle}{\partial x_k \partial x_k} \right)^{0.5} \tag{U.8}$$

Note than in a CFD code, Eq. 22.3 and 22.2 are always used but when the flow is steady it is equal to Eqs. U.7 and U.8.

U.7 Different ways to evaluate resolution

In [128, 129] different ways to estimate the resolution of an LES were investigated. It is also discussed in Section 18.26. There are many ways to estimate resolution:

1. Ratio between viscous and modelled turbulent viscosity (not recommended in [128, 129]).
   Plot $\langle \nu_t \rangle / \nu$ at a couple of positions $0.65 \ldots 1.30$. This quantity does not say much about how good the LES resolution is. It tells us how close the LES is to a DNS.

2. Ratio between modeled and total shear stress (recommended in [128, 129]).
   Plot $|\langle \tau_{12} \rangle / (\langle \tau_{12} \rangle + \langle \overline{v}_1' \overline{v}_2' \rangle)|$ at a couple of positions $0.65 \ldots 1.30$. Compute the modeled shear stress from the Boussinesq approximation.

3. Ratio between modeled and total turbulent kinetic energy (recommended in [205]).
   Plot $\langle k_{model} \rangle / (\langle k_{model} \rangle + k_{res})$ ratio at a couple of positions $0.65 \ldots 1.30$.

4. Anisotropic error (optional).
   Plot the anisotropic error at a couple of positions $0.65 \ldots 1.30, 2.00, 3.00$. This quantity should be as constant as possible. Some further instructions are given in Section U.7.1.

5. Two-point correlations.
   Plot the two-point correlations (recommended in [128, 129]). You find info on how to proceed in Section U.7.2

6. Ratio of boundary-layer thickness, δ to Δx and Δz. This is a measure of the resolution in the log-law region. It is commended that $\delta / \Delta x < 10$ and $\delta / \Delta z < 20$, see Section 18.26.
   Plot $\delta / \Delta x$ and $\delta / \Delta z$ vs. x. One question arises: how the estimate the boundary-layer thickness. One useful definition is to define δ as the location where ν_t / ν falls below one.

In [205] he suggest that if $k_{model} / (k_{model} + k_{res}) < 0.2$ (ratio of modelled to total turbulence), then the turbulence is well resolved by the LES. Assume we make the same definition for the shear stress, i.e. $|\langle \tau_{12} \rangle / (\langle \tau_{12} \rangle + \langle \overline{v}_1' \overline{v}_2' \rangle)| < 0.2$.

Do these methods give similar results? Do you trust anyone more than the other? (hint: look at [129].

U.7.1 Anisotropic errors (optional)

You plotted this error estimate also for the channel flow, see Section T.8. You can use the first and second-order derivatives that you computed in Section U.6. The Δx and Δy needed in Eq. T.6 can be computed as

```
dx=diff(xp2d);
dx=repmat(dx,[1 1 nk-1]);
dx(ni,:)=dx(ni-1,:);

dy=diff(yp2d,1,2);
dy(:,nj)=dy(:,nj-1);
dy=repmat(dy,[1 1 nk-1]);
```

(Δz is constant). Only two sets of velocity fields are loaded, `u1_pans_iddes.mat`, ...`w2_pans_iddes.mat`. When everything works, use more velocity fields (possibly all eight) in order to get better statistics.

U.7.2 Two-point correlations

In Item 5 in Section U.7 you should compute two-point correlation. To do that, you will use data from another simulation of the hump flow presented in [171, 174]. Download the

- Pythons file `pl_twocorr_computed_fk.py` or
- the Matlab/Octave file `pl_twocorr_computed_fk.m`.

They load a new grid with 305×109 grid points. The 3D grid has 32 cells in the spanwise direction ($z_{max} = 0.2$). `pl_twocorr_computed_fk` will also load time histories of the spanwise velocity $\bar{v}_3$ at $x = 0.65, 0.8, 1.1$ and $x = 1.3$ (files `w_time_z65`, `w_time_z80`, `w_time_z110` and `w_time_z130`). These files include time histories at cell center $j = 10, 30, 50, 70, 90$ in the wall-normal direction at 16 positions in the x_3 direction. The time series are re-arranged into 3D arrays `w_y_z_t_65(j,k,t)`, `w_y_z_t_80(j,k,t)`, `w_y_z_t_110(j,k,t)` and `w_y_z_t_130(j,k,t)` where index j, k, t denote wall-normal direction, spanwise direction and time, respectively.

Now, use the time series to compute two-point correlations using the formulas given in Section 10.1. Equation 10.3 gives the two-point correlation of v_1' at two points separated in the x_1 direction. In your case, you will compute the two-point correlation of v_3' at two points separated in the x_3 direction, i.e. $B_{33}^{norm}(x_3^A, \hat{x}_3)$. Plot it at all x_1 positions ($x_1 = 0.65, 0.8, 1.1$ and 1.3) and at a couple of x_2 locations. Then compute the integral lengthscale, see Eq. 10.6. A good approximation of the integral lengthscale is usually the point in the two-point correlation where $B_{33}^{norm} \simeq 0.2$ (i.e. $\mathcal{L} \simeq \hat{x}_3$ where $B_{33}^{norm} \simeq 0.2$). Check if it is good approximation.

In [128, 129] it is recommended that in a good LES (or in the LES region of a DES/hybrid LES-RANS), the integral lengthscale should cover 8-16 cells, depending on how accurate the user wants her/his LES/DES to be.

Here are some hints on how to compute the two-point correlation. Start by computing it for one y location (=0 in Python and =2 in Matlab/Octave) and one z separation, $\hat{z} = 2\Delta z$). In Python

```
B33=0
k=2
for n in range(1,N): # time average
  B33=B33+w[0,0,n]*w[0,0+k,n]/N
```

and in Matlab/Octave

```
B33=0;
k=2;
for n=1:N % time average
  B33=B33+w(2,1,n)*w(2,1+k,n)/N;
end
```

where $k = 2$ (because the separation is $\hat{z} = 2\Delta z$) and N is the number of time steps. This gives $B_{33}(2\Delta z)$. Next, do it for $\hat{z} = 3\Delta z$, i.e. (in Python)

```
B33=0
k=3
for n in range(1,N): # time average
  B33=B33+w[0,0,n]*w[0,0+k,n]/N
```

This gives $B_{33}(3\Delta z)$. Now, do it for all $k = 0, 1, \ldots 15$. And then normalize by w_{rms}^2 (or, easier, simply by `B33[0]` (Python) or `B33(1)` (Matlab/Octave)). Check that it is correct by plotting $B_{33}(\hat{z})$ versus $\hat{z}$. It should look someting like the two-point correlation in Fig. 10.1. Finally, compute the two-point correlation for all five y locations and four x locations and compute the integral length scale.