Chalmers University of Technology
MTF 270 - Turbulence Modeling

ASSIGNMENT 2

Group 14 - Project members:
Vishal Subramaniasivam

Date : April 4, 2022

Contents

1 Assignment 2a : Channel flow - DES, DDES and SAS

For this assignment, a fully developed flow through a channel is studied. The DES data was provided for 5 nodes located in the domain - Node 1 ($y/\delta = 0.0028$), Node 2 ($y/\delta = 0.0203$), Node 3 ($y/\delta = 0.0364$), Node 4 ($y/\delta = 0.0645$), and Node 5 ($y/\delta = 0.2$). The Reynolds number of the flow based on half channel width was $R_e = 8000$. The matching line determined by DES switch occurs at $y/\delta = 0.0175$.

According to the given data, we can figure out that the node close to the wall i.e. Node 1 lies in the URANS region whereas Node 2 is close to the interface between URANS and LES. All other points (i.e. Nodes 3, 4 and 5) lie in the LES region.

1.1 Time history

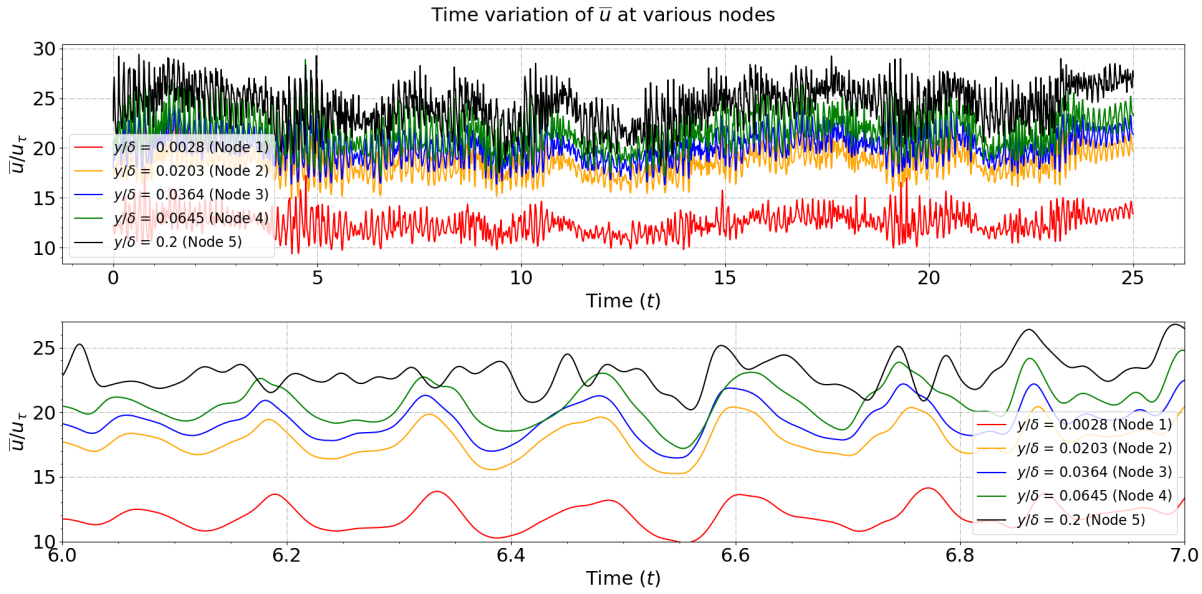


Figure 1: Time History of $\overline{u}$ at various nodes

Figure 1 and figure 2 shows the time variation of the velocities at the five node points. The figure at the bottom is the zoomed to clearly observe the variation of velocities between time 6 sec and 7 sec. As node 1 lies in URANS region, the turbulent viscosity is much larger here than for other node points. This means that the velocities are much more dampened out in URANS region as the turbulent viscosity appears in the diffusion term of the momentum equations.

A similar trend is observed in figure 1, $\overline{u}$ is very less for node 1, whereas it keeps increasing as we move away from the URANS region into the LES region. For velocity in spanwise direction, we expect a similar behaviour and can be seen in figure 2. Although there is not much of a difference in magnitude of $\overline{w}$, nevertheless we could see that it is slightly smaller at node 1.

Figure 3 shows the auto-corelation calculated at the five node points for the fluctuations $\overline{u'}$. Auto-corelation is the corelation of a variable with a separation in time and can be used to see how the events within the turbulent flow are related to each other. The auto-corelation is further used to calculate the integral timescale of the eddies (assuming that the mean flow is steady). For any turbulent quantity, if the samples are separated by one timescale, they are independent.

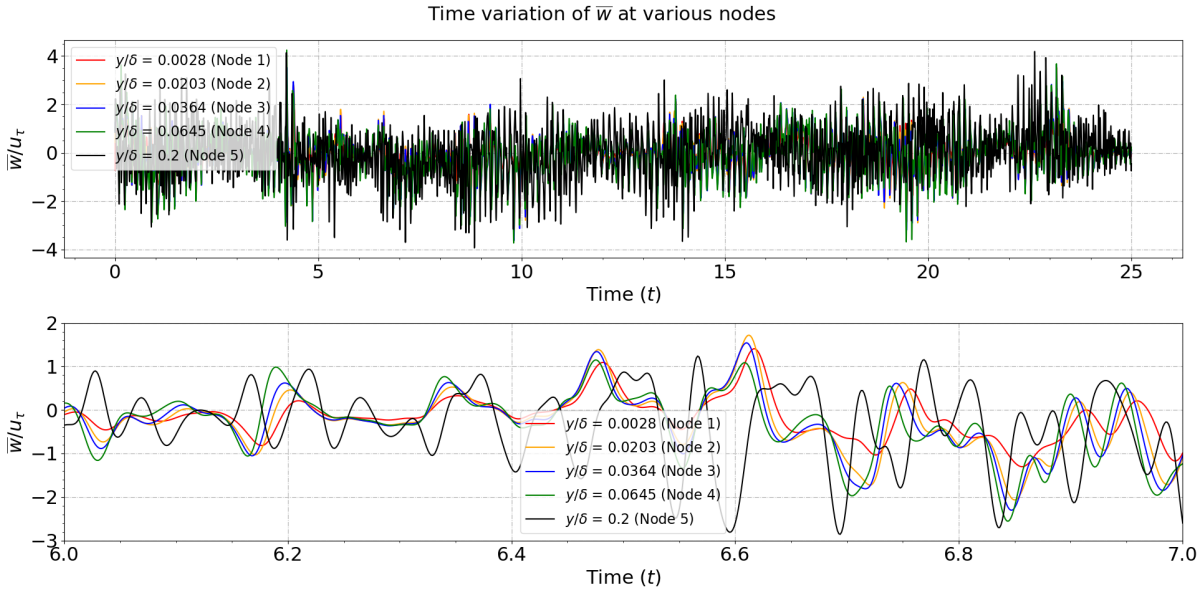


Figure 2: Time History of $\overline{w}$ at various nodes

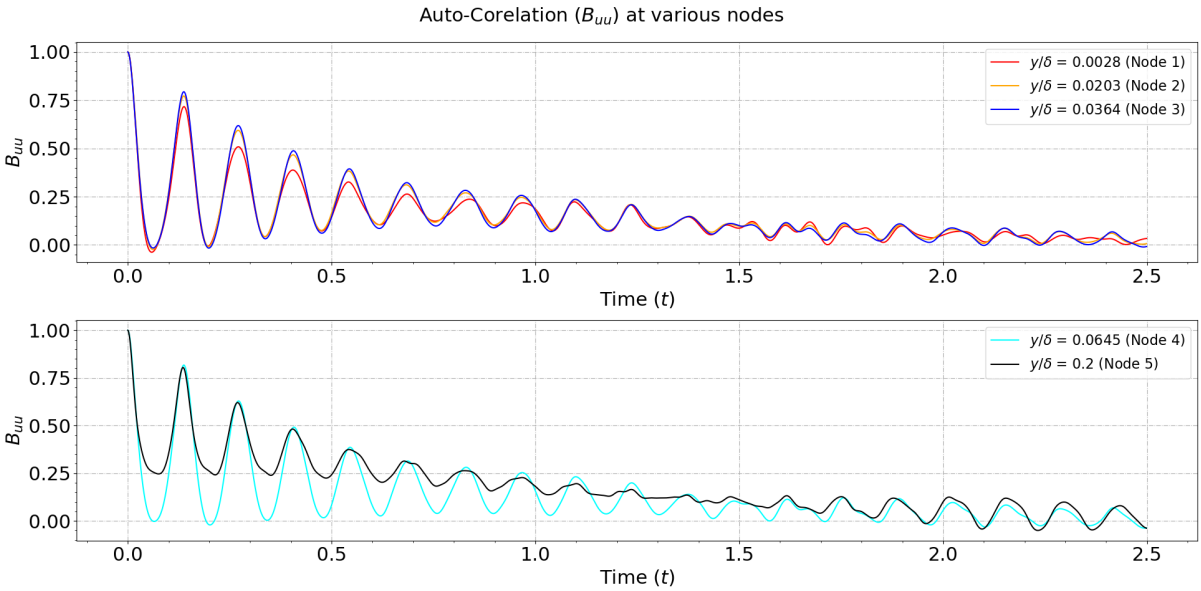


Figure 3: Auto-Corelation at various nodes

Looking at the graph of auto-corelation in figure 3, we see that the auto-corelation is 1 at time t=0, which is quite obvious as the turbulent quantity at a particular instance will depend on itself. But as we progress in time, the auto-corelation should converge to zero ideally. Here, due to numerical noise, it does not converges to zero.

Integral timescale is basically the area under the curve for the auto-corelation .vs. time graph. To calculate the integral time scale, we integrate till the point where the auto-corelation is zero or it starts increasing (see figure 4). The auto-corelation function at node 1 reaches zero faster as compared to other nodes, which means that there is a faster moving or small eddy at that point due to which the

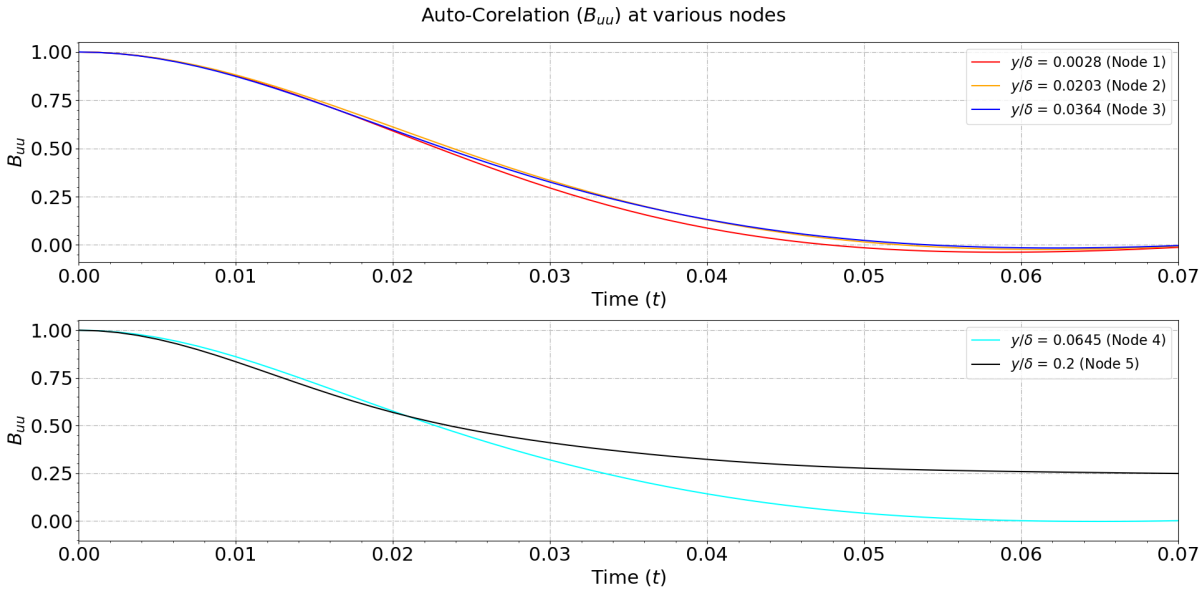


Figure 4: Auto-Corelation at various nodes (Zoomed-in)

turbulent quantity will change more rapidly at that point. At nodes 2-4, the auto-corelation is nearly the same hence the eddies at that points will have a similar time scale. For node 5, the auto-corelation function reaches zero at later stage and hence a larger eddy exists at this point which will have a large timescale as compared to other points and the turbulent quantities at this point will change slowly.

Integral timescale for various node points :

- For Node 1 = 0.023485 seconds
- For Node 2 = 0.024714 seconds
- For Node 3 = 0.024495 seconds
- For Node 4 = 0.024400 seconds
- For Node 5 = 0.034502 seconds

The timestep used for the simulation is 0.00125 seconds, which is smaller than the timescales of the eddies at the particular points as shown above. This is good as we evaluate the turbulent quantities which are statistically dependent in the subsequent timestep of the simulation. Also, we could use the knowledge of timescales to manipulate how we can store the data. Instead of storing it every timestep, we could store the flow data every subsequent integral timescale, this could save a lot of space in simulations.

1.2 Mean velocity profile

Time-averagiing is performed on homogenous x_1 and x_3 directions. The resultant $< \overline{v_1} >$ is compared with the DNS data available. From figure 5, we observe that the mean velocity is slightly over-predicted in DES model as compared to the DNS mean velocity. The matching line at $y/\delta = 0.0175$ can also be observed in the figure.

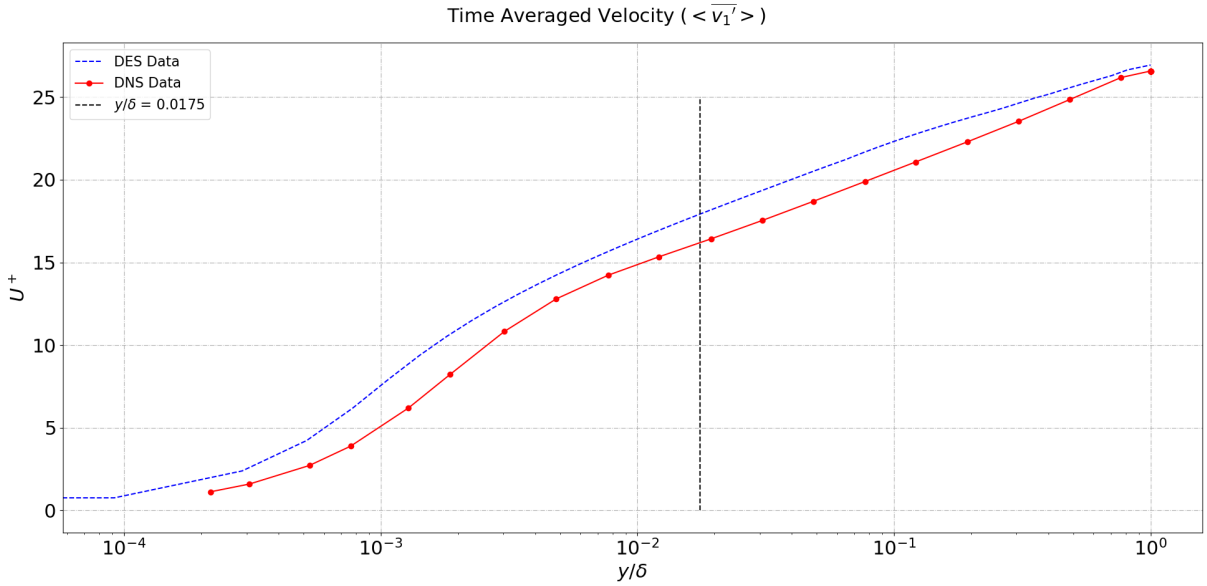


Figure 5: Time averaged velocity $< \overline{v_1} >$

1.3 Modeled and Resolved turbulent shear stresses (S3 and S5)

Below $y/\delta = 0.0175$, URANS is used to evaluate the turbulent quantities. URANS (Unsteady RANS) is an eddy viscosity based model, where the turbulent viscosity is calculated based on the Boussinesq assumption to calculate various turbulent quantities. The turbulent quantities are based on an empirical model, hence the stresses in this region can be called as modeled shear stress - they should be negligible in LES region. On the other hand, above $y/\delta = 0.0175$, LES is used to evaluate the turbulent quantities in the mean flow. In LES, the discretized momentum equation is solved and the turbulent quantities are evaluated without using any model. The stresses calculated without any assumptions can be called as resolved stresses.

The total shear stress for a DES model is calculated by adding the modeled and resolved shear stresses. In URANS region, the resolved stresses should be small and in LES region, the modeled stresses should be negligible. The DES model switches between the LES and URANS formulation based on the turbulent length scale and the cell size (Δ).

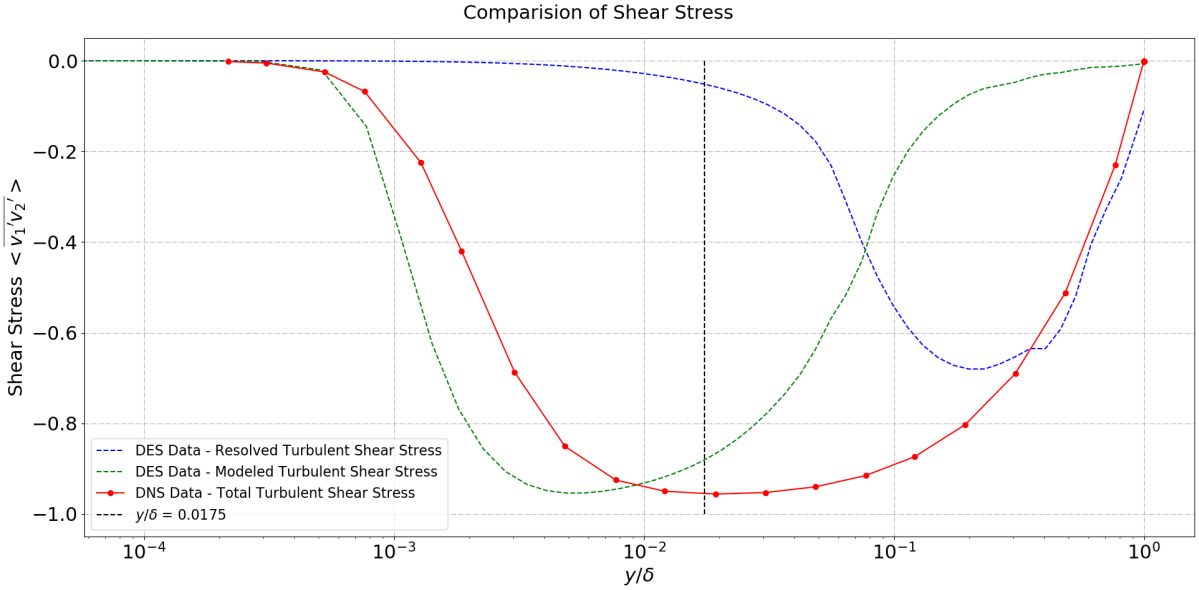


Figure 6: Comparision of modeled and resolved turbulent shear stress

Figure 6 shows the modeled and the resolved turbulent shear stresses calculated based on the given DES data.

- It can be clearly seen that the resolved shear stress is negligible in the most of the URANS region (below $y/\delta = 0.0175$) and it starts increasing near the interface and peaks at nearly about $y/\delta = 0.1$. Most of the shear stress in LES region is resolved by solving the discretized momentum equations.
- The modeled shear stress is larger in the URANS region and starts decreasing after the interface and is negligible where modeled shear stress is high. This is good as the near wall turbulence is modeled using eddy viscosity model, which was the aim of DES model - to not use LES in near wall region and reduce mesh requirements and computation time.
- The reason we model the near wall turbulence is to relax the meshing requirements of LES models when the eddies are very small, especially near the wall. It saves the computation time and we obtain an acceptable accuracy in terms of result.

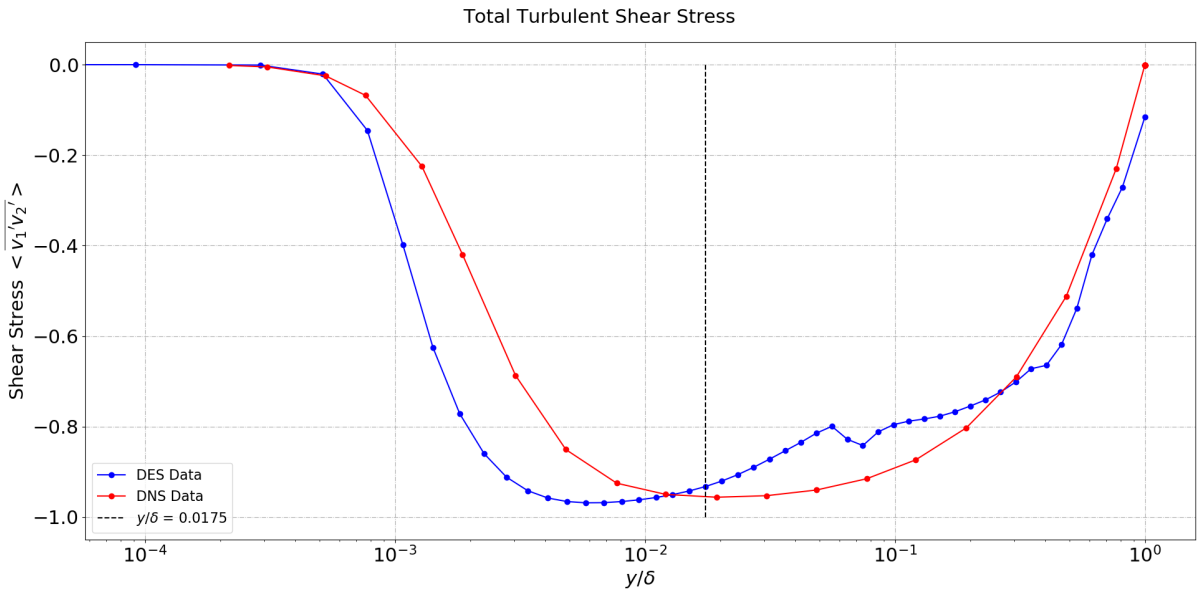


Figure 7: Total turbulent shear stress

Figure 7 shows the total turbulent shear stress evaluated using the DES data and is compared with the data of a DNS simulation. It is seen that the shear stress is over-predicted in the RANS region, whereas it matches fairly good with the data in LES region. This is because the turbulent viscosity is higher in RANS region as it is an eddy-viscosity model. ($\overline{v_i' v_j'} = -\nu_t \overline{s_{ij}}$)

1.4 Turbulent kinetic energy

Figure 8 shows the turbulent kinetic energy calculated using the given DES data and is compared with the DNS data.

- In RANS region (below $y/\delta = 0.0175$), the modeled kinetic energy is more than the resolved kinetic energy. It agrees with the fact that most of the turbulence is modeled in DES below the matching line. The modeled kinetic energy decreases above $y/\delta = 0.0175$ in LES region and is negligible in outer part of the boundary layer.
- In LES region (above $y/\delta = 0.0175$), the resolved kinetic energy is higher than the modeled kinetic energy. Also, it is smaller in the RANS region.
- Near the interface the contribution of both modeled and resolved kinetic energy is significant.
- The total kinetic energy is also compared with the DNS data. In RANS region the kinetic energy is smaller than the DNS kinetic energy. But in most of the LES region the DNS and DES kinetic energies show a fairly good match.

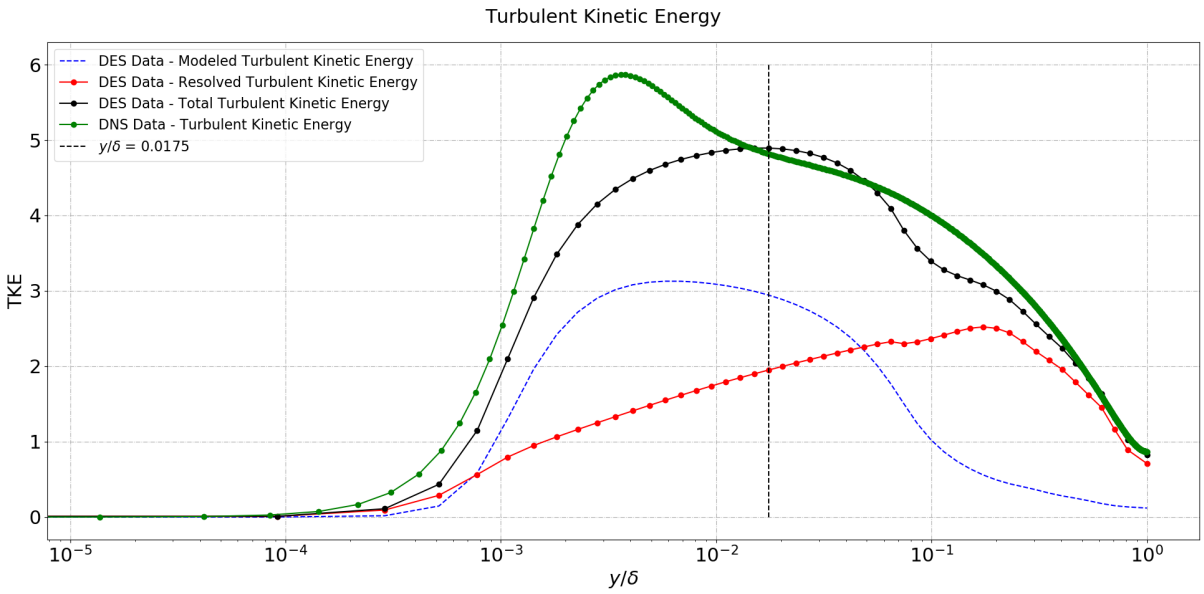


Figure 8: Turbulent Kinetic Energy

1.5 Location of interfaces in DES and DDES

In this simulation the interface ($y/\delta = 0.0175$) is calculated based on the DES switch. The interface is calculated based on various criterions for different turbulence models.

- **SA-DES model :** interface is defined as the location where wall distance is equal to $C_{DES}\Delta$, where $\Delta = \max(\Delta_x,\Delta_y,\Delta_z)$
- **SST-DES model :** interface is defined as the location where RANS length scale ($\propto k^{0.5}/\omega$) is equal to $C_{DES}\Delta$.
- **DDES model :** interface defined in the same way as SST-DES model, but it is located in outer part of boundary layer and achieved using a shielding function (F_S) in the formulation.

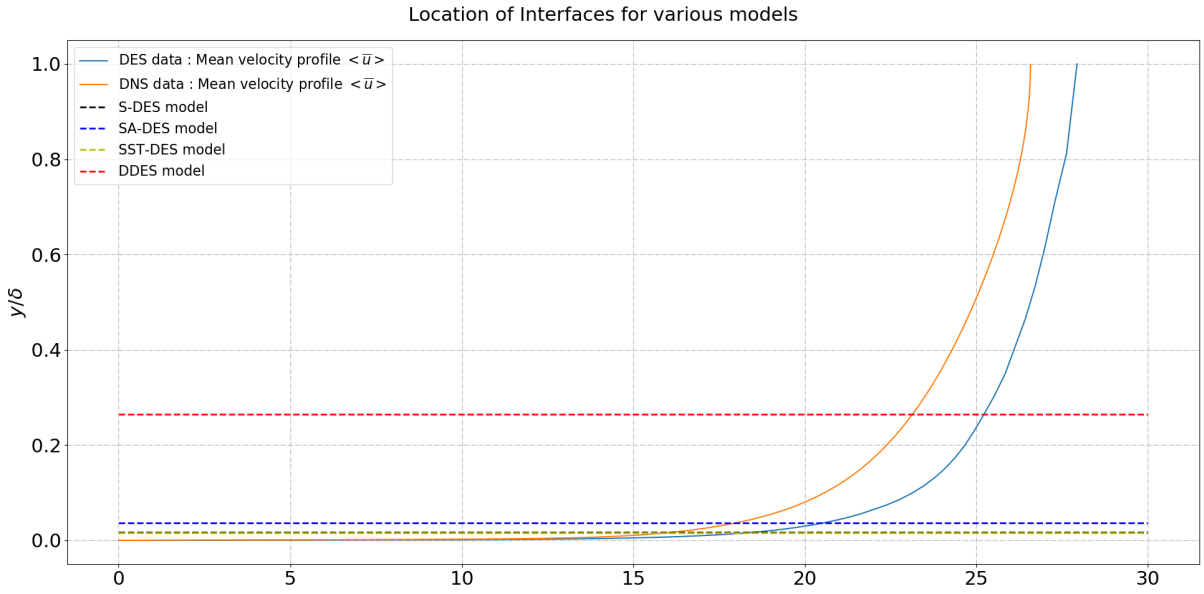


Figure 9: Location of Interface

In figure 9, the interfaces calculated for various turbulence models are shown. SST-DES and the DES model used in this simulation have nearly the same location of interface. The interface predicted by SA-DES model is slightly higher because it considers the wall distance as the parameter for switch rather than any length scale. For DDES, the interface is located the farthest from the wall in the outer region of the boundary layer. In DDES model, we want most of the boundary layer to be modeled rather than resolved by LES as sometimes the mesh resolution can be poor in this region (transition region) and we then risk solving the LES equations on a poor mesh, which can give wrong results. So we achieve this by using the DDES formulation and as seen in the figure most of the boundary layer uses RANS model to evaluate turbulence.

Switch line locations:

- S-DES model : $y/\delta = 0.0175$
- SA-DES model : $y/\delta = 0.0363$
- SST-DES model : $y/\delta = 0.0150$
- DDES model : $y/\delta = 0.2648$

1.6 SAS Turbulent length scales

The von-karman length scales are calculated in this section. The 1D von-karman length scale is given as -

$$L_{vK,1D} = \kappa \left| \frac{\partial \langle \overline{v_1} \rangle / \partial x_2}{\partial^2 \langle \overline{v_1} \rangle / \partial x_2^2} \right| \tag{1}$$

In figure 10, we evaluated the 1D Von-karman length scale using the provided DES and DNS data. Near the wall, we can see a good agreement between DES and DNS length scales. After $y/\delta = 0.5$, the DES length scale is very large which might be due to a relatively coarse mesh (we can see very less data points in that region as compared to near the wall). In section 1.7, it is seen that the mesh resolution in y-direction can be improved.

Also near the wall, we observe that the 1D Von-karman length scale does not go to zero as it approaches the wall, i.e. $\mathcal{O}(x_2^0)$. This can be proved using Taylor's expansion as given in equation 11.155 in Ebook -

$$\begin{aligned} \partial \overline{v_1}/\partial x_2 &= \overline{a_1} + ... = \mathcal{O}(x_2^0) \\ \partial^2 \overline{v_1}/\partial x_2^2 &= 1 + \overline{a_2} + ... = \mathcal{O}(x_2^0) \\ \text{Hence, } \frac{\partial \overline{v_1}/\partial x_2}{\partial^2 \overline{v_1}/\partial x_2^2} &= \mathcal{O}(x_2^0) \end{aligned} \tag{2}$$

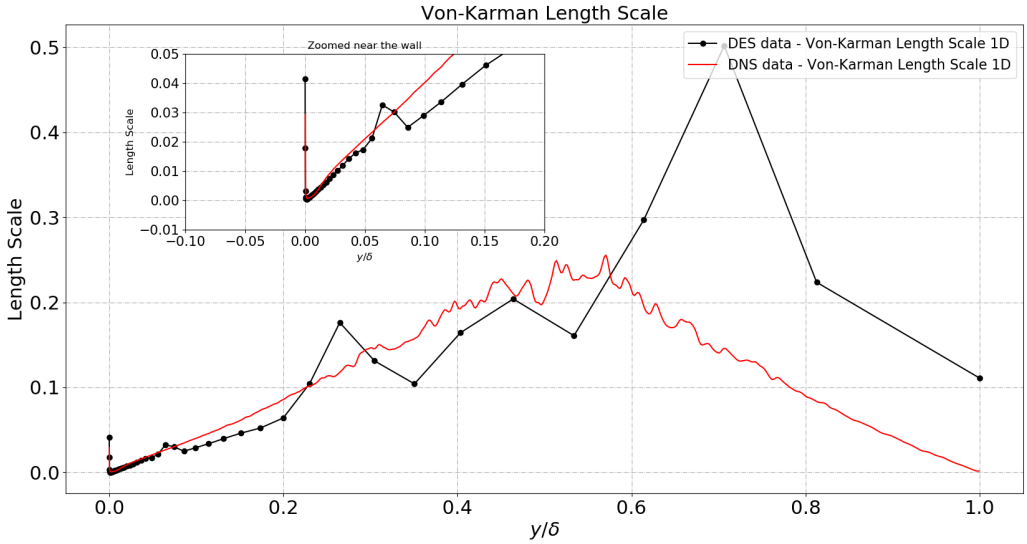


Figure 10: 1D Von-karman length scales

The 3D von-karman length scale is computed as -

$$\begin{aligned} L_{vK,3D,inst} &= \kappa \left| \frac{S}{U"} \right| \\ S &= 2(\overline{S_{ij} S_{ij}})^{0.5} \\ U" &= \left(\frac{\partial^2 \overline{v_i}}{\partial x_j \partial x_j} \frac{\partial^2 \overline{v_i}}{\partial x_j \partial x_j} \right)^{0.5} \end{aligned} \tag{3}$$

Other way to calculate the 3D von-karman length scale is by using equation 3 but a different method to compute the second derivative (U") -

$$U" = \left(\frac{\partial^2 \overline{v_i}}{\partial x_j \partial x_k} \frac{\partial^2 \overline{v_i}}{\partial x_j \partial x_k} \right)^{0.5} \tag{4}$$

Figure 11 shows the 3D von-karman length scales evaluated using both the methods described in equation 3 and equation 4. The length scales in calculated based on averaged velocities($L_{vK,1D}$) are larger than that calculated using fluctuating velocities ($L_{vK,3D,inst}$). Also the 3D length scale calculated using method 1 is slighty higher than method 2 (where we also consider cross-diffusion terms while calculating U").

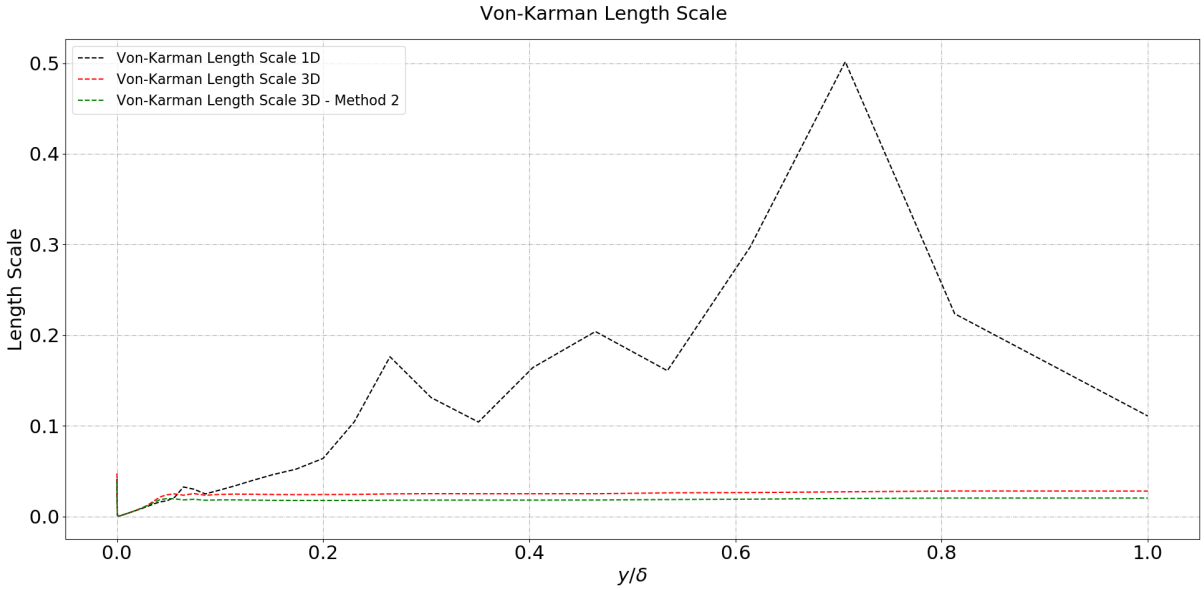


Figure 11: 3D Von-karman length scales

1.7 Estimation of resolution

The anisotropic error in n-direction is evaluated as follows -

$$\mathcal{A}(n)v = \Delta_x \Delta_y \Delta_z (\langle \overline{v}_i^{*(n)} \overline{v}_i^{*(n)} \rangle)^{0.5}$$
$$\text{where, } \overline{v}_i^{*(n)} = \frac{-\Delta_n^2}{4} \frac{\partial \overline{v_i}}{\partial x_n \partial x_n} \tag{5}$$

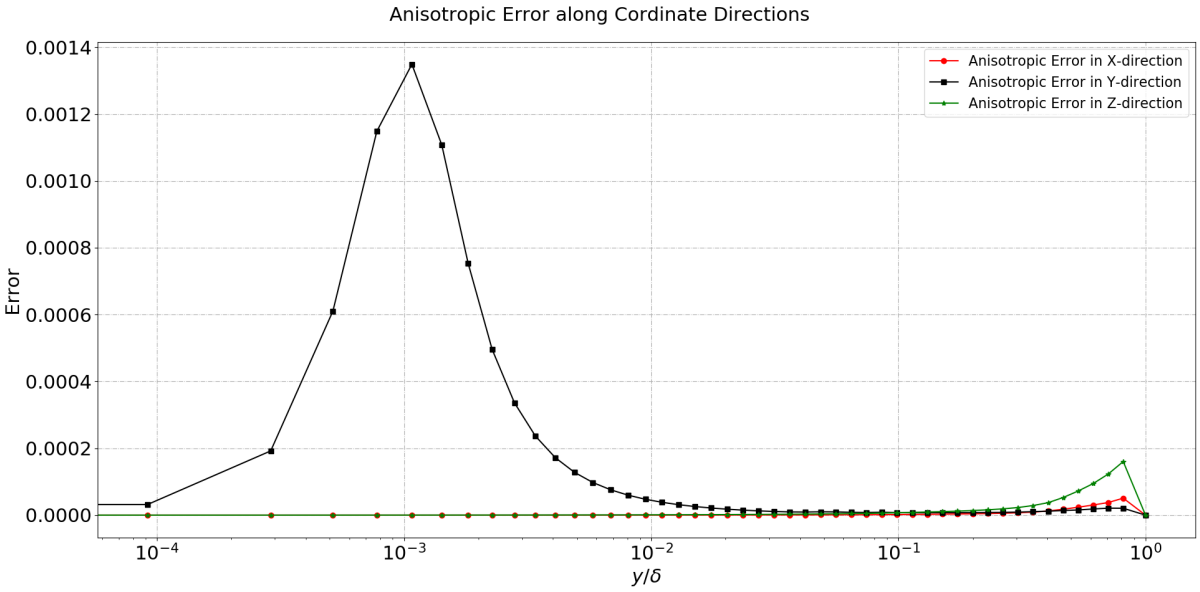


Figure 12: Anisotropic error

The quantity $\mathcal{A}(n)$ is used to define where the grid needs to be refined. Using equation 5 we have calculated the anisotropic error for all the three directions and it can be seen in figure 12.

- The error in X and Z direction is constant throughout except a small bump towards the end which can be neglected if it is less the the threshold anisotropic error ($\mathcal{A}_{thresh}$). Hence, the mesh in X and Z direction need not be refined further.
- The error in Y direction is significantly higher than in other two directions, hence the mesh along Y direction can be refined to obtain a result which closely matched the DNS data.

2 Assignment 2b : Recirculating flow - PANS, DES, DDES and SAS

For this assignment, we studied the flow over a hump. The data from a PANS and Zonal PANS SAS is used for analysis and is compared with the experimental data locations shown in figure 13. The results obtained from the two publications were analyzed for this study -

- **Reference [169]** - L. Davidson and S.-H. Peng. *'Embedded large-eddy simulation using the partially averaged Navier-Stokes model.'* AIAA Journal, 51(5):1066–1079, 2013.
- **Reference [203]** - L. Davidson and S.-H. Peng. *'Embedded LES with PANS.'* In 6th AIAA Theoretical Fluid Mechanics Conference, AIAA paper 2011-3108, 27-30 June, Honolulu, Hawaii, 2011.

The main difference is that in [203] pure central differencing was used whereas in [169] a blend of 95% central differencing and 5% bounded second-order upwind scheme (van Leer scheme) 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.

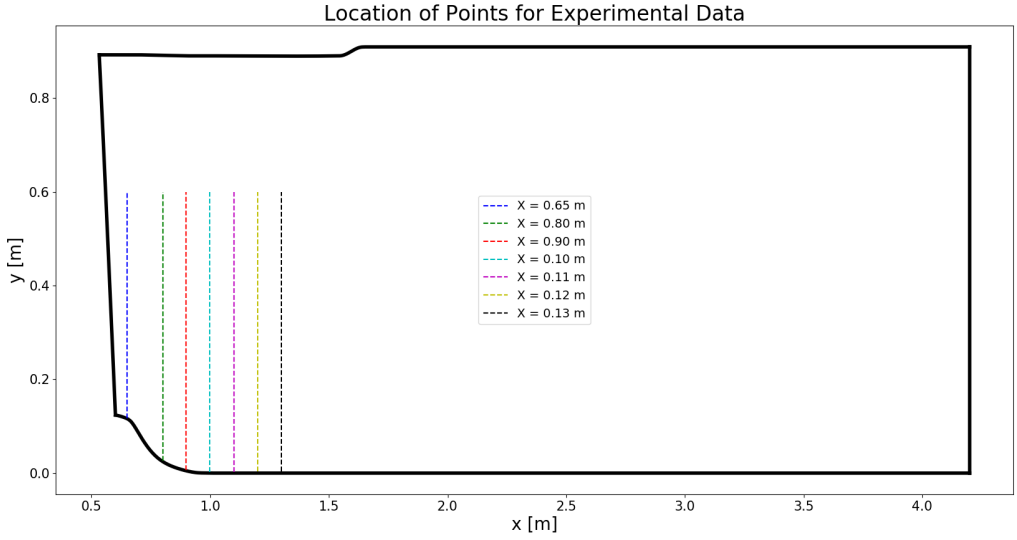


Figure 13: Locations for experimental data

2.1 Discretization schemes

From figure 13, it can be seen that the values at x = 0.65 m corresponds to where we have an attached boundary layer whereas, for x > 0.8 m we have the recirculation region. At x = 0 in further images represents the inlet shown in figure 13. Figure 14 shows the values of $\langle \overline{u} \rangle$ velocity obtained from data in reference [169] and [203] and are compared with the experimental data at the locations mentioned above. It can be observed that the simulation results matches closely with the experiments. On the other hand, figure 15 shows plots of $\langle \overline{v} \rangle$ velocity, which shows a relatively good match (similar trend of velocity profile) with experiments apart from values at x = 0.65 m.

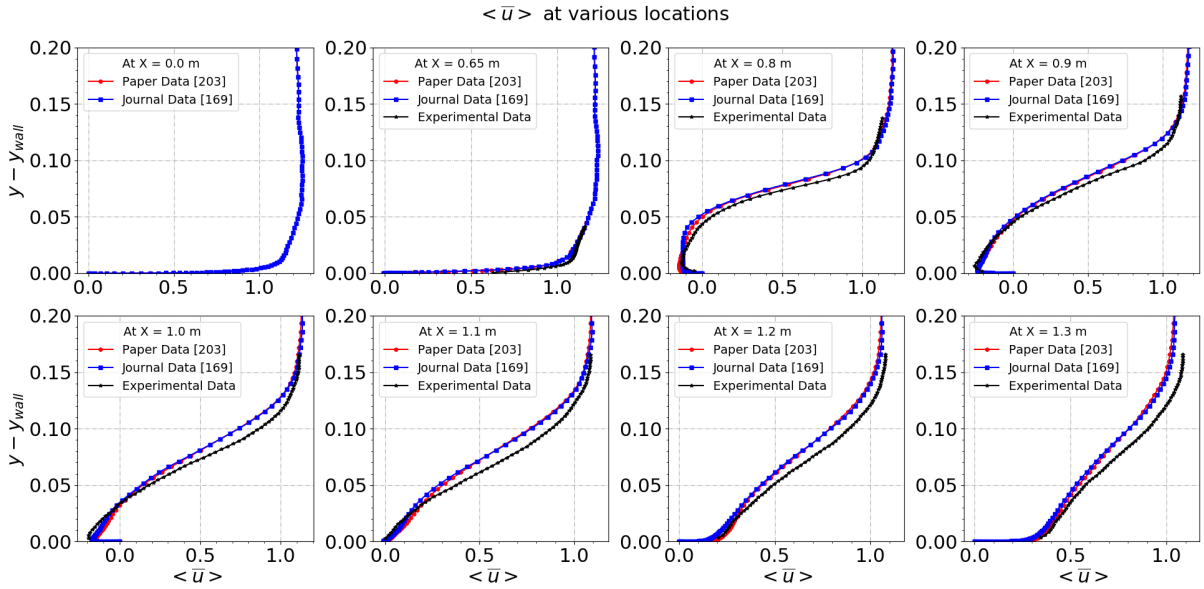


Figure 14: $\langle \overline{u} \rangle$ at various locations

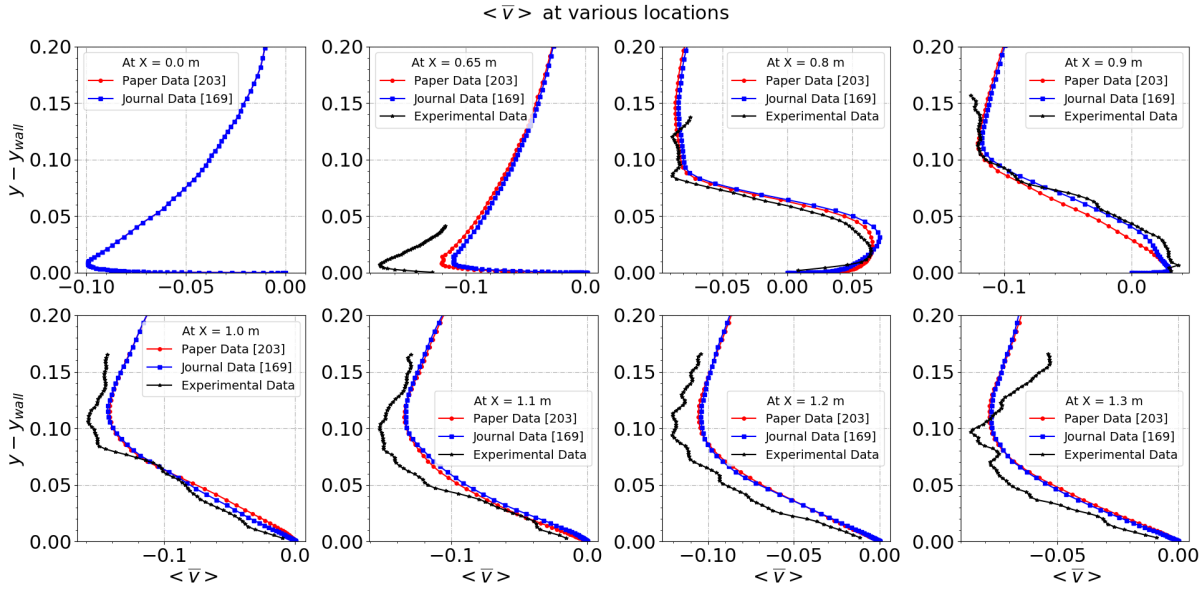


Figure 15: $\langle \overline{v} \rangle$ at various locations

Another observation which can be made from figure 14 is regarding the separation and reattachment points predicted by reference [169] and [203] and its comparison with the experimental data.

- Figure 16 shows the near wall values of $\langle \overline{u} \rangle$ at x = 0.65 m and x = 1.1 m. According to the experiments, the flow is attached at x = 0.65 m, but the simulations shows otherwise as we have flow in backward direction near the wall which implies the flow is separated at that point. Perhaps this can be the reason why we see a deviation in simulation and experimental value of $\langle \overline{v} \rangle$ at x = 0.65 m.
- At x = 1.1 m in figure 16, experimental values seems to suggest that the flow is not reattached,

which is correctly predicted by data obtained from reference [169]. But data from reference [203] shows that the flow is reattached at x = 1.1 m. This can conclude that the reattachment point is correctly predicted by the simulation methodology implemented in reference [169].

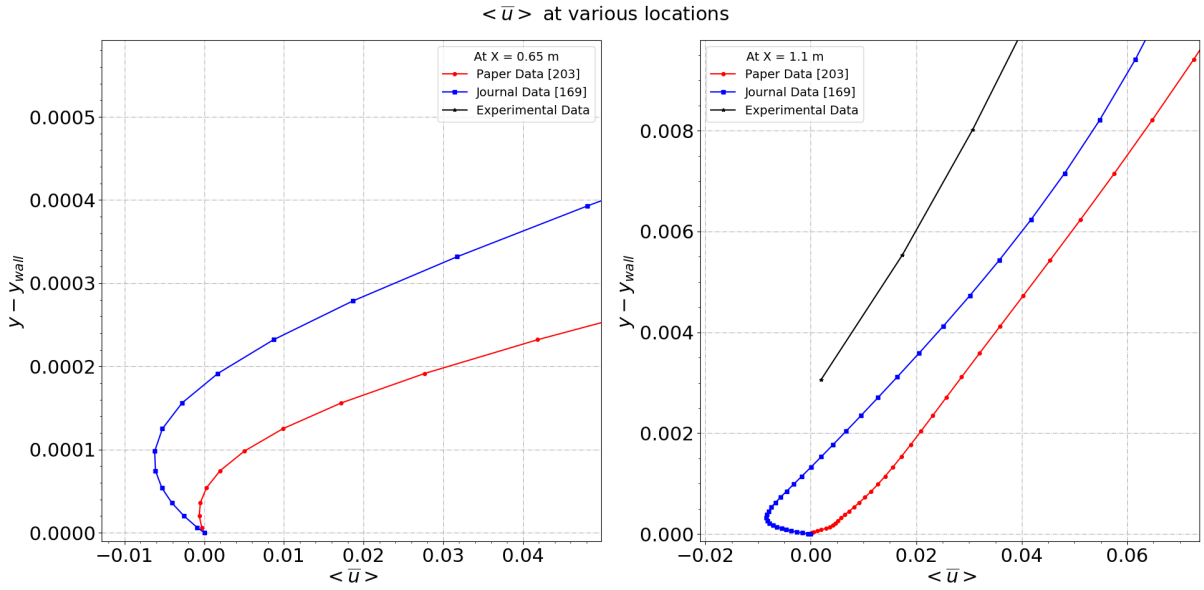


Figure 16: $\langle\overline{u}\rangle$ at x = 0.65 m and x = 1.1 m

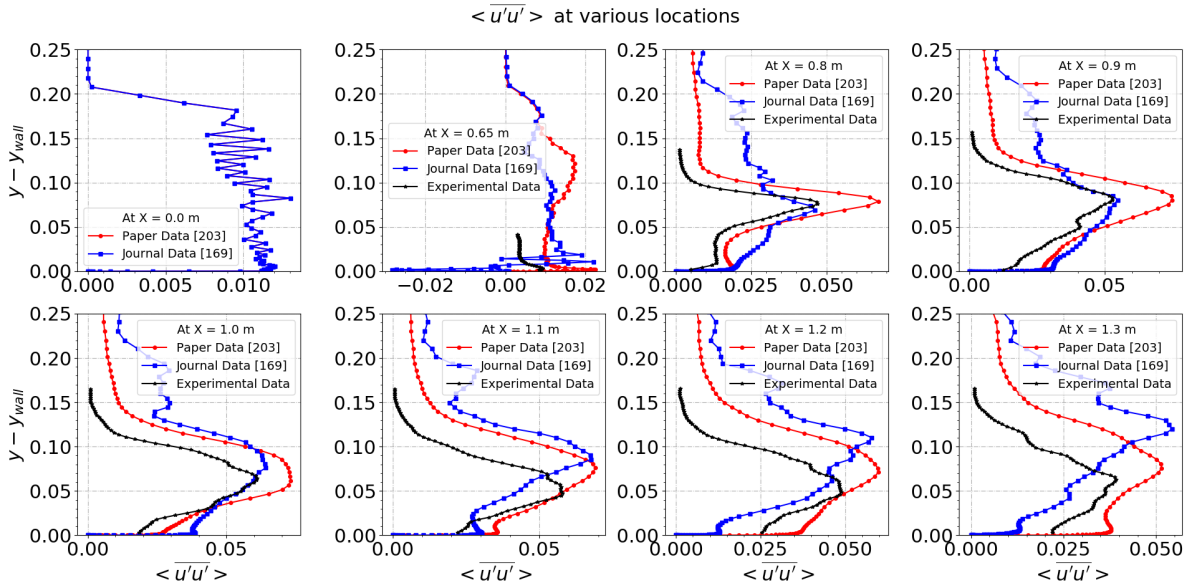


Figure 17: $\langle\overline{u'u'}\rangle$ at various locations

Figures 17, 18 and 19 shows the turbulent fluctuations $\langle\overline{u'u'}\rangle$, $\langle\overline{v'v'}\rangle$ and $\langle\overline{u'v'}\rangle$ respectively in the domain at some locations. We can see the synthetic fluctuations provided at the inlet of the domain. They are provided to generate large resolved turbulent fluctuations which in turn can help to resolve most of the turbulence in these regions.

- Consider the normal fluctuations ($\langle\overline{u'u'}\rangle$) at x = 0.65 m as shown in figure 17. We can see the

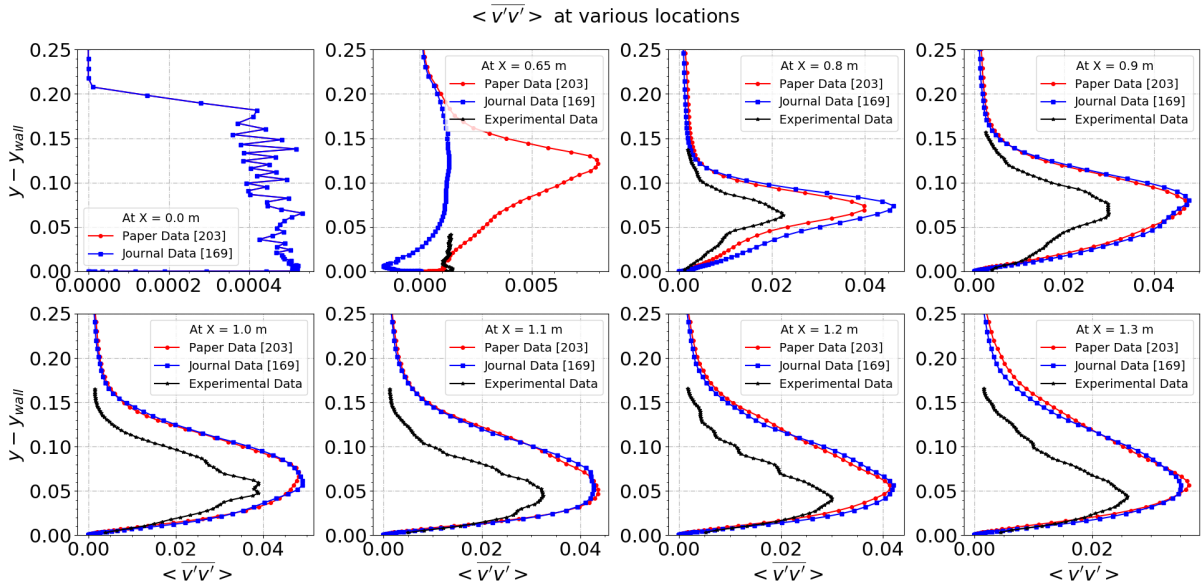


Figure 18: $\langle \overline{v'v'} \rangle$ at various locations

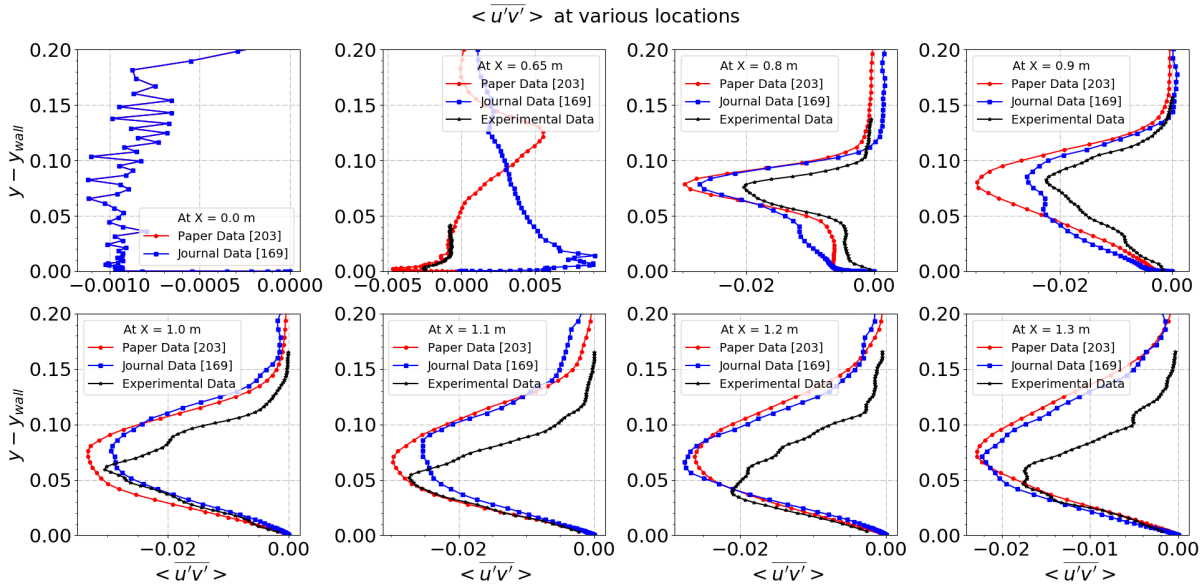


Figure 19: $\langle \overline{u'v'} \rangle$ at various locations

non-physical fluctuations in the bulk flow region (around $y > 0.05$) and it exists for both the data-sets (Reference [169] and [203]). The non-physical fluctuations are due to the synthetic inlet resolved fluctuations provided and die out in the outer region (around $y > 0.4$). Interesting thing is that in the recirculation region ($x > 0.8$), the non-physical fluctuations die out when using a central differencing scheme [203], but for a hybrid scheme [169] the fluctuations still persists in the recirculation region as well.

- When we look at the $\langle \overline{v'v'} \rangle$ fluctuations in figure 18, at x = 0.65 m the non-physical fluctuations in bulk flow region are very large for reference [203] than for reference [169] but they disappear further downstream in the recirculation region for both the data-sets. However, the turbulent

fluctuations are over-predicted in bulk flow region in both the references.

- For $\langle\overline{u'v'}\rangle$ turbulent fluctuations as shown in figure 19, when using the central differencing scheme non-physical fluctuations are present at x = 0.65 m, which are reduced by applying a hydrid central differencing (95%) and upwind scheme (5%).

2.2 Modeled turbulent shear stress

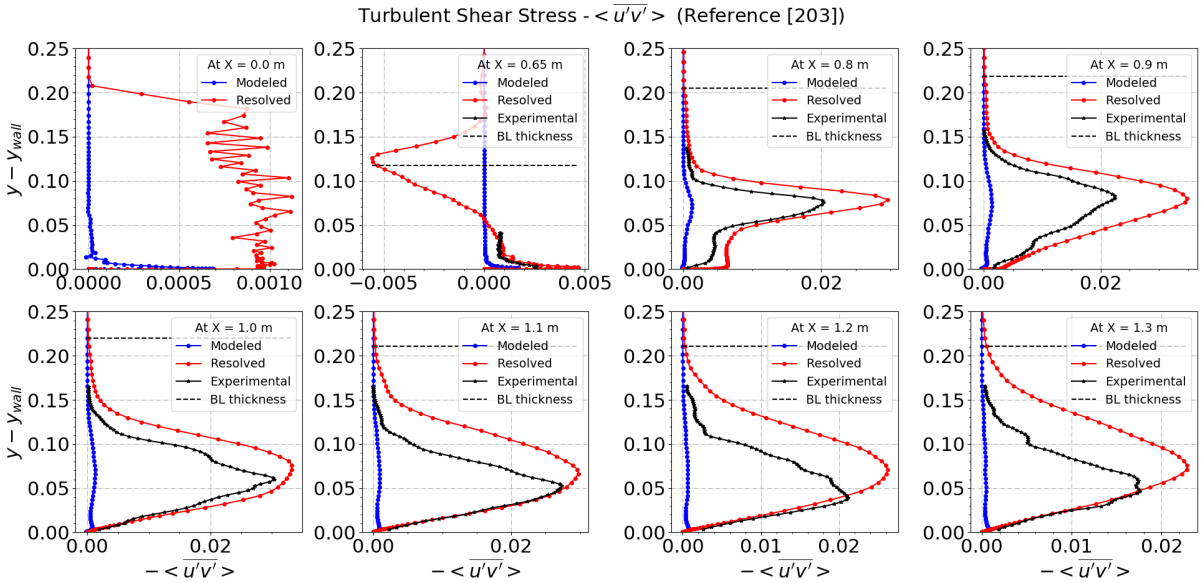


Figure 20: Turbulent shear stress -$\langle\overline{u'v'}\rangle$ at various locations (Data from reference[203])

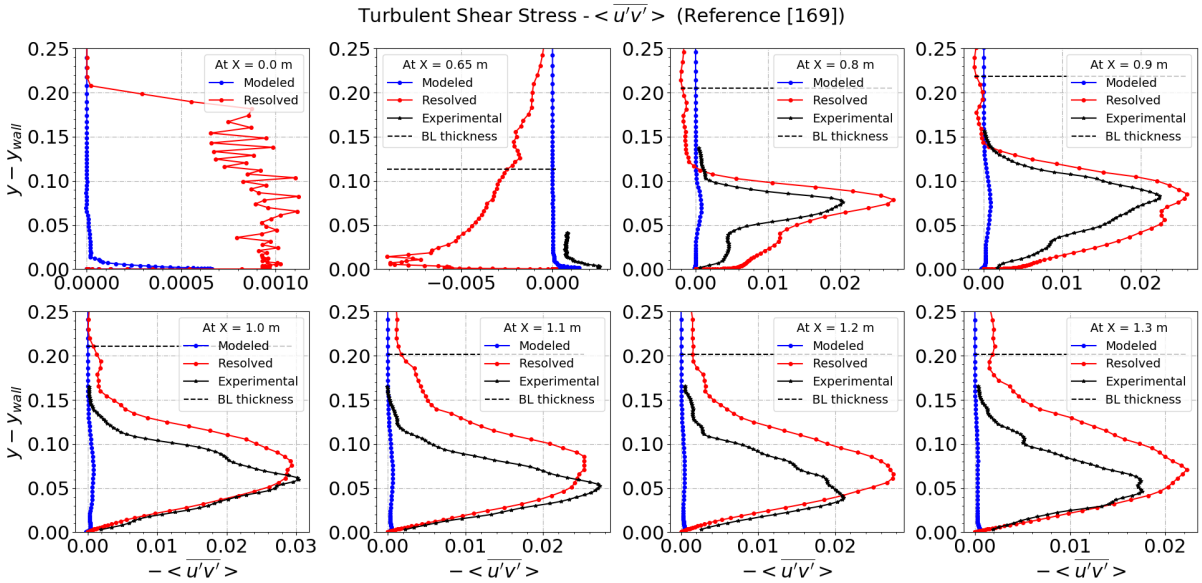
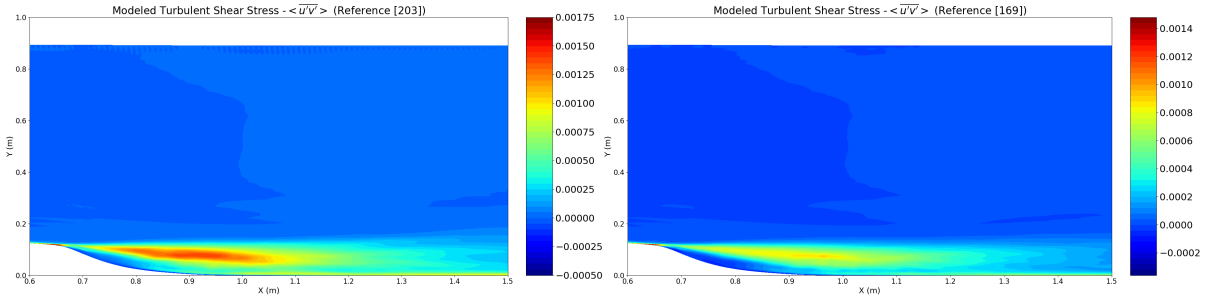


Figure 21: Turbulent shear stress -$\langle\overline{u'v'}\rangle$ at various locations (Data from reference[169])

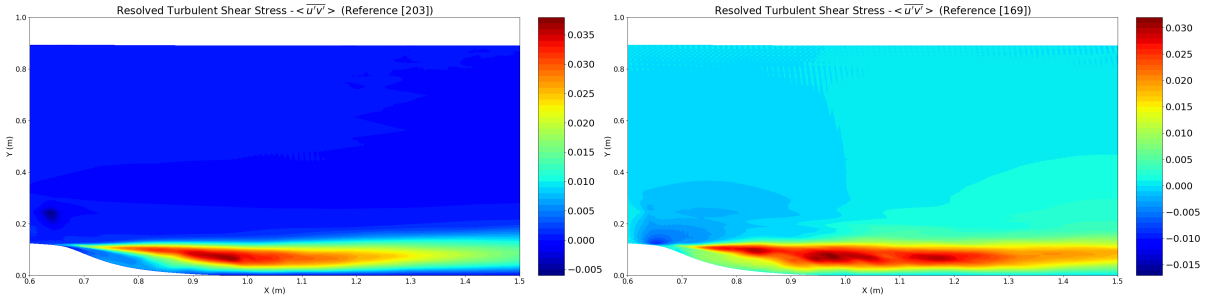
In this section, the turbulent shear stresses (-$\langle\overline{u'v'}\rangle$) are plotted at various experimental locations to see how much of the turbulence has been resolved and how much is modelled in the recirculation as well as boundary layer region. Figure 20 and figure 21 shows the shear stresses evaluated using reference [203] and [169] respectively. The turbulent boundary layer thickness is also calculated at the experimental locations - it is where the ratio of turbulent viscosity and viscosity (ν_t/ν) falls below one. When the turbulent viscosity ratio is less than the one, it means that the flow is laminar in that region. The flow is laminar in the viscous sub-layer near the wall and outside outside the boundary layer whereas it is turbulent everywhere else.

It can be observed that the magnitude of modeled stress is very small compared to the resolved stress. Hence, we can conclude that most of the turbulence in boundary layer (x = 0.65 m) and recirculation region (x > 0.8 m) is resolved. Figure 22 shows the contours of the modeled and resolved shear stress for both the references. The colorbar gives an idea of how small the modeled stress is compared to the resolved stress. Also it can be seen that the magnitude of resolved turbulence is higher in recirculation region than in the boundary layer.



(a) Modeled - Data from reference [203]

(b) Modeled - Data from reference [169]



(c) Resolved - Data from reference [203]

(d) Resolved - Data from reference [169]

Figure 22: Turbulent Shear Stress (-$\langle\overline{u'v'}\rangle$)

2.3 Turbulent viscosity

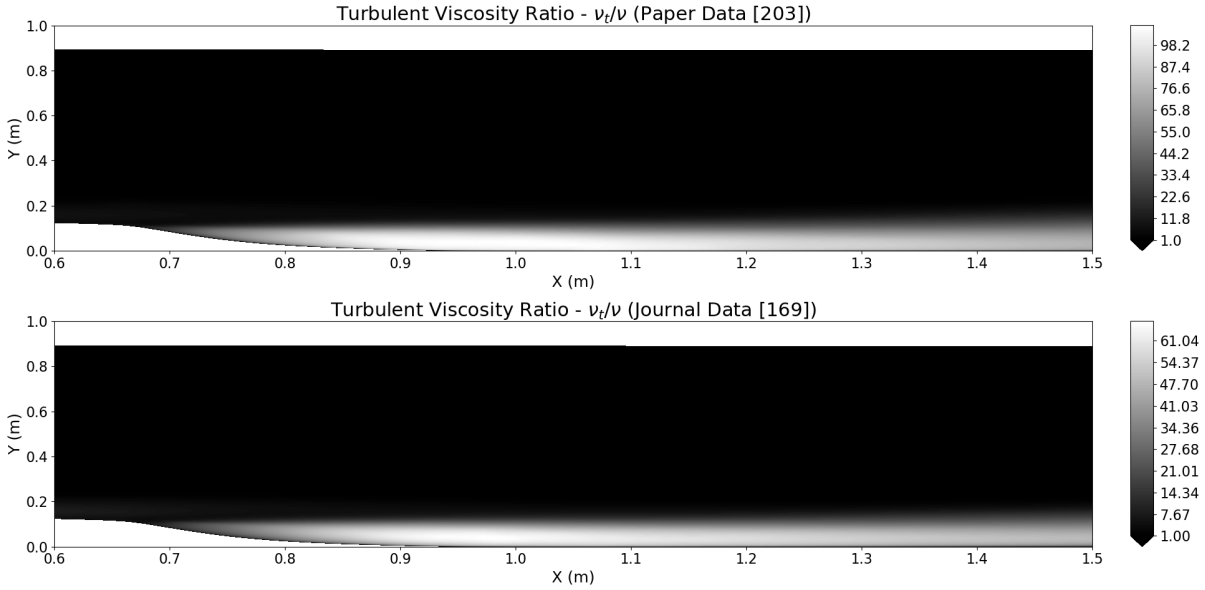


Figure 23: Turbulent viscosity ratio (ν_t/ν)

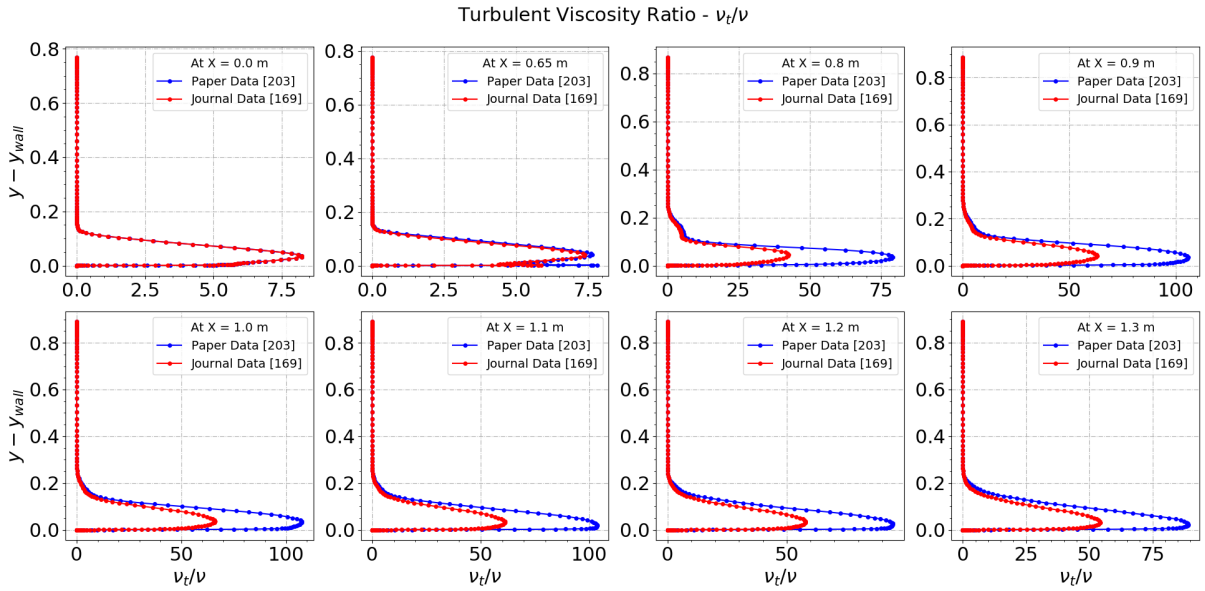


Figure 24: Turbulent viscosity ratio (ν_t/ν)

Figure 23 shows the contour of viscosity ratio (ν_t/ν) calculated using data from both the references [169] and [203]. In general, the viscosity ratio is higher in the recirculating region and it is below 1 near the wall in most of the outer region. This means that the flow is turbulent in recirculating region where $\nu_t/\nu > 1$. As the flow is laminar outside the boundary layer and also in the viscous sub-layer of the boundary layer near the wall, the viscosity ratio there is below 1.

From figure 24 it can be concluded that the viscosity ratio is higher for reference [203] in the recirculation region (x $>$ 0.8 m). But it is nearly the same in attached boundary layer region (x $<$ 0.65 m). The

difference is basically due the way turbulent prandtl number is calculated in both papers. This in turns affect the calculation of turbulent kinetic energy and dissipation and effectively the turbulent viscosity ($\nu_t = C_\mu \frac{k^2}{\epsilon}$).

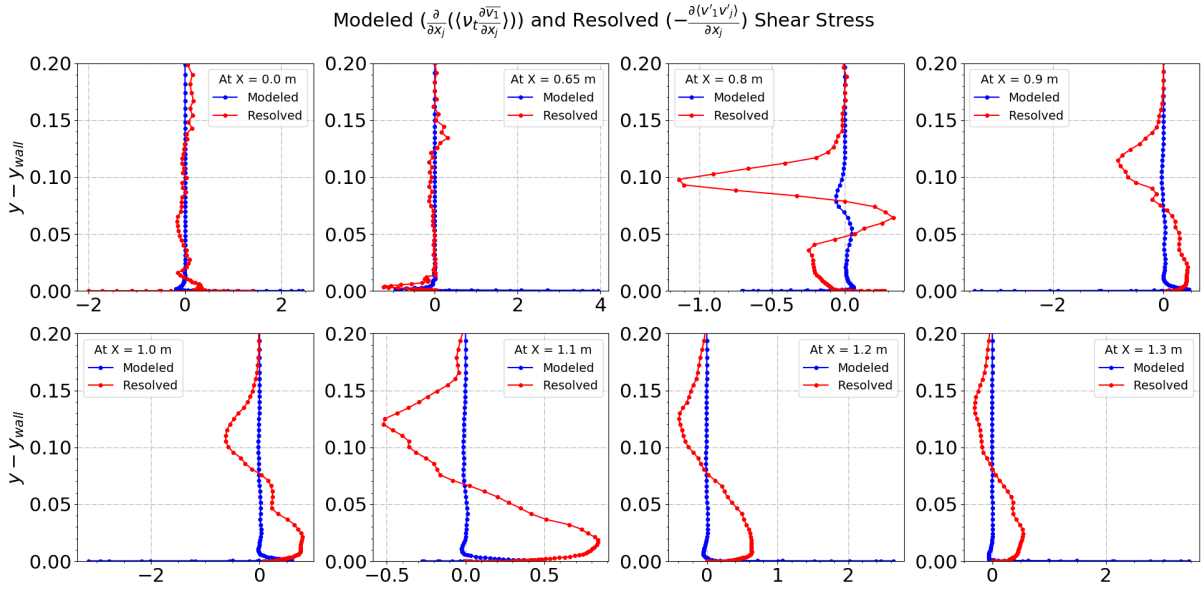


Figure 25: Modeled and resolved term in momentum equation (for i=1)

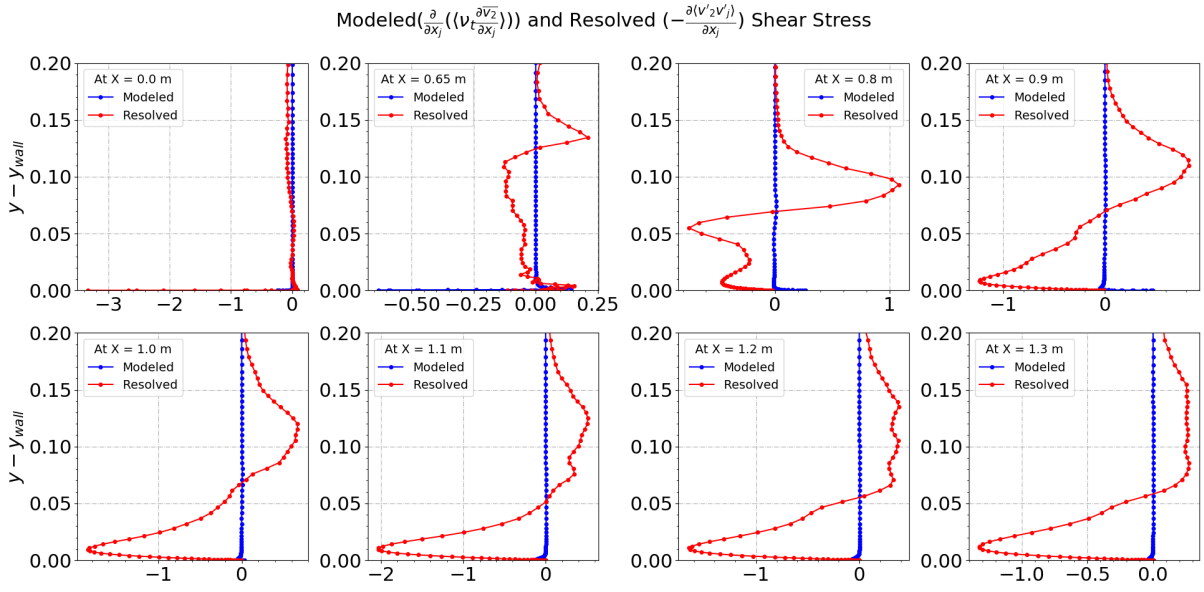


Figure 26: Modeled and resolved term in momentum equation (for i=2)

In figure 25 and figure 26, two terms from the momentum equations are plotted - modeled ($\frac{\partial}{\partial x_j}(\langle \nu_t \frac{\partial \overline{v_1}}{\partial x_j} \rangle)$) and resolved ($-\frac{\partial \langle v'_1 v'_j \rangle}{\partial x_j}$) shear stress terms. In both the figures (for i=1 and i=2), the magnitude of resolved stress is higher than the modeled stress. A similar conclusion to what was discussed for figure 20 and figure 21 in section 2.2.

2.4 Location of interface

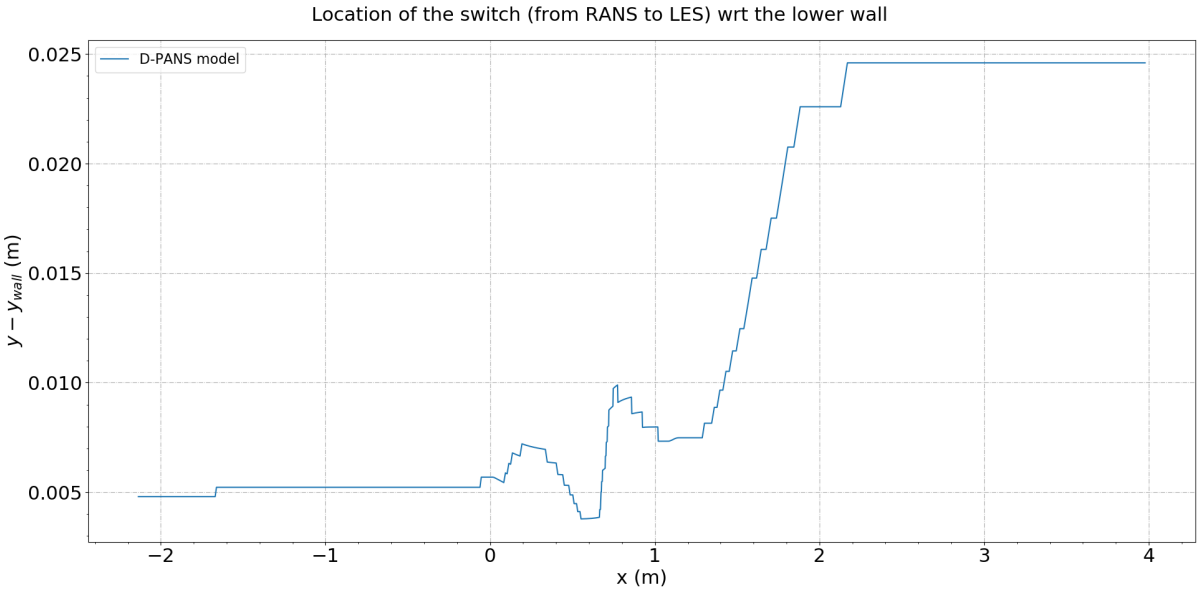


Figure 27: Location of interface for D-PANS model

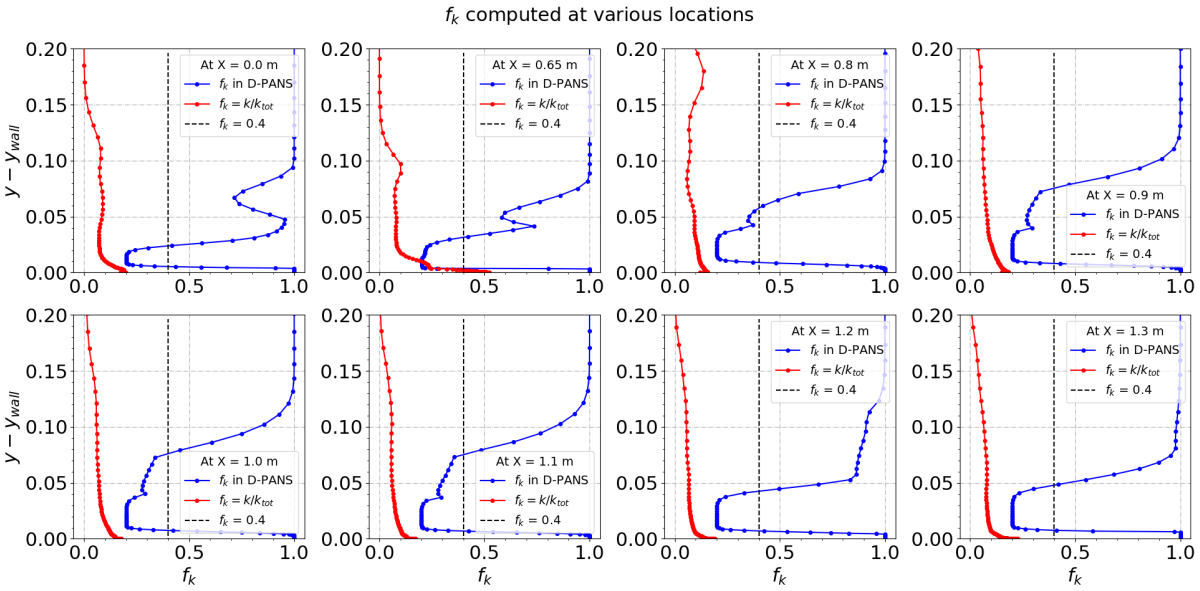


Figure 28: Comparison of f_k obtained using (i)D-PANS and (ii)k/k_{tot}

Figure 27 shows the location of the interface for the D-PANS model. In PANS the model switches from RANS mode to LES when the value of f_k drops beow 0.4. Below the interface, it is a RANS region and above it is LES region i.e. ν_t is the RANS viscosity in RANS region and it is SGS viscosity in LES region. Theoretically, f_k is the ratio of modeled and total kinetic energy. But in this simulation of D-PANS, a new formulation of f_k is given and compared with the theoretical definition and also with the definition as provided in **Reference [216]** - B. Basara, S. Krajnovi´c, S. Girimaji, and Z. Pavlovi´c. *'Near-wall formulation of the Partially Averaged Navier Stokes turbulence model.'* AIAA

Journal, 49(12):2627–2636, 2011.

$$\text{Theoretical definition of } f_k = \frac{k}{k_{tot}}$$
$$f_k \text{ as per reference [216]} = C_\mu{}^{-}0.5(\frac{\Delta}{L_t})^{2/3} \tag{6}$$

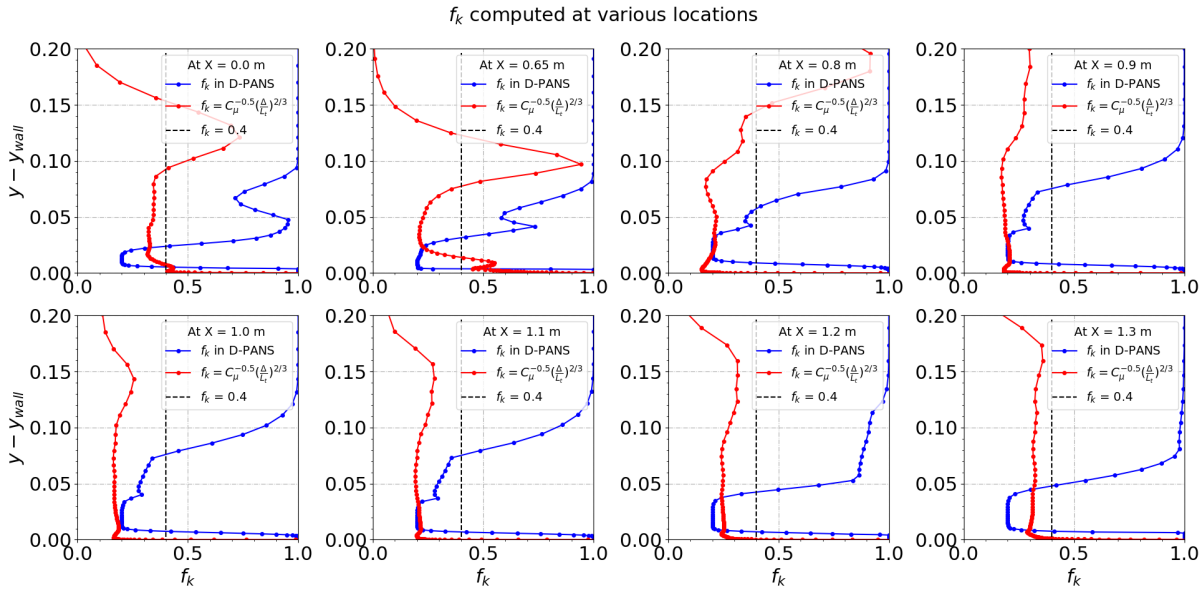


Figure 29: Comparison of f_k obtained using (i)D-PANS and (ii)Reference [216] - Basara

Figure 28 and figure 29 compares the various formulations of f_k.

- If we see the values of f_k for D-PANS formulation, very near the wall it is more than 0.4, but it goes below 0.4 for some distance and again increases quickly to 1 in bulk flow region. Hence, the RANS mode is activated only near the wall in D-PANS and in bulk flow region.
- According to definition of f_k in reference [216], near the wall the RANS mode is used as f_k is higher than 0.4. But unlike the D-PANS model, in the bulk flow region f_k is less than 0.4 and functions in LES mode.
- f_k as per theoretical definition is always less than 0.4.

2.5 Location of interface in DES and DDES

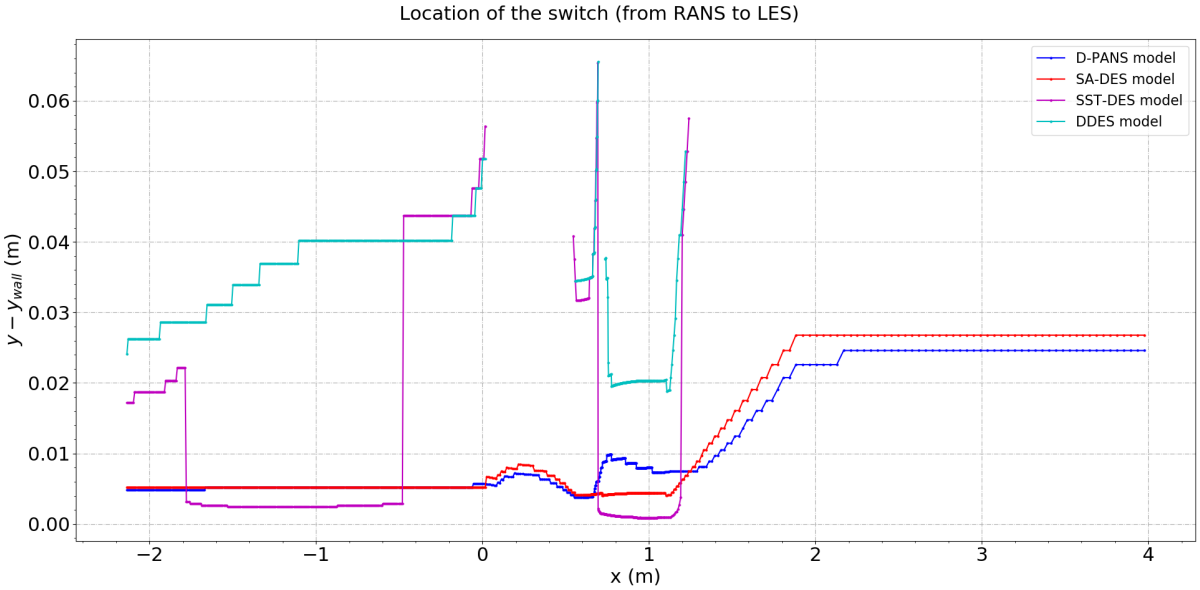


Figure 30: Location of interface for various models

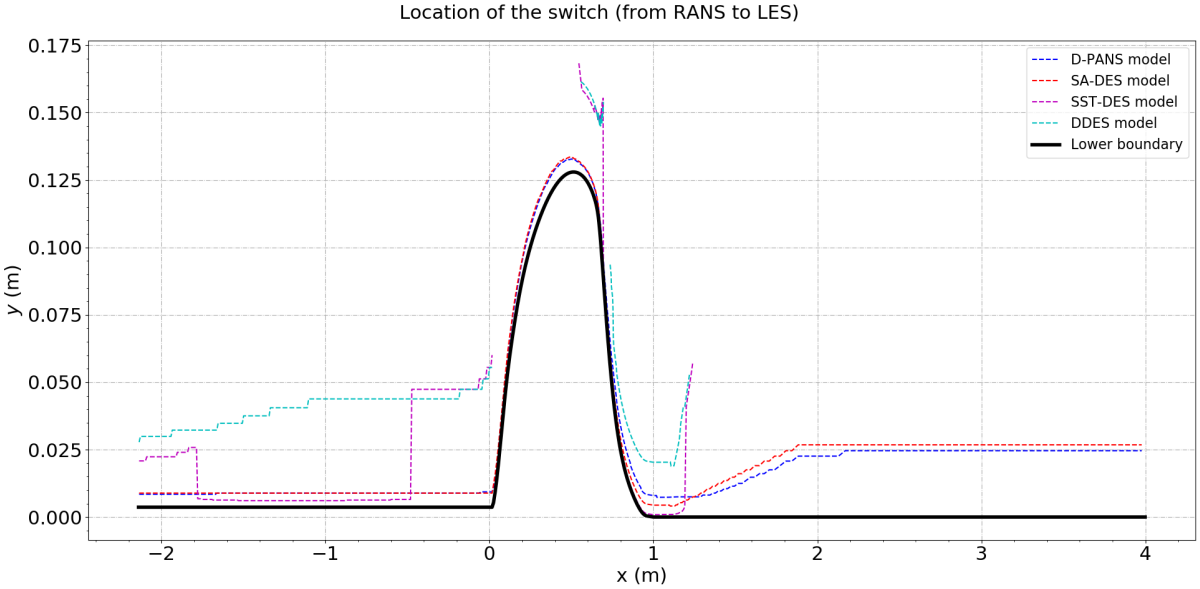


Figure 31: Location of interface for various models - visualisation wrt lower wall

Figure 30 shows the location of switch line obtained for various models. The concept off switch lines was discussed in section 1.5 for different models like SA-DES, SST-DES and DDES. The switch line for DDES model is based on the location where f_k goes below 0.4 as shown in section 2.4. Figure 31 is the same as figure 30, the only difference is that the switch location can be visualised wrt the lower boundary of the domain where the hump is located.

- For the D-PANS and SA-DES model, the interface is located very close to the lower wall till the

recirculation region. In the recirculation region however the switch takes place at a later stage.

- For SST-DES and DDES model, there are some locations where the switching line is not plotted - it means that the model is in RANS mode throughout at that location. Also, the DDES model has the interface higher than the other models especially before and after the hump. Hence it can be said that DDES works and makes the model to be in RANS mode in most part of the boundary layer.

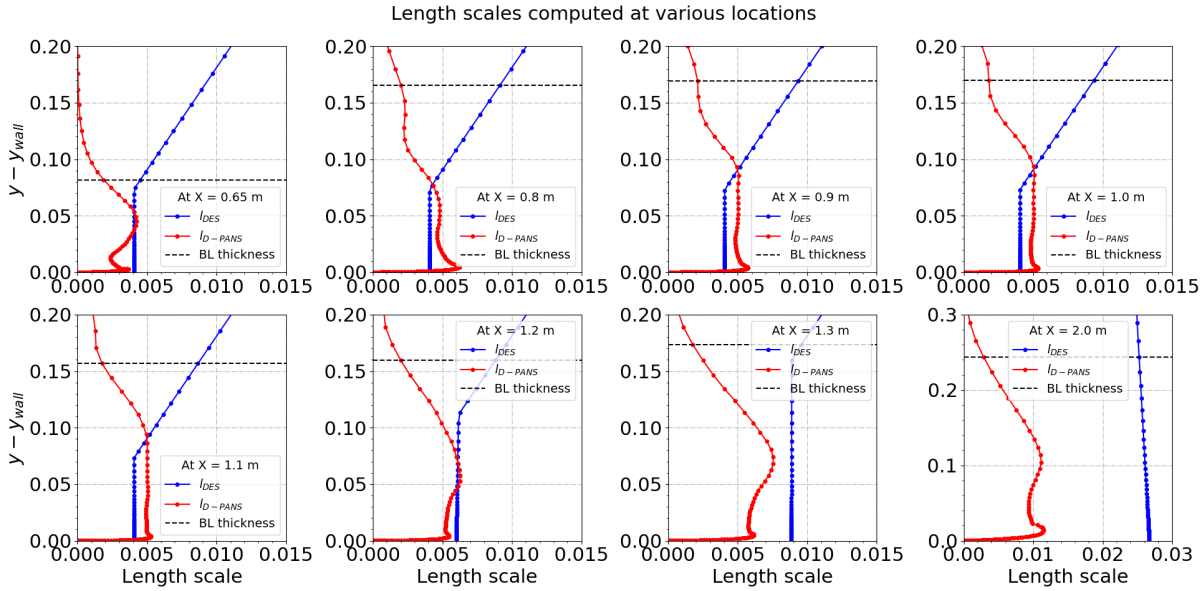


Figure 32: Length scale computed for DES and D-PANS model

Figure 32 compares the DES and D-PANS length scales, the boundary layer is also shown at the experimental locations and calculated in the same way as explained in section 2.2. PANS length scale decreases after the turbulent boundary layer and at x = 2 m, it is larger than at other locations in the recirculation region.

$$l_{DES} = C_{DES}\Delta$$
$$l_{PANS} = k^{1.5}/\epsilon \tag{7}$$

2.6 SAS model

Figure 33: Length scale computed for SAS model

Figure 33 shows the Von-karman length scales computed using the formulas as shown in section 1.6. The Von-karman length scales for the mean and instantaneous velocities are plotted and compared at the experimental locations and also compared with the PANS and DES length scales calculated in section 2.5.

The PANS and DES length scales are smaller as compared to the Von-karman length scales. Von-karman length scale for instantaneous velocity ($L_{vk,3D,s,type1}$ and $L_{vk,3D,s,type2}$) and Von-karman length scale for time averaged velocity ($L_{vk,3D,mean,type1}$ and $L_{vk,3D,mean,type2}$) are computed using two methods as

shown previously. It can be seen that the length scales for time averaged velocity are bigger than the length scales for instantaneous velocity at most of the locations.

In figure 34, the von-karman length scale for time averaged velocity is compared with the RANS length scale. It can be seen that the RANS length scale is very large as compared to the von-karman length scales.

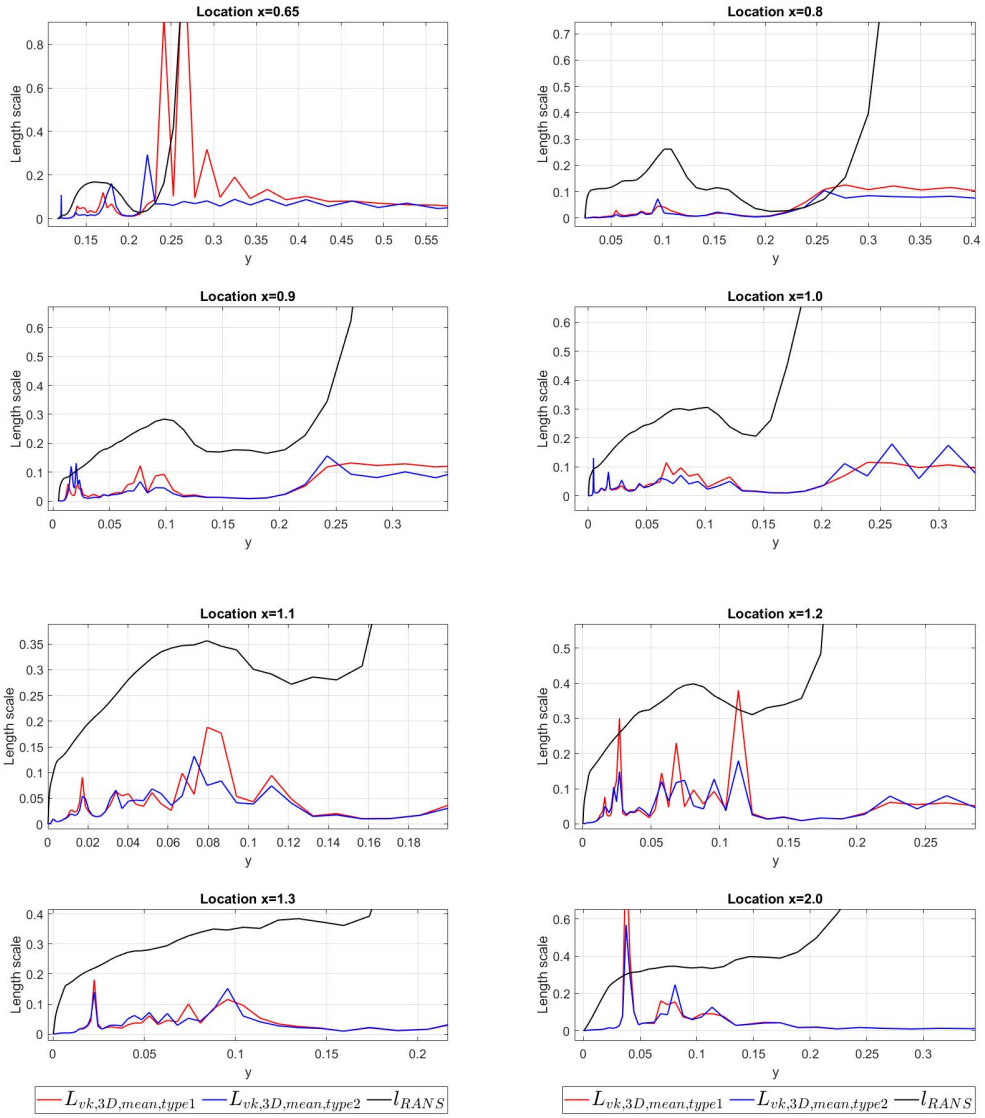


Figure 34: Length scale computed for SAS and RANS model

2.7 Different ways to evaluate resolution

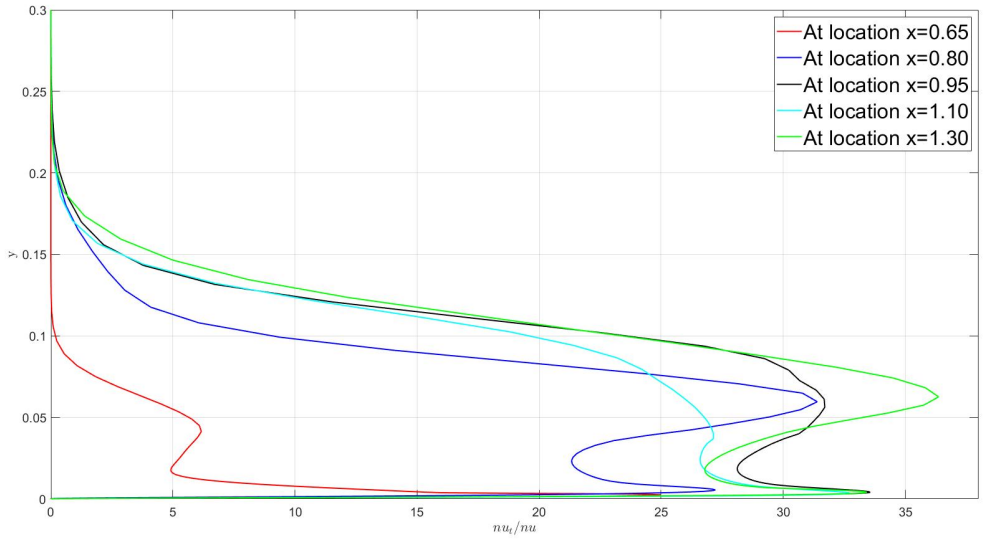


Figure 35: Ratio Between Viscous and Modelled Turbulent Viscosity

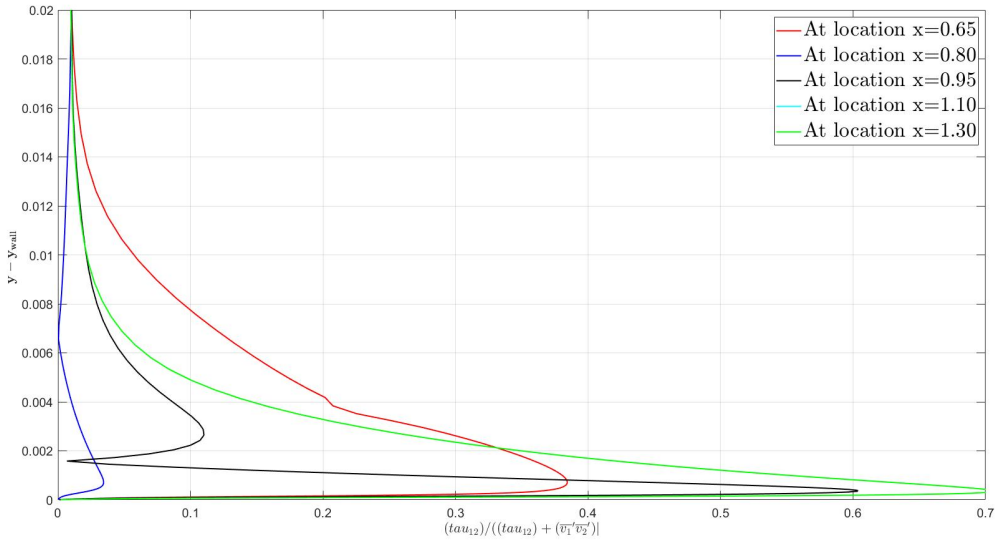


Figure 36: Ratio Between Modeled and Total Shear Stress

In figure 35, the ratio between viscous viscosity and turbulent viscosity is computed and plotted at different locations. The viscosity ratio has the highest values at the re-circulation region and immediate downstream from the hill. This graph depicts on how close the LES is to DNS, however it does not say how good the LES resolution is.

In figure 36, The plot shows the ratio between modeled and total shear stress.The modeled shear stress is computed from the Boussinesq approximation. Regions with small ratio values depicts good resolution

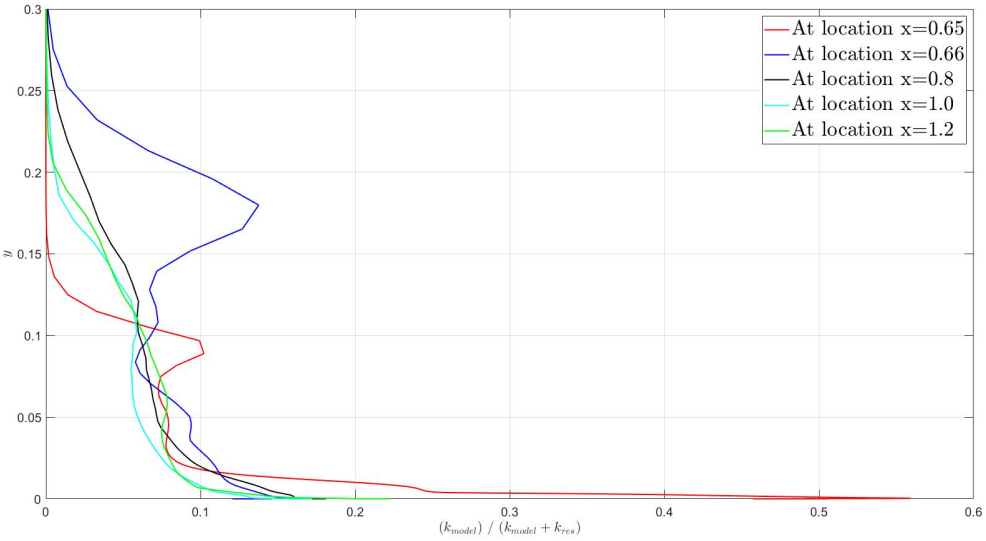


Figure 37: Ratio Between the Modelled and Total Turbulent Kinetic Energy

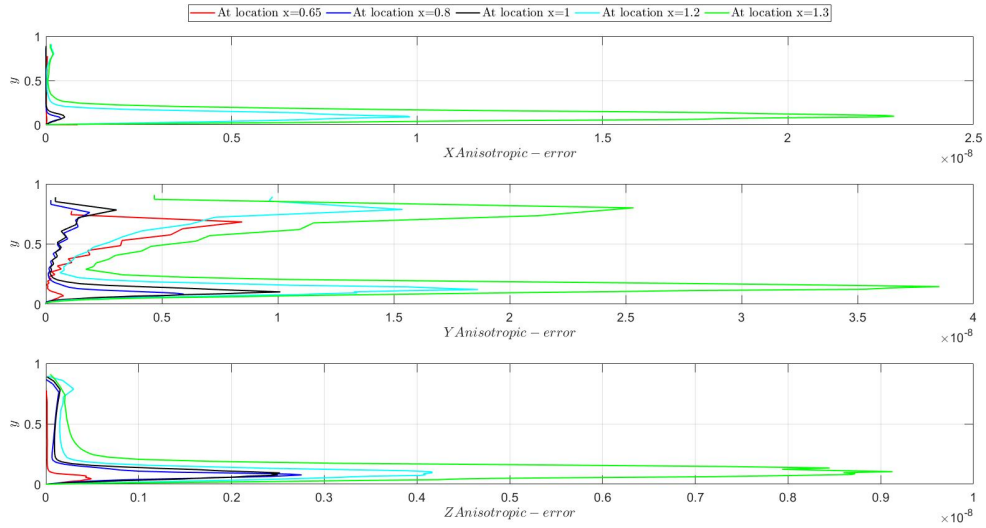


Figure 38: Anisotropic error

of turbulence. In figure 37, the plot shows the ratio between modeled and total turbulent kinetic energy. Similar to the shear stress plots, the lower ratios between modeled and total turbulent kinetic energy depict better resolved turbulence.

In figure 38, the anisotropic error is used as a tool to determine the quality of the grid throughout the computational domain. It helps determining regions that need refinement. It is evident that all X,Y and Z require higher refinement near the wall.

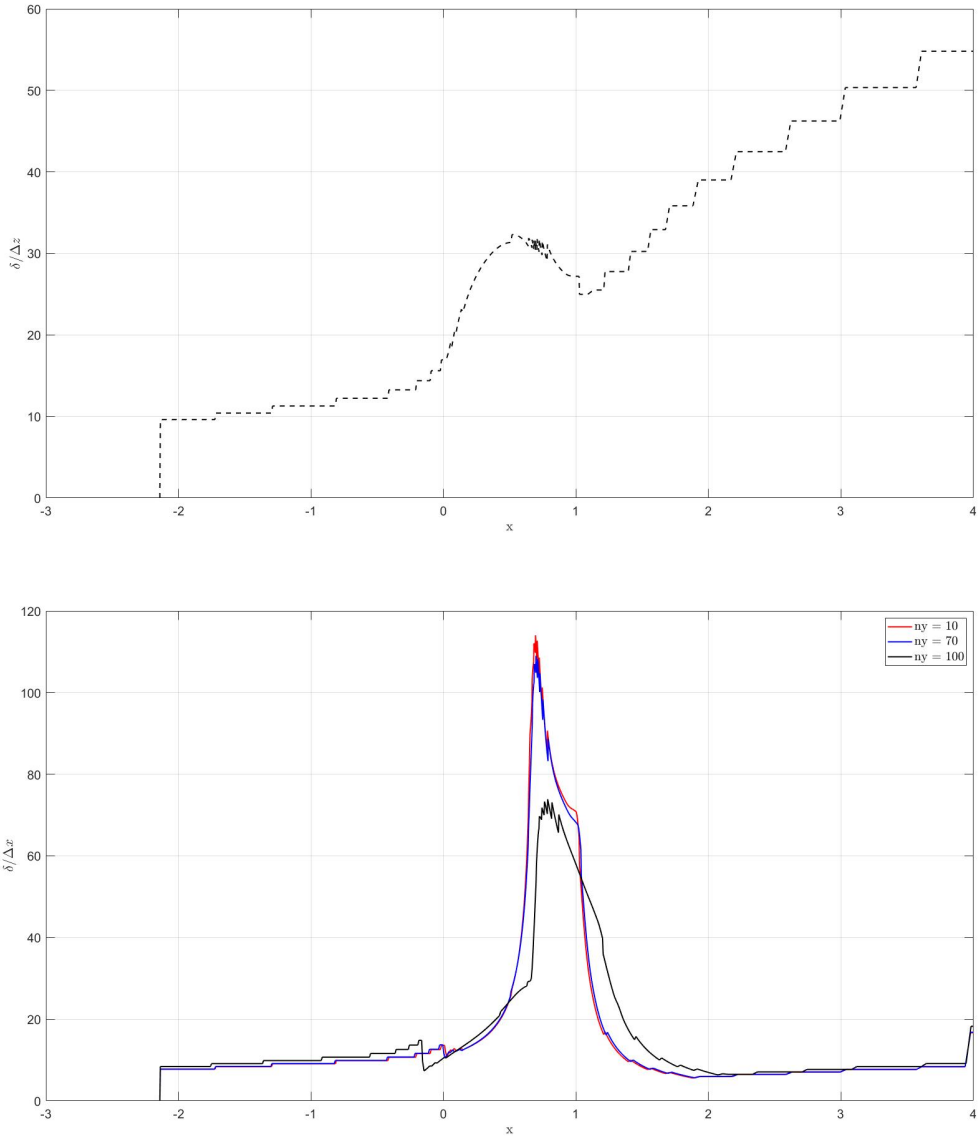


Figure 39: Ratio of boundary layer thickness to cell length

A Appendix

For Assignment 2a - Section 1.1

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import welch, hann
from IPython import display
from matplotlib.ticker import (MultipleLocator, FormatStrFormatter,
                               AutoMinorLocator)
#plt.rcParams.update({'font.size': 20})

# makes sure figures are updated when using ipython
display.clear_output()

# ***** read u
data = np.genfromtxt("u_w_time_5nodes_sdes.dat", dtype=None)

u1=data[:,0]   #v_1 at point 1
u2=data[:,1]   #v_1 at point 2
u3=data[:,2]   #v_1 at point 3
u4=data[:,3]   #v_1 at point 4
u5=data[:,4]   #v_1 at point 5

w1=data[:,5]   #w_1 at point 1
w2=data[:,6]   #w_1 at point 2
w3=data[:,7]   #w_1 at point 3
w4=data[:,8]   #w_1 at point 4
w5=data[:,9]   #w_1 at point 5

#print("u1=",u1)

dx=3.2/32
dt= 0.25*dx/20
t_tot=dt*len(u1)

t = np.linspace(0,t_tot,len(u1))

# %%%%%%%%%%%%%%%% plotting section %%%%%%%%%%%%%%%%%%%%%%%%%%
# plot u
#fig1 = plt.figure("Figure 1")
#plt.plot(t,u1,'b--')
#plt.plot(t,u4,'r-')
#plt.xlabel("t")
#plt.ylabel("u")
#plt.savefig('utime_python.eps',bbox_inches='tight')

####################### # zoom
#fig2 = plt.figure("Figure 2")
#plt.plot(t,u1,'b--')
#plt.plot(t,u4,'r-')
#plt.xlabel("t")
#plt.ylabel("u")
#plt.axis([6, 7, 10, 22])
#plt.savefig('utime_zoom_python.eps',bbox_inches='tight')

# %% Assignment S1 - Time History
# Plot Time Variation of V1 at all nodes
fig1,axes = plt.subplots(2,1,constrained_layout=True)
axes = axes.reshape(-1)
axes[0].plot(t,u1,color='red',label=r'$y/\delta$ = 0.0028 (Node 1)')
axes[0].plot(t,u2,color='orange',label=r'$y/\delta$ = 0.0203 (Node 2)')
axes[0].plot(t,u3,color='blue',label=r'$y/\delta$ = 0.0364 (Node 3)')
axes[0].plot(t,u4,color='green',label=r'$y/\delta$ = 0.0645 (Node 4)')
axes[0].plot(t,u5,color='black',label=r'$y/\delta$ = 0.2 (Node 5)')
```

```
axes[1].plot(t,u1,color='red',label=r'$y/\delta$ = 0.0028 (Node 1)')
axes[1].plot(t,u2,color='orange',label=r'$y/\delta$ = 0.0203 (Node 2)')
axes[1].plot(t,u3,color='blue',label=r'$y/\delta$ = 0.0364 (Node 3)')
axes[1].plot(t,u4,color='green',label=r'$y/\delta$ = 0.0645 (Node 4)')
axes[1].plot(t,u5,color='black',label=r'$y/\delta$ = 0.2 (Node 5)')
axes[1].set_xlim(6,7)
axes[1].set_ylim(10,27)
for ax in axes:
    ax.set_xlabel(r'Time ($t$)')
    ax.set_ylabel(r'$\overline{u}/u_{\tau}$')
    ax.legend(fontsize=16,loc='best')
    ax.tick_params(labelsize='medium')
    ax.xaxis.set_minor_locator(AutoMinorLocator())
    ax.yaxis.set_minor_locator(AutoMinorLocator())
    ax.grid(True, linestyle='-.')
fig1.suptitle(r'Time variation of $\overline{u}$ at various nodes',fontsize=22)

# Plot Time Variation of V3 at all nodes
fig2,axes = plt.subplots(2,1,constrained_layout=True)
axes = axes.reshape(-1)
axes[0].plot(t,w1,color='red',label=r'$y/\delta$ = 0.0028 (Node 1)')
axes[0].plot(t,w2,color='orange',label=r'$y/\delta$ = 0.0203 (Node 2)')
axes[0].plot(t,w3,color='blue',label=r'$y/\delta$ = 0.0364 (Node 3)')
axes[0].plot(t,w4,color='green',label=r'$y/\delta$ = 0.0645 (Node 4)')
axes[0].plot(t,w5,color='black',label=r'$y/\delta$ = 0.2 (Node 5)')
axes[1].plot(t,w1,color='red',label=r'$y/\delta$ = 0.0028 (Node 1)')
axes[1].plot(t,w2,color='orange',label=r'$y/\delta$ = 0.0203 (Node 2)')
axes[1].plot(t,w3,color='blue',label=r'$y/\delta$ = 0.0364 (Node 3)')
axes[1].plot(t,w4,color='green',label=r'$y/\delta$ = 0.0645 (Node 4)')
axes[1].plot(t,w5,color='black',label=r'$y/\delta$ = 0.2 (Node 5)')
axes[1].set_xlim(6,7)
axes[1].set_ylim(-3,2)
for ax in axes:
    ax.set_xlabel(r'Time ($t$)')
    ax.set_ylabel(r'$\overline{w}/u_{\tau}$')
    ax.legend(fontsize=16,loc='best')
    ax.tick_params(labelsize='medium')
    ax.xaxis.set_minor_locator(AutoMinorLocator())
    ax.yaxis.set_minor_locator(AutoMinorLocator())
    ax.grid(True, linestyle='-.')
fig2.suptitle(r'Time variation of $\overline{w}$ at various nodes',fontsize=22)

# Compute Auto-Corelation
u1_fluct = u1 - np.mean(u1)
u2_fluct = u2 - np.mean(u2)
u3_fluct = u3 - np.mean(u3)
u4_fluct = u4 - np.mean(u4)
u5_fluct = u5 - np.mean(u5)

B11_1 = np.correlate(u1_fluct,u1_fluct,'full')
nmax = np.argmax(B11_1)
magmax = np.max(B11_1)
norm_B11_1 = B11_1[nmax:]/magmax

B11_2 = np.correlate(u2_fluct,u2_fluct,'full')
nmax = np.argmax(B11_2)
magmax = np.max(B11_2)
norm_B11_2 = B11_2[nmax:]/magmax

B11_3 = np.correlate(u3_fluct,u3_fluct,'full')
nmax = np.argmax(B11_3)
magmax = np.max(B11_3)
norm_B11_3 = B11_3[nmax:]/magmax

B11_4 = np.correlate(u4_fluct,u4_fluct,'full')
nmax = np.argmax(B11_4)
```

```
magmax = np.max(B11_4)
norm_B11_4 = B11_4[nmax:]/magmax

B11_5 = np.correlate(u5_fluct,u5_fluct,'full')
nmax = np.argmax(B11_5)
magmax = np.max(B11_5)
norm_B11_5 = B11_5[nmax:]/magmax

# Plotting Auto-Corelation
imax = 2000
fig3,axes = plt.subplots(2,1,constrained_layout=True)
axes[0].plot(t[0:imax],norm_B11_1[0:imax],color='red',label=r'$y/\delta$ = 0.0028 (Node 1)')
axes[0].plot(t[0:imax],norm_B11_2[0:imax],color='orange',label=r'$y/\delta$ = 0.0203 (Node 2)')
axes[0].plot(t[0:imax],norm_B11_3[0:imax],color='blue',label=r'$y/\delta$ = 0.0364 (Node 3)')
axes[1].plot(t[0:imax],norm_B11_4[0:imax],color='cyan',label=r'$y/\delta$ = 0.0645 (Node 4)')
axes[1].plot(t[0:imax],norm_B11_5[0:imax],color='black',label=r'$y/\delta$ = 0.2 (Node 5)')
for ax in axes:
    ax.set_xlabel(r'Time ($t$)')
    ax.set_ylabel(r'$B_{uu}$')
    ax.legend(fontsize=16,loc='best')
    ax.tick_params(labelsize='medium')
    ax.xaxis.set_minor_locator(AutoMinorLocator())
    ax.yaxis.set_minor_locator(AutoMinorLocator())
    ax.grid(True, linestyle='-.')
    ax.set_xlim([0,0.07])
fig3.suptitle(r'Auto-Corelation ($B_{uu}$) at various nodes',fontsize=22)

# Compute Integral Timescale
# All normalised auto-corelations go negative. Find the instance when it goes
# negative and integrate till that point..
norm_B11_1_M = norm_B11_1[0:np.argmax(norm_B11_1<=0)]
norm_B11_2_M = norm_B11_2[0:np.argmax(norm_B11_2<=0)]
norm_B11_3_M = norm_B11_3[0:np.argmax(norm_B11_3<=0)]
norm_B11_4_M = norm_B11_4[0:np.argmax(norm_B11_4<=0)]
norm_B11_5_M = norm_B11_5[0:62]
#norm_B11_5_M = norm_B11_5[0:np.argmax(norm_B11_5<=0)]

int_T_1 = np.trapz(norm_B11_1_M) * dt
int_T_2 = np.trapz(norm_B11_2_M) * dt
int_T_3 = np.trapz(norm_B11_3_M) * dt
int_T_4 = np.trapz(norm_B11_4_M) * dt
int_T_5 = np.trapz(norm_B11_5_M) * dt
```

For Assignment 2a - Section 1.2 to Section 1.7

```
import scipy.io as sio
import numpy as np
import matplotlib.pyplot as plt
from IPython import display
plt.rcParams.update({'font.size': 22})

# makes sure figures are updated when using ipython
display.clear_output()

dx=0.05
dz=0.025

ni=34
nj=49
nk=34

viscos=1./8000.

#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  read v_1 & transform v_1 to a 3D array (file 1)
# V1 Velocity
uvw = sio.loadmat('u1_sdes.mat')
ttu=uvw['u1_sdes']
u3d1= np.reshape(ttu,(nk,nj,ni))

uvw = sio.loadmat('u2_sdes.mat')
ttu=uvw['u2_sdes']
u3d2= np.reshape(ttu,(nk,nj,ni))

uvw = sio.loadmat('u3_sdes.mat')
ttu=uvw['u3_sdes']
u3d3= np.reshape(ttu,(nk,nj,ni))

uvw = sio.loadmat('u4_sdes.mat')
ttu=uvw['u4_sdes']
u3d4= np.reshape(ttu,(nk,nj,ni))

uvw = sio.loadmat('u5_sdes.mat')
ttu=uvw['u5_sdes']
u3d5= np.reshape(ttu,(nk,nj,ni))

uvw = sio.loadmat('u6_sdes.mat')
ttu=uvw['u6_sdes']
u3d6= np.reshape(ttu,(nk,nj,ni))

uvw = sio.loadmat('u7_sdes.mat')
ttv=uvw['u7_sdes']
u3d7= np.reshape(ttv,(nk,nj,ni))

uvw = sio.loadmat('u8_sdes.mat')
ttv=uvw['u8_sdes']
u3d8= np.reshape(ttv,(nk,nj,ni))

# V2 Velocity
uvw = sio.loadmat('v1_sdes.mat')
ttv=uvw['v1_sdes']
v3d1= np.reshape(ttv,(nk,nj,ni))

uvw = sio.loadmat('v2_sdes.mat')
ttv=uvw['v2_sdes']
v3d2= np.reshape(ttv,(nk,nj,ni))

uvw = sio.loadmat('v3_sdes.mat')
ttv=uvw['v3_sdes']
v3d3= np.reshape(ttv,(nk,nj,ni))
```

```
65
66  uvw = sio.loadmat('v4_sdes.mat')
67  ttv=uvw['v4_sdes']
68  v3d4= np.reshape(ttv,(nk,nj,ni))
69
70  uvw = sio.loadmat('v5_sdes.mat')
71  ttv=uvw['v5_sdes']
72  v3d5= np.reshape(ttv,(nk,nj,ni))
73
74  uvw = sio.loadmat('v6_sdes.mat')
75  ttv=uvw['v6_sdes']
76  v3d6= np.reshape(ttv,(nk,nj,ni))
77
78  uvw = sio.loadmat('v7_sdes.mat')
79  ttv=uvw['v7_sdes']
80  v3d7= np.reshape(ttv,(nk,nj,ni))
81
82  uvw = sio.loadmat('v8_sdes.mat')
83  ttv=uvw['v8_sdes']
84  v3d8= np.reshape(ttv,(nk,nj,ni))
85
86  # V3 Velocity
87  uvw = sio.loadmat('w1_sdes.mat')
88  ttw=uvw['w1_sdes']
89  w3d1= np.reshape(ttw,(nk,nj,ni))
90
91  uvw = sio.loadmat('w2_sdes.mat')
92  ttw=uvw['w2_sdes']
93  w3d2= np.reshape(ttw,(nk,nj,ni))
94
95  uvw = sio.loadmat('w3_sdes.mat')
96  ttw=uvw['w3_sdes']
97  w3d3= np.reshape(ttw,(nk,nj,ni))
98
99  uvw = sio.loadmat('w4_sdes.mat')
100 ttw=uvw['w4_sdes']
101 w3d4= np.reshape(ttw,(nk,nj,ni))
102
103 uvw = sio.loadmat('w5_sdes.mat')
104 ttw=uvw['w5_sdes']
105 w3d5= np.reshape(ttw,(nk,nj,ni))
106
107 uvw = sio.loadmat('w6_sdes.mat')
108 ttw=uvw['w6_sdes']
109 w3d6= np.reshape(ttw,(nk,nj,ni))
110
111 uvw = sio.loadmat('w7_sdes.mat')
112 ttw=uvw['w7_sdes']
113 w3d7= np.reshape(ttw,(nk,nj,ni))
114
115 uvw = sio.loadmat('w8_sdes.mat')
116 ttw=uvw['w8_sdes']
117 w3d8= np.reshape(ttw,(nk,nj,ni))
118
119 # Turbulent Kinetic Energy
120 uvw = sio.loadmat('te1_sdes.mat')
121 ttke=uvw['te1_sdes']
122 te3d1= np.reshape(ttke,(nk,nj,ni))
123
124 uvw = sio.loadmat('te2_sdes.mat')
125 ttke=uvw['te2_sdes']
126 te3d2= np.reshape(ttke,(nk,nj,ni))
127
128 uvw = sio.loadmat('te3_sdes.mat')
129 ttke=uvw['te3_sdes']
130 te3d3= np.reshape(ttke,(nk,nj,ni))
```

```
131
132 uvw = sio.loadmat('te4_sdes.mat')
133 ttke=uvw['te4_sdes']
134 te3d4= np.reshape(ttke,(nk,nj,ni))
135
136 uvw = sio.loadmat('te5_sdes.mat')
137 ttke=uvw['te5_sdes']
138 te3d5= np.reshape(ttke,(nk,nj,ni))
139
140 uvw = sio.loadmat('te6_sdes.mat')
141 ttke=uvw['te6_sdes']
142 te3d6= np.reshape(ttke,(nk,nj,ni))
143
144 uvw = sio.loadmat('te7_sdes.mat')
145 ttke=uvw['te7_sdes']
146 te3d7= np.reshape(ttke,(nk,nj,ni))
147
148 uvw = sio.loadmat('te8_sdes.mat')
149 ttke=uvw['te8_sdes']
150 te3d8= np.reshape(ttke,(nk,nj,ni))
151
152 # Specific Dissipation
153 uvw = sio.loadmat('om1_sdes.mat')
154 ttom=uvw['om1_sdes']
155 om3d1= np.reshape(ttom,(nk,nj,ni))
156
157 uvw = sio.loadmat('om2_sdes.mat')
158 ttom=uvw['om2_sdes']
159 om3d2= np.reshape(ttom,(nk,nj,ni))
160
161 uvw = sio.loadmat('om3_sdes.mat')
162 ttom=uvw['om3_sdes']
163 om3d3= np.reshape(ttom,(nk,nj,ni))
164
165 uvw = sio.loadmat('om4_sdes.mat')
166 ttom=uvw['om4_sdes']
167 om3d4= np.reshape(ttom,(nk,nj,ni))
168
169 uvw = sio.loadmat('om5_sdes.mat')
170 ttom=uvw['om5_sdes']
171 om3d5= np.reshape(ttom,(nk,nj,ni))
172
173 uvw = sio.loadmat('om6_sdes.mat')
174 ttom=uvw['om6_sdes']
175 om3d6= np.reshape(ttom,(nk,nj,ni))
176
177 uvw = sio.loadmat('om7_sdes.mat')
178 ttom=uvw['om7_sdes']
179 om3d7= np.reshape(ttom,(nk,nj,ni))
180
181 uvw = sio.loadmat('om8_sdes.mat')
182 ttom=uvw['om8_sdes']
183 om3d8= np.reshape(ttom,(nk,nj,ni))
184
185 # merge 2 files. This means that new ni = 2*ni
186 u3d=np.concatenate((u3d1,u3d2,u3d3,u3d4,u3d5,u3d6,u3d7,u3d8), axis=0)
187 v3d=np.concatenate((v3d1,v3d2,v3d3,v3d4,v3d5,v3d6,v3d7,v3d8), axis=0)
188 w3d=np.concatenate((w3d1,w3d2,w3d3,w3d4,w3d5,w3d6,w3d7,w3d8), axis=0)
189 te3d=np.concatenate((te3d1,te3d2,te3d3,te3d4,te3d5,te3d6,te3d7,te3d8), axis=0)
190 om3d=np.concatenate((om3d1,om3d2,om3d3,om3d4,om3d5,om3d6,om3d7,om3d8), axis=0)
191
192 # x coordinate direction = index 0, first index
193 # y coordinate direction = index 1, second index
194 # z coordinate direction = index 2, third index
195
196 ni=len(u3d)
```

```

x=dx*ni
z=dz*nk

umean=np.mean(u3d, axis=(0,2))
vmean=np.mean(v3d, axis=(0,2))
wmean=np.mean(w3d, axis=(0,2))
temean=np.mean(te3d, axis=(0,2))
ommean=np.mean(om3d, axis=(0,2))

# face coordinates
yc = np.loadtxt("yc.dat")

# cell cener coordinates
y= np.zeros(nj)
dy=np.zeros(nj)
for j in range (1,nj-1):
# dy = cell width
   dy[j]=yc[j]-yc[j-1]
   y[j]=0.5*(yc[j]+yc[j-1])

y[nj-1]=yc[nj-1]
tauw=viscos*umean[1]/y[1]
ustar=tauw**0.5
yplus=y*ustar/viscos

DNS=np.genfromtxt("LM_Channel_5200_mean_prof.dat", dtype=None,comments="%")
y_DNS=DNS[:,0]
yplus_DNS=DNS[:,1]
u_DNS=DNS[:,2]

DNS=np.genfromtxt("LM_Channel_5200_vel_fluc_prof.dat", dtype=None,comments="%")
u2_DNS=DNS[:,2]
v2_DNS=DNS[:,3]
w2_DNS=DNS[:,4]
uv_DNS=DNS[:,5]

k_DNS=0.5*(u2_DNS+v2_DNS+w2_DNS)

# find equi.distant DNS cells in log-scale
xx=0.
jDNS=[1]*40
for i in range (0,40):
   i1 = (np.abs(10.**xx-yplus_DNS)).argmin()
   jDNS[i]=int(i1)
   xx=xx+0.2

# %% Assignment S2 - Mean Velocity Profile
fig1,axes = plt.subplots(1,1,constrained_layout=True)
axes.semilogx(y,umean/ustar,'b--',label='DES Data')
axes.semilogx(y_DNS[jDNS],u_DNS[jDNS],'r-o',label='DNS Data')
axes.semilogx(0.0175*np.ones(20),np.linspace(0,25,20),'k--',label=r'$y/\delta$ =
    0.0175')
axes.set_xlabel(r"$y/\delta$")
axes.set_ylabel(r"$U^+$")
axes.legend(fontsize=16,loc='best')
axes.tick_params(labelsize='medium')
axes.grid(True, linestyle='-.')
fig1.suptitle(r'Time Averaged Velocity ($<\overline{{v_{1}}^{\prime}}>$)',fontsize=22)

# %% Assignment S3 - Resolved Turbulent Shear Stress
uv_mean = np.mean((u3d-umean[None,:,None])*(v3d-vmean[None,:,None]),axis=(0,2))

fig2,axes = plt.subplots(1,1,constrained_layout=True)
axes.semilogx(y,uv_mean,'b--',label='DES Data - Resolved Turbulent Shear Stress')
axes.semilogx(y_DNS[jDNS],uv_DNS[jDNS],'r-o',label='DNS Data - Total Turbulent Shear
```

```
    Stress')
axes.semilogx(0.0175*np.ones(20),np.linspace(-1,0,20),'k--',label=r'$y/\delta$ =
    0.0175')
axes.set_xlabel(r"$y/\delta$")
axes.set_ylabel(r'Shear Stress $<{\overline{{v_{1}}^{\prime}}}{\overline{{v_{2}}^{\prime
    }}}>$')
axes.legend(fontsize=16,loc='best')
axes.tick_params(labelsize='medium')
axes.grid(True, linestyle='-.')
fig2.suptitle(r'Resolved Turbulent Shear Stress',fontsize=22)

# %% Assignment S4 - Turbulent Kinetic Energy
uu_mean = np.mean((u3d-umean[None,:,None])**2,axis=(0,2))
vv_mean = np.mean((v3d-vmean[None,:,None])**2,axis=(0,2))
ww_mean = np.mean((w3d-wmean[None,:,None])**2,axis=(0,2))

#uu_mean = np.mean(u3d*u3d,axis=(0,2)) - (umean*umean)
#vv_mean = np.mean(v3d*v3d,axis=(0,2)) - (vmean*vmean)
#ww_mean = np.mean(w3d*w3d,axis=(0,2)) - (wmean*wmean)

ke_res = 0.5*(uu_mean+vv_mean+ww_mean)
ke_res_DNS =  0.5*(u2_DNS+v2_DNS+w2_DNS)

fig3,axes = plt.subplots(1,1,constrained_layout=True)
axes.semilogx(y,temean,'b--',label='DES Data - Modeled Turbulent Kinetic Energy')
axes.semilogx(y,ke_res,'r-o',label='DES Data - Resolved Turbulent Kinetic Energy')
axes.semilogx(y,ke_res+temean,'k-o',label='DES Data - Total Turbulent Kinetic Energy')
axes.semilogx(y_DNS,ke_res_DNS,'g-o',label='DNS Data - Turbulent Kinetic Energy')
axes.semilogx(0.0175*np.ones(10),np.linspace(0,6,10),'k--',label=r'$y/\delta$ = 0.0175')
axes.set_xlabel(r"$y/\delta$")
axes.set_ylabel(r'TKE')
axes.legend(fontsize=16,loc='best')
axes.tick_params(labelsize='medium')
axes.grid(True, linestyle='-.')
fig3.suptitle(r'Turbulent Kinetic Energy',fontsize=22)

# %% Assignment S5 - Modeled Turbulent Shear Stress
turb_visc = te3d/om3d
dudx,dudy,dudz = np.gradient(u3d,dx,y,dz)
dvdx,dvdy,dvdz = np.gradient(v3d,dx,y,dz)
dwdx,dwdy,dwdz = np.gradient(w3d,dx,y,dz)
tau_12 = -turb_visc * (dudy+dvdx)
tau_12_mean = np.mean(tau_12,axis=(0,2))
tau_12_mean[0] = 0

fig4,axes = plt.subplots(1,1,constrained_layout=True)
axes.semilogx(y,uv_mean,'b--',label='DES Data - Resolved Turbulent Shear Stress')
axes.semilogx(y,tau_12_mean,'g--',label='DES Data - Modeled Turbulent Shear Stress')
axes.semilogx(y_DNS[jDNS],uv_DNS[jDNS],'r-o',label='DNS Data - Total Turbulent Shear
    Stress')
axes.semilogx(0.0175*np.ones(20),np.linspace(-1,0,20),'k--',label=r'$y/\delta$ =
    0.0175')
axes.set_xlabel(r"$y/\delta$")
axes.set_ylabel(r'Shear Stress $<{\overline{{v_{1}}^{\prime}}}{\overline{{v_{2}}^{\prime
    }}}>$')
axes.legend(fontsize=16,loc='best')
axes.tick_params(labelsize='medium')
axes.grid(True, linestyle='-.')
fig4.suptitle(r'Comparision of Shear Stress',fontsize=22)

fig5,axes = plt.subplots(1,1,constrained_layout=True)
axes.semilogx(y,uv_mean+tau_12_mean,'b-o',label='DES Data')
axes.semilogx(y_DNS[jDNS],uv_DNS[jDNS],'r-o',label='DNS Data')
axes.semilogx(0.0175*np.ones(20),np.linspace(-1,0,20),'k--',label=r'$y/\delta$ =
    0.0175')
axes.set_xlabel(r"$y/\delta$")
```

```
axes.set_ylabel(r'Shear Stress $<{\overline{{v_{1}}^{\prime}}}{\overline{{v_{2}}^{\prime}}}>$')
axes.legend(fontsize=16,loc='best')
axes.tick_params(labelsize='medium')
axes.grid(True, linestyle='-.')
fig5.suptitle(r'Total Turbulent Shear Stress',fontsize=22)

# %% Assignment S6 - Location of Interface in DES and DDES
C_DES_SA = 0.65
delta_SA = C_DES_SA * np.maximum(dx*np.ones([49,1]),dy.reshape(49,1),dz*np.ones([49,1]))

Int_SA = y[(y.reshape(49,1)>=delta_SA).argmax()]
print('Switch for SA-DES model occurs at y+ = '+str(np.around(Int_SA,3)))

C_DES_SST = 0.61
beta_star = 0.09
delta_SST = C_DES_SST * np.maximum(dx*np.ones([49,1]),dy.reshape(49,1),dz*np.ones([49,1]))
Lt = ((temean**0.5)/ommean)/beta_star
term1 = Lt.reshape(49,1)/(delta_SST)
term1[0] = 1e-10

Int_SST = y[(np.ones([49,1])>=term1).argmin()]
print('Switch for SST-DES model occurs at y+ = '+str(np.around(Int_SST,3)))

term2 = 2*(Lt[1:]/y[1:])
term3 = (500*viscos)/(ommean[1:]*(y[1:]**2))
eta = np.insert(np.maximum(term2,term3),0,[1e-10])
F2 = np.tanh(eta**2)
F_DDES = np.maximum(term1*(1-F2.reshape(49,1)),1)
Int_DDES = y[(np.ones([49,1])>=(term1*(1-F2.reshape(49,1)))).argmin()]
print('Switch for DDES model occurs at y+ = '+str(np.around(Int_DDES,3)))

fig6,axes = plt.subplots(1,1,constrained_layout=True)
axes.plot(umean,y,label=r'DES data : Mean velocity profile $<\overline{u}>$')
axes.plot(u_DNS,y_DNS,label=r'DNS data : Mean velocity profile $<\overline{u}>$')
axes.plot(np.linspace(0,30,20),0.0175*np.ones(20),'k--',lw=2,label=r'S-DES model')
axes.plot(np.linspace(0,30,20),Int_SA*np.ones(20),'b--',lw=2,label=r'SA-DES model')
axes.plot(np.linspace(0,30,20),Int_SST*np.ones(20),'y--',lw=2,label=r'SST-DES model')
axes.plot(np.linspace(0,30,20),Int_DDES*np.ones(20),'r--',lw=2,label=r'DDES model')
axes.set_ylabel(r"$y/\delta$")
axes.legend(fontsize=16,loc='best')
axes.tick_params(labelsize='medium')
axes.grid(True, linestyle='-.')
fig6.suptitle(r'Location of Interfaces for various models',fontsize=22)

# %% Assignment S7 - SAS Turbulent Length Scale
# 1D Length Scale
dua_dy = np.gradient(umean,y)
dua2_dy = np.gradient(dua_dy,y)
DNSdua_dy = np.gradient(u_DNS,y_DNS)
DNSdua2_dy = np.gradient(DNSdua_dy,y_DNS)
# Von-karman constant
kappa = 0.4

LVK_1D = kappa * np.abs(dua_dy/dua2_dy)
DNSLVK_1D = kappa * np.abs(DNSdua_dy/DNSdua2_dy)

# 3D Length Scale
du2_dxdx,du2_dxdy,du2_dxdz = np.gradient(dudx,dx,y,dz)
dv2_dxdx,dv2_dxdy,dv2_dxdz = np.gradient(dvdx,dx,y,dz)
dw2_dxdx,dw2_dxdy,dw2_dxdz = np.gradient(dwdx,dx,y,dz)

du2_dydx,du2_dydy,du2_dydz = np.gradient(dudy,dx,y,dz)
dv2_dydx,dv2_dydy,dv2_dydz = np.gradient(dvdy,dx,y,dz)
dw2_dydx,dw2_dydy,dw2_dydz = np.gradient(dwdy,dx,y,dz)
```

```
385
386 du2_dzdx,du2_dzdy,du2_dzdz = np.gradient(dudz,dx,y,dz)
387 dv2_dzdx,dv2_dzdy,dv2_dzdz = np.gradient(dvdz,dx,y,dz)
388 dw2_dzdx,dw2_dzdy,dw2_dzdz = np.gradient(dwdz,dx,y,dz)
389
390 S11 = 0.5*(2*dudx)
391 S12 = 0.5*(dudy+dvdx)
392 S13 = 0.5*(dudz+dwdx)
393 S22 = 0.5*(2*dvdy)
394 S23 = 0.5*(dvdz+dwdy)
395 S33 = 0.5*(2*dwdz)
396
397 S = np.sqrt(2*((S11**2)+(S22**2)+(S33**2)+(2*(S12**2))+(2*(S13**2))+(2*(S23**2))))
398
399 U1 = np.sqrt((du2_dxdx*du2_dxdx)+(du2_dydy*du2_dydy)+(du2_dzdz*du2_dzdz)+
400              (dv2_dxdx*dv2_dxdx)+(dv2_dydy*dv2_dydy)+(dv2_dzdz*dv2_dzdz)+
401              (dw2_dxdx*dw2_dxdx)+(dw2_dydy*dw2_dydy)+(dw2_dzdz*dw2_dzdz))
402 U2 = np.sqrt((du2_dxdx**2)+(du2_dxdy**2)+(du2_dxdz**2)+(du2_dydx**2)+(du2_dydy**2)+
403              (du2_dydz**2)+(du2_dzdx**2)+(du2_dzdy**2)+(du2_dzdz**2)+
404              (dv2_dxdx**2)+(dv2_dxdy**2)+(dv2_dxdz**2)+(dv2_dydx**2)+(dv2_dydy**2)+
405              (dv2_dydz**2)+(dv2_dzdx**2)+(dv2_dzdy**2)+(dv2_dzdz**2)+
406              (dw2_dxdx**2)+(dw2_dxdy**2)+(dw2_dxdz**2)+(dw2_dydx**2)+(dw2_dydy**2)+
407              (dw2_dydz**2)+(dw2_dzdx**2)+(dw2_dzdy**2)+(dw2_dzdz**2))
408
409 LVK_3D = kappa*np.abs(S/U1)
410 LVK_3D_2 = kappa*np.abs(S/U2)
411 delta = (dx*dy*dx)**(1/3)
412
413 fig7,axes = plt.subplots(1,1,constrained_layout=True)
414 axes.plot(y[:],LVK_1D[:],'k--',label=r'Von-Karman Length Scale 1D')
415 axes.plot(y[:],np.mean(LVK_3D[:],axis=(0,2))[:],'r--',label=r'Von-Karman Length Scale 3D
    ')
416 axes.plot(y[:],np.mean(LVK_3D_2[:],axis=(0,2))[:],'g--',label=r'Von-Karman Length Scale
    3D - Method 2')
417 #axes.plot(y[:],delta[:])
418 #axes.plot(y[:],dy[:])
419 axes.set_ylabel(r"Length Scale")
420 axes.set_xlabel(r"$y/\delta$")
421 axes.legend(fontsize=16,loc='best')
422 axes.tick_params(labelsize='medium')
423 axes.grid(True, linestyle='-.')
424 fig7.suptitle(r'Von-Karman Length Scale',fontsize=22)
425
426 fig7a = plt.figure()
427 axes1 = fig7a.add_axes([0.1, 0.1, 0.8, 0.8]) # main axes
428 axes2 = fig7a.add_axes([0.2, 0.55, 0.3, 0.3]) # inset axes
429 # Larger Figure Axes 1
430 axes1.plot(y[:],LVK_1D[:],'k-o',label=r'DES data - Von-Karman Length Scale 1D')
431 axes1.plot(y_DNS[4:],DNSLVK_1D[4:],'r-',label=r'DNS data - Von-Karman Length Scale 1D')
432 axes1.set_ylabel(r"Length Scale")
433 axes1.set_xlabel(r"$y/\delta$")
434 axes1.legend(fontsize=16,loc='upper right')
435 axes1.tick_params(labelsize='medium')
436 axes1.grid(True, linestyle='-.')
437 axes1.set_title(r'Von-Karman Length Scale',fontsize=22)
438 # Insert Figure Axes 2
439 axes2.plot(y[:],LVK_1D[:],'k-o',label=r'DES data - Von-Karman Length Scale 1D')
440 axes2.plot(y_DNS[4:],DNSLVK_1D[4:],'r-',label=r'DNS data - Von-Karman Length Scale 1D')
441 axes2.set_ylabel(r"Length Scale",fontsize=12)
442 axes2.set_xlabel(r"$y/\delta$",fontsize=12)
443 axes2.tick_params(labelsize='x-small')
444 axes2.grid(True, linestyle='-.')
445 axes2.set_ylim([-0.01,0.05])
446 axes2.set_xlim([-0.1,0.2])
447 axes2.set_title(r'Zoomed near the wall',fontsize=12)
448
```

```
# %% Assignment S8 - Estimate of Resolution
ustar_x = -((dx**2)/4) * (du2_dxdx + dv2_dxdx + dw2_dxdx)
vstar_x = -((dy[None,:,None]**2)/4) * (du2_dxdx + dv2_dxdx + dw2_dxdx)
wstar_x = -((dz**2)/4) * (du2_dxdx + dv2_dxdx + dw2_dxdx)

ustar_y = -((dx**2)/4) * (du2_dydy + dv2_dydy + dw2_dydy)
vstar_y = -((dy[None,:,None]**2)/4) * (du2_dydy + dv2_dydy + dw2_dydy)
wstar_y = -((dz**2)/4) * (du2_dydy + dv2_dydy + dw2_dydy)

ustar_z = -((dx**2)/4) * (du2_dzdz + dv2_dzdz + dw2_dzdz)
vstar_z = -((dy[None,:,None]**2)/4) * (du2_dzdz + dv2_dzdz + dw2_dzdz)
wstar_z = -((dz**2)/4) * (du2_dzdz + dv2_dzdz + dw2_dzdz)

Error_x = (dx*dy*dz) * np.sqrt((np.mean(vstar_x**2,axis=(0,2)))+(np.mean(ustar_x**2,axis
    =(0,2)))+(np.mean(wstar_x**2,axis=(0,2))))
Error_y = (dx*dy*dz) * np.sqrt((np.mean(vstar_y**2,axis=(0,2)))+(np.mean(ustar_y**2,axis
    =(0,2)))+(np.mean(wstar_y**2,axis=(0,2))))
Error_z = (dx*dy*dz) * np.sqrt((np.mean(vstar_z**2,axis=(0,2)))+(np.mean(ustar_z**2,axis
    =(0,2)))+(np.mean(wstar_z**2,axis=(0,2))))

fig8,axes = plt.subplots(1,1,constrained_layout=True)
axes.semilogx(y,Error_x,'r-o',label=r'Anisotropic Error in X-direction')
axes.semilogx(y,Error_y,'k-s',label=r'Anisotropic Error in Y-direction')
axes.semilogx(y,Error_z,'g-*',label=r'Anisotropic Error in Z-direction')
axes.set_xlabel(r"$y/\delta$")
axes.set_ylabel(r"Error")
axes.legend(fontsize=16,loc='best')
axes.tick_params(labelsize='medium')
axes.grid(True, linestyle='-.')
fig8.suptitle(r'Anisotropic Error along Cordinate Directions',fontsize=22)
```

For Assignment 2b : Section 2.1 to 2.3

```
import scipy.io as sio
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1.inset_locator import inset_axes
from dphidx_dy import dphidx_dy
from matplotlib.ticker import (MultipleLocator, FormatStrFormatter,
                               AutoMinorLocator)
plt.rcParams.update({'font.size': 22})

re =9.36e+5
viscos =1/re

xy_hump_fine = np.loadtxt("xy_hump.dat")
x=xy_hump_fine[:,0]
y=xy_hump_fine[:,1]

ni=314
nj=122

nim1=ni-1
njm1=nj-1

# Read Data File (from paper)
vectz=np.genfromtxt("vectz_aiaa_paper.dat",comments="%")
ntstep=vectz[0]
n=len(vectz)

# Read Data File (from journal)
vectz_aiaa = np.genfromtxt("vectz_aiaa_journal.dat",comments="%")

#             write(48,*)uvec(i,j)
#             write(48,*)vvec(i,j)
#             write(48,*)dummy(i,j)
#             write(48,*)uvec2(i,j)
#             write(48,*)vvec2(i,j)
#             write(48,*)wvec2(i,j)
#             write(48,*)uvvec(i,j)
#             write(48,*)p2D(i,j)
#             write(48,*)rk2D(i,j)
#             write(48,*)vis2D(i,j)
#             write(48,*)dissp2D(i,j)
#             write(48,*)uvturb(i,j)



nn=12
nst=0
iu=range(nst+1,n,nn)
iv=range(nst+2,n,nn)
ifk=range(nst+3,n,nn)
iuu=range(nst+4,n,nn)
ivv=range(nst+5,n,nn)
iww=range(nst+6,n,nn)
iuv=range(nst+7,n,nn)
ip=range(nst+8,n,nn)
ik=range(nst+9,n,nn)
ivis=range(nst+10,n,nn)
idiss=range(nst+11,n,nn)
iuv_model=range(nst+12,n,nn)

# From paper
u=vectz[iu]/ntstep
v=vectz[iv]/ntstep
fk=vectz[ifk]/ntstep
```

```
uu=vectz[iuu]/ntstep
vv=vectz[ivv]/ntstep
ww=vectz[iww]/ntstep
uv=vectz[iuv]/ntstep
p=vectz[ip]/ntstep
k_model=vectz[ik]/ntstep
vis=vectz[ivis]/ntstep
diss=vectz[idiss]/ntstep
uv_model=vectz[iuv_model]/ntstep
# From journal
u1=vectz_aiaa[iu]/ntstep
v1=vectz_aiaa[iv]/ntstep
fk1=vectz_aiaa[ifk]/ntstep
uu1=vectz_aiaa[iuu]/ntstep
vv1=vectz_aiaa[ivv]/ntstep
ww1=vectz_aiaa[iww]/ntstep
uv1=vectz_aiaa[iuv]/ntstep
p1=vectz_aiaa[ip]/ntstep
k_model1=vectz_aiaa[ik]/ntstep
vis1=vectz_aiaa[ivis]/ntstep
diss1=vectz_aiaa[idiss]/ntstep
uv1_model=vectz_aiaa[iuv_model]/ntstep


# uu is total inst. velocity squared. Hence the resolved turbulent resolved stresses are
    obtained as
# From paper
uu=uu-u**2
vv=vv-v**2
uv=uv-u*v
# From journal
uu1=uu1-u**2
vv1=vv1-v**2
uv1=uv1-u*v

# From paper
p_2d=np.reshape(p,(ni,nj))
u_2d=np.reshape(u,(ni,nj))
v_2d=np.reshape(v,(ni,nj))
fk_2d=np.reshape(fk,(ni,nj))
uu_2d=np.reshape(uu,(ni,nj))
uv_2d=np.reshape(uv,(ni,nj))
vv_2d=np.reshape(vv,(ni,nj))
ww_2d=np.reshape(ww,(ni,nj))
uv_model_2d=np.reshape(uv_model,(ni,nj))
k_model_2d=np.reshape(k_model,(ni,nj))
vis_2d=np.reshape(vis,(ni,nj)) #this is to total viscosity, i.e. vis_tot=vis+vis_turb
diss_2d=np.reshape(diss,(ni,nj)) #this is to total viscosity, i.e. vis_tot=vis+vis_turb
x_2d=np.transpose(np.reshape(x,(nj,ni)))
y_2d=np.transpose(np.reshape(y,(nj,ni)))
# From journal
p1_2d=np.reshape(p1,(ni,nj))
u1_2d=np.reshape(u1,(ni,nj))
v1_2d=np.reshape(v1,(ni,nj))
fk1_2d=np.reshape(fk1,(ni,nj))
uu1_2d=np.reshape(uu1,(ni,nj))
uv1_2d=np.reshape(uv1,(ni,nj))
vv1_2d=np.reshape(vv1,(ni,nj))
ww1_2d=np.reshape(ww1,(ni,nj))
uv1_model_2d=np.reshape(uv1_model,(ni,nj))
k_model1_2d=np.reshape(k_model1,(ni,nj))
vis1_2d=np.reshape(vis1,(ni,nj)) #this is to total viscosity, i.e. vis_tot=vis+vis_turb
diss1_2d=np.reshape(diss1,(ni,nj)) #this is to total viscosity, i.e. vis_tot=vis+
    vis_turb

# set fk_2d=1 at upper boundary
```

```
fk_2d[:,nj-1]=fk_2d[:,nj-2]
fk1_2d[:,nj-1]=fk_2d[:,nj-2]

x065_off=np.genfromtxt("x065_off.dat",comments="%")
x066_off=np.genfromtxt("x066_off.dat",comments="%")
x080_off=np.genfromtxt("x080_off.dat",comments="%")
x090_off=np.genfromtxt("x090_off.dat",comments="%")
x100_off=np.genfromtxt("x100_off.dat",comments="%")
x110_off=np.genfromtxt("x110_off.dat",comments="%")
x120_off=np.genfromtxt("x120_off.dat",comments="%")
x130_off=np.genfromtxt("x130_off.dat",comments="%")

# face cordinates for dphidx_dy function
#xf2d = np.zeros((ni,nj))
#yf2d = np.zeros((ni,nj))
xf2d = 0.5 * (x_2d[1:-2,:] + x_2d[2:-1,:])
yf2d = 0.5 * (y_2d[:,1:-2] + y_2d[:,2:-1])
#xf2d[] = x_2d[]
#yf2d[] = y_2d[]

# the funtion dphidx_dy wants x and y arrays to be one cell smaller than u2d. Hence I
    take away the last row and column below
x_2d_new=np.delete(x_2d,-1,0)
x_2d_new=np.delete(x_2d_new,-1,1)
y_2d_new=np.delete(y_2d,-1,0)
y_2d_new=np.delete(y_2d_new,-1,1)
# compute the gradient
dudx,dudy=dphidx_dy(x_2d_new,y_2d_new,u_2d)
dvdx,dvdy=dphidx_dy(x_2d_new,y_2d_new,v_2d)

#*************************
# Mesh for visualization
a = np.transpose(x_2d)
b = np.transpose(y_2d)
fig16 = plt.figure()
axes = fig16.add_axes([0.1,0.1,0.8,0.8])
axes.plot(x_2d,y_2d,color='black')
axes.plot(a,b,color='black')
axes.tick_params(axis="x", labelsize=15)
axes.tick_params(axis="y", labelsize=15)
axes.set_xlabel('x [m]',fontsize=20)
axes.set_ylabel('y [m]',fontsize=20)
axes.set_title('Computational mesh in the domain',fontsize=24)

#*************************
# plot u
fig1 = plt.figure("Figure 1")
plt.clf() #clear the figure
xx=0.65;
i1 = (np.abs(xx-x_2d[:,1])).argmin()  # find index which closest fits xx
plt.plot(u_2d[i1,:],y_2d[i1,:]-y_2d[i1,0],'b-')
plt.plot(x065_off[:,2],x065_off[:,1]-y_2d[i1,0],'bo')
plt.xlabel("$U$")
plt.ylabel("$y-y_{wall}$")
plt.title("$x=0.65$")
plt.axis([0, 1.3,0,0.2])
#plt.savefig('u065_hump_python.eps')

#*************************
# plot vv
fig2 = plt.figure("Figure 2")
plt.clf() #clear the figure
xx=0.65;
i1 = (np.abs(xx-x_2d[:,1])).argmin()  # find index which closest fits xx
plt.plot(vv_2d[i1,:],y_2d[i1,:]-y_2d[i1,0],'b-')
plt.plot(x065_off[:,5],x065_off[:,1]-y_2d[i1,0],'bo')
```

```
plt.xlabel("$\overline{v'v'}$")
plt.ylabel("$y-y_{wall}$")
plt.title("$x=0.65$")
plt.axis([0, 0.01,0,0.2])
#plt.savefig('vv065_hump_python.eps')

#************************
# plot uu
fig3 = plt.figure("Figure 3")
plt.clf() #clear the figure
xx=0.65;
i1 = (np.abs(xx-x_2d[:,1])).argmin()  # find index which closest fits xx
plt.plot(uu_2d[i1,:],y_2d[i1,:]-y_2d[i1,0],'b-')
plt.plot(x065_off[:,4],x065_off[:,1]-y_2d[i1,0],'bo')
plt.xlabel("$\overline{u'u'}$")
plt.ylabel("$y-y_{wall}$")
plt.title("$x=0.65$")
plt.axis([0, 0.01,0,0.2])
#plt.savefig('uu065_hump_python.eps')

################################ contoue plot
fig4 = plt.figure("Figure 4")
plt.clf() #clear the figure
plt.contourf(x_2d,y_2d,uu_2d, 50)
plt.xlabel("$x$")
plt.ylabel("$y$")
plt.clim(0,0.05)
plt.axis([0.6,1.5,0,1])
plt.title("contour $\overline{u'u'}$")
#plt.savefig('piso_python.eps')

################################ vector plot
fig5 = plt.figure("Figure 5")
plt.clf() #clear the figure
k=6# plot every forth vector
ss=3.2 #vector length
plt.quiver(x_2d[::k,::k],y_2d[::k,::k],u_2d[::k,::k],v_2d[::k,::k],width=0.01)
plt.xlabel("$x$")
plt.ylabel("$y$")
plt.axis([0.6,1.5,0,1])
plt.title("vector plot")
#plt.savefig('vect_python.eps')

# %% Assignment T1 - Discretization Schemes
# Paper Data [203]
## Plot vv
xx = np.array([0,0.65,0.8,0.9,1,1.1,1.2,1.3])
i1 = list(map(lambda m : (np.abs(m-x_2d[:,1])).argmin(),xx))
fig6,axes = plt.subplots(2,4,constrained_layout=True)
axes = axes.reshape(-1)
axes[0].plot(vv_2d[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0],'b-o',ms=4,label='x = 0 m')
axes[1].plot(vv_2d[i1[1],:],y_2d[i1[1],:]-y_2d[i1[1],0],'g-v',ms=4,label='x = 0.65 m')
axes[2].plot(vv_2d[i1[2],:],y_2d[i1[2],:]-y_2d[i1[2],0],'r-1',ms=4,label='x = 0.8 m')
axes[3].plot(vv_2d[i1[3],:],y_2d[i1[3],:]-y_2d[i1[3],0],'c-s',ms=4,label='x = 0.9 m')
axes[4].plot(vv_2d[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0],'m-+',ms=4,label='x = 1 m')
axes[5].plot(vv_2d[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0],'y-*',ms=4,label='x = 1.1 m')
axes[6].plot(vv_2d[i1[6],:],y_2d[i1[6],:]-y_2d[i1[6],0],'k-d',ms=4,label='x = 1.2 m')
axes[7].plot(vv_2d[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0],label='x = 1.3 m')
for ax in axes:
    ax.xaxis.set_minor_locator(AutoMinorLocator())
    ax.yaxis.set_minor_locator(AutoMinorLocator())
    ax.legend(fontsize=14,loc='best')
    ax.tick_params(labelsize='medium')
    ax.grid(True, linestyle='-.')
    ax.set_ylabel("$y-y_{wall}$")
    if (ax==axes[4] or ax==axes[5] or ax==axes[6] or ax==axes[7]):
```

```
        ax.set_xlabel("$\overline{v'v'}$")
fig6.suptitle("$<\overline{v'v'}>$ at various locations",fontsize=22)

## Plot uv
fig7,axes = plt.subplots(2,4,constrained_layout=True)
axes = axes.reshape(-1)
axes[0].plot(uv_2d[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0],'b-o',ms=4,label='x = 0 m')
axes[1].plot(uv_2d[i1[1],:],y_2d[i1[1],:]-y_2d[i1[1],0],'g-v',ms=4,label='x = 0.65 m')
axes[2].plot(uv_2d[i1[2],:],y_2d[i1[2],:]-y_2d[i1[2],0],'r-1',ms=4,label='x = 0.8 m')
axes[3].plot(uv_2d[i1[3],:],y_2d[i1[3],:]-y_2d[i1[3],0],'c-s',ms=4,label='x = 0.9 m')
axes[4].plot(uv_2d[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0],'m-+',ms=4,label='x = 1 m')
axes[5].plot(uv_2d[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0],'y-*',ms=4,label='x = 1.1 m')
axes[6].plot(uv_2d[i1[6],:],y_2d[i1[6],:]-y_2d[i1[6],0],'k-d',ms=4,label='x = 1.2 m')
axes[7].plot(uv_2d[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0],label='x = 1.3 m')
for ax in axes:
    ax.xaxis.set_minor_locator(AutoMinorLocator())
    ax.yaxis.set_minor_locator(AutoMinorLocator())
    ax.legend(fontsize=14,loc='best')
    ax.tick_params(labelsize='medium')
    ax.grid(True, linestyle='-.')
    ax.set_ylabel("$y-y_{wall}$")
    if (ax==axes[4] or ax==axes[5] or ax==axes[6] or ax==axes[7]):
        ax.set_xlabel("$\overline{u'v'}$")
fig7.suptitle("<$\overline{u'v'}$> at various locations",fontsize=22)

# Journal Data [169]
## Plot vv
fig8,axes = plt.subplots(2,4,constrained_layout=True)
axes = axes.reshape(-1)
axes[0].plot(vv1_2d[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0],'b-o',ms=4,label='x = 0 m')
axes[1].plot(vv1_2d[i1[1],:],y_2d[i1[1],:]-y_2d[i1[1],0],'g-v',ms=4,label='x = 0.65 m')
axes[2].plot(vv1_2d[i1[2],:],y_2d[i1[2],:]-y_2d[i1[2],0],'r-1',ms=4,label='x = 0.8 m')
axes[3].plot(vv1_2d[i1[3],:],y_2d[i1[3],:]-y_2d[i1[3],0],'c-s',ms=4,label='x = 0.9 m')
axes[4].plot(vv1_2d[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0],'m-+',ms=4,label='x = 1 m')
axes[5].plot(vv1_2d[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0],'y-*',ms=4,label='x = 1.1 m')
axes[6].plot(vv1_2d[i1[6],:],y_2d[i1[6],:]-y_2d[i1[6],0],'k-d',ms=4,label='x = 1.2 m')
axes[7].plot(vv1_2d[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0],label='x = 1.3 m')
for ax in axes:
    ax.xaxis.set_minor_locator(AutoMinorLocator())
    ax.yaxis.set_minor_locator(AutoMinorLocator())
    ax.legend(fontsize=14,loc='best')
    ax.tick_params(labelsize='medium')
    ax.grid(True, linestyle='-.')
    ax.set_ylabel("$y-y_{wall}$")
    if (ax==axes[4] or ax==axes[5] or ax==axes[6] or ax==axes[7]):
        ax.set_xlabel("$\overline{v'v'}$")
fig8.suptitle("$<\overline{v'v'}>$ at various locations",fontsize=22)

## Plot uv
fig9,axes = plt.subplots(2,4,constrained_layout=True)
axes = axes.reshape(-1)
axes[0].plot(uv1_2d[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0],'b-o',ms=4,label='x = 0 m')
axes[1].plot(uv1_2d[i1[1],:],y_2d[i1[1],:]-y_2d[i1[1],0],'g-v',ms=4,label='x = 0.65 m')
axes[2].plot(uv1_2d[i1[2],:],y_2d[i1[2],:]-y_2d[i1[2],0],'r-1',ms=4,label='x = 0.8 m')
axes[3].plot(uv1_2d[i1[3],:],y_2d[i1[3],:]-y_2d[i1[3],0],'c-s',ms=4,label='x = 0.9 m')
axes[4].plot(uv1_2d[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0],'m-+',ms=4,label='x = 1 m')
axes[5].plot(uv1_2d[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0],'y-*',ms=4,label='x = 1.1 m')
axes[6].plot(uv1_2d[i1[6],:],y_2d[i1[6],:]-y_2d[i1[6],0],'k-d',ms=4,label='x = 1.2 m')
axes[7].plot(uv1_2d[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0],label='x = 1.3 m')
for ax in axes:
    ax.xaxis.set_minor_locator(AutoMinorLocator())
    ax.yaxis.set_minor_locator(AutoMinorLocator())
    ax.legend(fontsize=14,loc='best')
    ax.tick_params(labelsize='medium')
    ax.grid(True, linestyle='-.')
    ax.set_ylabel("$y-y_{wall}$")
```

```
    if (ax==axes[4] or ax==axes[5] or ax==axes[6] or ax==axes[7]):
        ax.set_xlabel("$\overline{u'v'}$")
fig9.suptitle("<$\overline{v'v'}$> at various locations",fontsize=22)

# Comparision of both dataset
## Plot uu
fig10,axes = plt.subplots(2,4,constrained_layout=True)
axes = axes.reshape(-1)
axes[0].plot(uu_2d[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0],'r-o',ms=4,label='Paper Data [203]')
axes[1].plot(uu_2d[i1[1],:],y_2d[i1[1],:]-y_2d[i1[1],0],'r-o',ms=4,label='Paper Data [203]')
axes[2].plot(uu_2d[i1[2],:],y_2d[i1[2],:]-y_2d[i1[2],0],'r-o',ms=4,label='Paper Data [203]')
axes[3].plot(uu_2d[i1[3],:],y_2d[i1[3],:]-y_2d[i1[3],0],'r-o',ms=4,label='Paper Data [203]')
axes[4].plot(uu_2d[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0],'r-o',ms=4,label='Paper Data [203]')
axes[5].plot(uu_2d[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0],'r-o',ms=4,label='Paper Data [203]')
axes[6].plot(uu_2d[i1[6],:],y_2d[i1[6],:]-y_2d[i1[6],0],'r-o',ms=4,label='Paper Data [203]')
axes[7].plot(uu_2d[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0],'r-o',ms=4,label='Paper Data [203]')
axes[0].plot(uu1_2d[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0],'b-s',ms=4,label='Journal Data [169]')
axes[1].plot(uu1_2d[i1[1],:],y_2d[i1[1],:]-y_2d[i1[1],0],'b-s',ms=4,label='Journal Data [169]')
axes[2].plot(uu1_2d[i1[2],:],y_2d[i1[2],:]-y_2d[i1[2],0],'b-s',ms=4,label='Journal Data [169]')
axes[3].plot(uu1_2d[i1[3],:],y_2d[i1[3],:]-y_2d[i1[3],0],'b-s',ms=4,label='Journal Data [169]')
axes[4].plot(uu1_2d[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0],'b-s',ms=4,label='Journal Data [169]')
axes[5].plot(uu1_2d[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0],'b-s',ms=4,label='Journal Data [169]')
axes[6].plot(uu1_2d[i1[6],:],y_2d[i1[6],:]-y_2d[i1[6],0],'b-s',ms=4,label='Journal Data [169]')
axes[7].plot(uu1_2d[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0],'b-s',ms=4,label='Journal Data [169]')
axes[1].plot(x065_off[:,4],x065_off[:,1]-y_2d[i1[1],0],'k-*',ms=4,label='Experimental Data')
axes[2].plot(x080_off[:,4],x080_off[:,1]-y_2d[i1[2],0],'k-*',ms=4,label='Experimental Data')
axes[3].plot(x090_off[:,4],x090_off[:,1]-y_2d[i1[3],0],'k-*',ms=4,label='Experimental Data')
axes[4].plot(x100_off[:,4],x100_off[:,1]-y_2d[i1[4],0],'k-*',ms=4,label='Experimental Data')
axes[5].plot(x110_off[:,4],x110_off[:,1]-y_2d[i1[5],0],'k-*',ms=4,label='Experimental Data')
axes[6].plot(x120_off[:,4],x120_off[:,1]-y_2d[i1[6],0],'k-*',ms=4,label='Experimental Data')
axes[7].plot(x130_off[:,4],x130_off[:,1]-y_2d[i1[7],0],'k-*',ms=4,label='Experimental Data')
for ax,xloc in zip(axes,xx):
    ax.xaxis.set_minor_locator(AutoMinorLocator())
    ax.yaxis.set_minor_locator(AutoMinorLocator())
    ax.legend(title='At X = '+str(xloc)+' m',title_fontsize=14,fontsize=14,loc='best')
    ax.tick_params(labelsize='medium')
    ax.grid(True, linestyle='-.')
    ax.set_ylim([0,0.25])
    if (ax==axes[4] or ax==axes[5] or ax==axes[6] or ax==axes[7]):
        ax.set_xlabel("$<\overline{u'u'}>$")
    if (ax==axes[0] or ax==axes[4]):
        ax.set_ylabel("$y-y_{wall}$")
fig10.suptitle("$<\overline{u'u'}>$ at various locations",fontsize=22)
```

```
369
370 ## Plot vv
371 fig11,axes = plt.subplots(2,4,constrained_layout=True)
372 axes = axes.reshape(-1)
373 axes[0].plot(vv_2d[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0],'r-o',ms=4,label='Paper Data
    [203]')
374 axes[1].plot(vv_2d[i1[1],:],y_2d[i1[1],:]-y_2d[i1[1],0],'r-o',ms=4,label='Paper Data
    [203]')
375 axes[2].plot(vv_2d[i1[2],:],y_2d[i1[2],:]-y_2d[i1[2],0],'r-o',ms=4,label='Paper Data
    [203]')
376 axes[3].plot(vv_2d[i1[3],:],y_2d[i1[3],:]-y_2d[i1[3],0],'r-o',ms=4,label='Paper Data
    [203]')
377 axes[4].plot(vv_2d[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0],'r-o',ms=4,label='Paper Data
    [203]')
378 axes[5].plot(vv_2d[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0],'r-o',ms=4,label='Paper Data
    [203]')
379 axes[6].plot(vv_2d[i1[6],:],y_2d[i1[6],:]-y_2d[i1[6],0],'r-o',ms=4,label='Paper Data
    [203]')
380 axes[7].plot(vv_2d[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0],'r-o',ms=4,label='Paper Data
    [203]')
381 axes[0].plot(vv1_2d[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0],'b-s',ms=4,label='Journal Data
    [169]')
382 axes[1].plot(vv1_2d[i1[1],:],y_2d[i1[1],:]-y_2d[i1[1],0],'b-s',ms=4,label='Journal Data
    [169]')
383 axes[2].plot(vv1_2d[i1[2],:],y_2d[i1[2],:]-y_2d[i1[2],0],'b-s',ms=4,label='Journal Data
    [169]')
384 axes[3].plot(vv1_2d[i1[3],:],y_2d[i1[3],:]-y_2d[i1[3],0],'b-s',ms=4,label='Journal Data
    [169]')
385 axes[4].plot(vv1_2d[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0],'b-s',ms=4,label='Journal Data
    [169]')
386 axes[5].plot(vv1_2d[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0],'b-s',ms=4,label='Journal Data
    [169]')
387 axes[6].plot(vv1_2d[i1[6],:],y_2d[i1[6],:]-y_2d[i1[6],0],'b-s',ms=4,label='Journal Data
    [169]')
388 axes[7].plot(vv1_2d[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0],'b-s',ms=4,label='Journal Data
    [169]')
389 axes[1].plot(x065_off[:,5],x065_off[:,1]-y_2d[i1[1],0],'k-*',ms=4,label='Experimental
    Data')
390 axes[2].plot(x080_off[:,5],x080_off[:,1]-y_2d[i1[2],0],'k-*',ms=4,label='Experimental
    Data')
391 axes[3].plot(x090_off[:,5],x090_off[:,1]-y_2d[i1[3],0],'k-*',ms=4,label='Experimental
    Data')
392 axes[4].plot(x100_off[:,5],x100_off[:,1]-y_2d[i1[4],0],'k-*',ms=4,label='Experimental
    Data')
393 axes[5].plot(x110_off[:,5],x110_off[:,1]-y_2d[i1[5],0],'k-*',ms=4,label='Experimental
    Data')
394 axes[6].plot(x120_off[:,5],x120_off[:,1]-y_2d[i1[6],0],'k-*',ms=4,label='Experimental
    Data')
395 axes[7].plot(x130_off[:,5],x130_off[:,1]-y_2d[i1[7],0],'k-*',ms=4,label='Experimental
    Data')
396 for ax,xloc in zip(axes,xx):
397     ax.xaxis.set_minor_locator(AutoMinorLocator())
398     ax.yaxis.set_minor_locator(AutoMinorLocator())
399     ax.legend(title='At X = '+str(xloc)+' m',title_fontsize=14,fontsize=14,loc='best')
400     ax.tick_params(labelsize='medium')
401     ax.grid(True, linestyle='-.')
402     ax.set_ylim([0,0.25])
403     if (ax==axes[4] or ax==axes[5] or ax==axes[6] or ax==axes[7]):
404         ax.set_xlabel("$<\overline{v'v'}>$")
405     if (ax==axes[0] or ax==axes[4]):
406         ax.set_ylabel("$y-y_{wall}$")
407 fig11.suptitle("$<\overline{v'v'}>$ at various locations",fontsize=22)
408
409 ## Plot uv
410 fig12,axes = plt.subplots(2,4,constrained_layout=True)
411 axes = axes.reshape(-1)
```

```
412 axes[0].plot(uv_2d[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0],'r-o',ms=4,label='Paper Data
        [203]')
413 axes[1].plot(uv_2d[i1[1],:],y_2d[i1[1],:]-y_2d[i1[1],0],'r-o',ms=4,label='Paper Data
        [203]')
414 axes[2].plot(uv_2d[i1[2],:],y_2d[i1[2],:]-y_2d[i1[2],0],'r-o',ms=4,label='Paper Data
        [203]')
415 axes[3].plot(uv_2d[i1[3],:],y_2d[i1[3],:]-y_2d[i1[3],0],'r-o',ms=4,label='Paper Data
        [203]')
416 axes[4].plot(uv_2d[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0],'r-o',ms=4,label='Paper Data
        [203]')
417 axes[5].plot(uv_2d[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0],'r-o',ms=4,label='Paper Data
        [203]')
418 axes[6].plot(uv_2d[i1[6],:],y_2d[i1[6],:]-y_2d[i1[6],0],'r-o',ms=4,label='Paper Data
        [203]')
419 axes[7].plot(uv_2d[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0],'r-o',ms=4,label='Paper Data
        [203]')
420 axes[0].plot(uv1_2d[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0],'b-s',ms=4,label='Journal Data
        [169]')
421 axes[1].plot(uv1_2d[i1[1],:],y_2d[i1[1],:]-y_2d[i1[1],0],'b-s',ms=4,label='Journal Data
        [169]')
422 axes[2].plot(uv1_2d[i1[2],:],y_2d[i1[2],:]-y_2d[i1[2],0],'b-s',ms=4,label='Journal Data
        [169]')
423 axes[3].plot(uv1_2d[i1[3],:],y_2d[i1[3],:]-y_2d[i1[3],0],'b-s',ms=4,label='Journal Data
        [169]')
424 axes[4].plot(uv1_2d[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0],'b-s',ms=4,label='Journal Data
        [169]')
425 axes[5].plot(uv1_2d[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0],'b-s',ms=4,label='Journal Data
        [169]')
426 axes[6].plot(uv1_2d[i1[6],:],y_2d[i1[6],:]-y_2d[i1[6],0],'b-s',ms=4,label='Journal Data
        [169]')
427 axes[7].plot(uv1_2d[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0],'b-s',ms=4,label='Journal Data
        [169]')
428 axes[1].plot(x065_off[:,6],x065_off[:,1]-y_2d[i1[1],0],'k-*',ms=4,label='Experimental
        Data')
429 axes[2].plot(x080_off[:,6],x080_off[:,1]-y_2d[i1[2],0],'k-*',ms=4,label='Experimental
        Data')
430 axes[3].plot(x090_off[:,6],x090_off[:,1]-y_2d[i1[3],0],'k-*',ms=4,label='Experimental
        Data')
431 axes[4].plot(x100_off[:,6],x100_off[:,1]-y_2d[i1[4],0],'k-*',ms=4,label='Experimental
        Data')
432 axes[5].plot(x110_off[:,6],x110_off[:,1]-y_2d[i1[5],0],'k-*',ms=4,label='Experimental
        Data')
433 axes[6].plot(x120_off[:,6],x120_off[:,1]-y_2d[i1[6],0],'k-*',ms=4,label='Experimental
        Data')
434 axes[7].plot(x130_off[:,6],x130_off[:,1]-y_2d[i1[7],0],'k-*',ms=4,label='Experimental
        Data')
435 for ax,xloc in zip(axes,xx):
436     ax.xaxis.set_minor_locator(AutoMinorLocator())
437     ax.yaxis.set_minor_locator(AutoMinorLocator())
438     ax.legend(title='At X = '+str(xloc)+' m',title_fontsize=14,fontsize=14,loc='best')
439     ax.tick_params(labelsize='medium')
440     ax.grid(True, linestyle='-.')
441     ax.set_ylim([0,0.2])
442     if (ax==axes[4] or ax==axes[5] or ax==axes[6] or ax==axes[7]):
443         ax.set_xlabel("$<\overline{u'v'}>$")
444     if (ax==axes[0] or ax==axes[4]):
445         ax.set_ylabel("$y-y_{wall}$")
446 fig12.suptitle("$<\overline{u'v'}>$ at various locations",fontsize=22)
447
448 ## Plot u
449 fig13,axes = plt.subplots(2,4,constrained_layout=True)
450 axes = axes.reshape(-1)
451 axes[0].plot(u_2d[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0],'r-o',ms=4,label='Paper Data
        [203]')
452 axes[1].plot(u_2d[i1[1],:],y_2d[i1[1],:]-y_2d[i1[1],0],'r-o',ms=4,label='Paper Data
        [203]')
```

```
axes[2].plot(u_2d[i1[2],:],y_2d[i1[2],:]-y_2d[i1[2],0],'r-o',ms=4,label='Paper Data [203]')
axes[3].plot(u_2d[i1[3],:],y_2d[i1[3],:]-y_2d[i1[3],0],'r-o',ms=4,label='Paper Data [203]')
axes[4].plot(u_2d[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0],'r-o',ms=4,label='Paper Data [203]')
axes[5].plot(u_2d[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0],'r-o',ms=4,label='Paper Data [203]')
axes[6].plot(u_2d[i1[6],:],y_2d[i1[6],:]-y_2d[i1[6],0],'r-o',ms=4,label='Paper Data [203]')
axes[7].plot(u_2d[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0],'r-o',ms=4,label='Paper Data [203]')
axes[0].plot(u1_2d[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0],'b-s',ms=4,label='Journal Data [169]')
axes[1].plot(u1_2d[i1[1],:],y_2d[i1[1],:]-y_2d[i1[1],0],'b-s',ms=4,label='Journal Data [169]')
axes[2].plot(u1_2d[i1[2],:],y_2d[i1[2],:]-y_2d[i1[2],0],'b-s',ms=4,label='Journal Data [169]')
axes[3].plot(u1_2d[i1[3],:],y_2d[i1[3],:]-y_2d[i1[3],0],'b-s',ms=4,label='Journal Data [169]')
axes[4].plot(u1_2d[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0],'b-s',ms=4,label='Journal Data [169]')
axes[5].plot(u1_2d[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0],'b-s',ms=4,label='Journal Data [169]')
axes[6].plot(u1_2d[i1[6],:],y_2d[i1[6],:]-y_2d[i1[6],0],'b-s',ms=4,label='Journal Data [169]')
axes[7].plot(u1_2d[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0],'b-s',ms=4,label='Journal Data [169]')
axes[1].plot(x065_off[:,2],x065_off[:,1]-y_2d[i1[1],0],'k-*',ms=4,label='Experimental Data')
axes[2].plot(x080_off[:,2],x080_off[:,1]-y_2d[i1[2],0],'k-*',ms=4,label='Experimental Data')
axes[3].plot(x090_off[:,2],x090_off[:,1]-y_2d[i1[3],0],'k-*',ms=4,label='Experimental Data')
axes[4].plot(x100_off[:,2],x100_off[:,1]-y_2d[i1[4],0],'k-*',ms=4,label='Experimental Data')
axes[5].plot(x110_off[:,2],x110_off[:,1]-y_2d[i1[5],0],'k-*',ms=4,label='Experimental Data')
axes[6].plot(x120_off[:,2],x120_off[:,1]-y_2d[i1[6],0],'k-*',ms=4,label='Experimental Data')
axes[7].plot(x130_off[:,2],x130_off[:,1]-y_2d[i1[7],0],'k-*',ms=4,label='Experimental Data')
for ax,xloc in zip(axes,xx):
    ax.xaxis.set_minor_locator(AutoMinorLocator())
    ax.yaxis.set_minor_locator(AutoMinorLocator())
    ax.legend(title='At X = '+str(xloc)+' m',title_fontsize=14,fontsize=14,loc='best')
    ax.tick_params(labelsize='medium')
    ax.grid(True, linestyle='-.')
    ax.set_ylim([0,0.15])
    if (ax==axes[4] or ax==axes[5] or ax==axes[6] or ax==axes[7]):
        ax.set_xlabel("$<\overline{u}>$")
    if (ax==axes[0] or ax==axes[4]):
        ax.set_ylabel("$y-y_{wall}$")
fig13.suptitle("$<\overline{u}>$ at various locations",fontsize=22)

## Plot v
fig14,axes = plt.subplots(2,4,constrained_layout=True)
axes = axes.reshape(-1)
axes[0].plot(v_2d[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0],'r-o',ms=4,label='Paper Data [203]')
axes[1].plot(v_2d[i1[1],:],y_2d[i1[1],:]-y_2d[i1[1],0],'r-o',ms=4,label='Paper Data [203]')
axes[2].plot(v_2d[i1[2],:],y_2d[i1[2],:]-y_2d[i1[2],0],'r-o',ms=4,label='Paper Data [203]')
axes[3].plot(v_2d[i1[3],:],y_2d[i1[3],:]-y_2d[i1[3],0],'r-o',ms=4,label='Paper Data [203]')
```

```
axes[4].plot(v_2d[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0],'r-o',ms=4,label='Paper Data [203]')
axes[5].plot(v_2d[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0],'r-o',ms=4,label='Paper Data [203]')
axes[6].plot(v_2d[i1[6],:],y_2d[i1[6],:]-y_2d[i1[6],0],'r-o',ms=4,label='Paper Data [203]')
axes[7].plot(v_2d[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0],'r-o',ms=4,label='Paper Data [203]')
axes[0].plot(v1_2d[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0],'b-s',ms=4,label='Journal Data [169]')
axes[1].plot(v1_2d[i1[1],:],y_2d[i1[1],:]-y_2d[i1[1],0],'b-s',ms=4,label='Journal Data [169]')
axes[2].plot(v1_2d[i1[2],:],y_2d[i1[2],:]-y_2d[i1[2],0],'b-s',ms=4,label='Journal Data [169]')
axes[3].plot(v1_2d[i1[3],:],y_2d[i1[3],:]-y_2d[i1[3],0],'b-s',ms=4,label='Journal Data [169]')
axes[4].plot(v1_2d[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0],'b-s',ms=4,label='Journal Data [169]')
axes[5].plot(v1_2d[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0],'b-s',ms=4,label='Journal Data [169]')
axes[6].plot(v1_2d[i1[6],:],y_2d[i1[6],:]-y_2d[i1[6],0],'b-s',ms=4,label='Journal Data [169]')
axes[7].plot(v1_2d[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0],'b-s',ms=4,label='Journal Data [169]')
axes[1].plot(x065_off[:,3],x065_off[:,1]-y_2d[i1[1],0],'k-*',ms=4,label='Experimental Data')
axes[2].plot(x080_off[:,3],x080_off[:,1]-y_2d[i1[2],0],'k-*',ms=4,label='Experimental Data')
axes[3].plot(x090_off[:,3],x090_off[:,1]-y_2d[i1[3],0],'k-*',ms=4,label='Experimental Data')
axes[4].plot(x100_off[:,3],x100_off[:,1]-y_2d[i1[4],0],'k-*',ms=4,label='Experimental Data')
axes[5].plot(x110_off[:,3],x110_off[:,1]-y_2d[i1[5],0],'k-*',ms=4,label='Experimental Data')
axes[6].plot(x120_off[:,3],x120_off[:,1]-y_2d[i1[6],0],'k-*',ms=4,label='Experimental Data')
axes[7].plot(x130_off[:,3],x130_off[:,1]-y_2d[i1[7],0],'k-*',ms=4,label='Experimental Data')
for ax,xloc in zip(axes,xx):
    ax.xaxis.set_minor_locator(AutoMinorLocator())
    ax.yaxis.set_minor_locator(AutoMinorLocator())
    ax.legend(title='At X = '+str(xloc)+' m',title_fontsize=14,fontsize=14,loc='best')
    ax.tick_params(labelsize='medium')
    ax.grid(True, linestyle='-.')
    ax.set_ylim([0,0.15])
    if (ax==axes[4] or ax==axes[5] or ax==axes[6] or ax==axes[7]):
        ax.set_xlabel("$<\overline{v}>$")
    if (ax==axes[0] or ax==axes[4]):
        ax.set_ylabel("$y-y_{wall}$")
fig14.suptitle("$<\overline{v}>$ at various locations",fontsize=22)

# %% Assignment T2 - Modeled Turbulent Shear Stress
visturb_2d = vis_2d - viscos
visturb1_2d = vis1_2d - viscos
uv_model_2d_calc = -visturb_2d * (dudy+dvdx)

# Finding boundary layer thickness at the measuring points (Paper Data)
tbl065 = np.max(np.where(((vis_2d[i1[1],:]/viscos-1)<=1) == 0))
tbl080 = np.max(np.where(((vis_2d[i1[2],:]/viscos-1)<=1) == 0))
tbl090 = np.max(np.where(((vis_2d[i1[3],:]/viscos-1)<=1) == 0))
tbl100 = np.max(np.where(((vis_2d[i1[4],:]/viscos-1)<=1) == 0))
tbl110 = np.max(np.where(((vis_2d[i1[5],:]/viscos-1)<=1) == 0))
tbl120 = np.max(np.where(((vis_2d[i1[6],:]/viscos-1)<=1) == 0))
tbl130 = np.max(np.where(((vis_2d[i1[7],:]/viscos-1)<=1) == 0))

blt060 = y_2d[i1[1],tbl065]
```

```
blt080 = y_2d[i1[2],tbl080]
blt090 = y_2d[i1[3],tbl090]
blt100 = y_2d[i1[4],tbl100]
blt110 = y_2d[i1[5],tbl110]
blt120 = y_2d[i1[6],tbl120]
blt130 = y_2d[i1[7],tbl130]

# Finding boundary layer thickness at the measuring points (Journal Data)
tbl1065 = np.max(np.where(((vis1_2d[i1[1],:]/viscos-1)<=1) == 0))
tbl1080 = np.max(np.where(((vis1_2d[i1[2],:]/viscos-1)<=1) == 0))
tbl1090 = np.max(np.where(((vis1_2d[i1[3],:]/viscos-1)<=1) == 0))
tbl1100 = np.max(np.where(((vis1_2d[i1[4],:]/viscos-1)<=1) == 0))
tbl1110 = np.max(np.where(((vis1_2d[i1[5],:]/viscos-1)<=1) == 0))
tbl1120 = np.max(np.where(((vis1_2d[i1[6],:]/viscos-1)<=1) == 0))
tbl1130 = np.max(np.where(((vis1_2d[i1[7],:]/viscos-1)<=1) == 0))

blt1060 = y_2d[i1[1],tbl1065]
blt1080 = y_2d[i1[2],tbl1080]
blt1090 = y_2d[i1[3],tbl1090]
blt1100 = y_2d[i1[4],tbl1100]
blt1110 = y_2d[i1[5],tbl1110]
blt1120 = y_2d[i1[6],tbl1120]
blt1130 = y_2d[i1[7],tbl1130]

fig17,axes = plt.subplots(2,4,constrained_layout=True)
axes = axes.reshape(-1)
axes[0].plot(-uv_model_2d[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0],'b-o',ms=4,label='Modeled')
axes[1].plot(-uv_model_2d[i1[1],:],y_2d[i1[1],:]-y_2d[i1[1],0],'b-o',ms=4,label='Modeled')
axes[2].plot(-uv_model_2d[i1[2],:],y_2d[i1[2],:]-y_2d[i1[2],0],'b-o',ms=4,label='Modeled')
axes[3].plot(-uv_model_2d[i1[3],:],y_2d[i1[3],:]-y_2d[i1[3],0],'b-o',ms=4,label='Modeled')
axes[4].plot(-uv_model_2d[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0],'b-o',ms=4,label='Modeled')
axes[5].plot(-uv_model_2d[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0],'b-o',ms=4,label='Modeled')
axes[6].plot(-uv_model_2d[i1[6],:],y_2d[i1[6],:]-y_2d[i1[6],0],'b-o',ms=4,label='Modeled')
axes[7].plot(-uv_model_2d[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0],'b-o',ms=4,label='Modeled')
axes[0].plot(-uv_2d[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0],'r-o',ms=4,label='Resolved')
axes[1].plot(-uv_2d[i1[1],:],y_2d[i1[1],:]-y_2d[i1[1],0],'r-o',ms=4,label='Resolved')
axes[2].plot(-uv_2d[i1[2],:],y_2d[i1[2],:]-y_2d[i1[2],0],'r-o',ms=4,label='Resolved')
axes[3].plot(-uv_2d[i1[3],:],y_2d[i1[3],:]-y_2d[i1[3],0],'r-o',ms=4,label='Resolved')
axes[4].plot(-uv_2d[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0],'r-o',ms=4,label='Resolved')
axes[5].plot(-uv_2d[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0],'r-o',ms=4,label='Resolved')
axes[6].plot(-uv_2d[i1[6],:],y_2d[i1[6],:]-y_2d[i1[6],0],'r-o',ms=4,label='Resolved')
axes[7].plot(-uv_2d[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0],'r-o',ms=4,label='Resolved')
axes[1].plot(-x065_off[:,6],x065_off[:,1]-y_2d[i1[1],0],'k-*',ms=4,label='Experimental')
axes[2].plot(-x080_off[:,6],x080_off[:,1]-y_2d[i1[2],0],'k-*',ms=4,label='Experimental')
axes[3].plot(-x090_off[:,6],x090_off[:,1]-y_2d[i1[3],0],'k-*',ms=4,label='Experimental')
axes[4].plot(-x100_off[:,6],x100_off[:,1]-y_2d[i1[4],0],'k-*',ms=4,label='Experimental')
axes[5].plot(-x110_off[:,6],x110_off[:,1]-y_2d[i1[5],0],'k-*',ms=4,label='Experimental')
axes[6].plot(-x120_off[:,6],x120_off[:,1]-y_2d[i1[6],0],'k-*',ms=4,label='Experimental')
axes[7].plot(-x130_off[:,6],x130_off[:,1]-y_2d[i1[7],0],'k-*',ms=4,label='Experimental')
axes[1].plot(np.linspace(np.min(-uv_2d[i1[1],:]),np.max(-uv_2d[i1[1],:]),10),(y_2d[i1
    [1],tbl065]-y_2d[i1[1],0])*np.ones(10),'k--',ms=4,label='BL thickness')
axes[2].plot(np.linspace(np.min(-uv_2d[i1[2],:]),np.max(-uv_2d[i1[2],:]),10),(y_2d[i1
    [2],tbl080]-y_2d[i1[2],0])*np.ones(10),'k--',ms=4,label='BL thickness')
axes[3].plot(np.linspace(np.min(-uv_2d[i1[3],:]),np.max(-uv_2d[i1[3],:]),10),(y_2d[i1
    [3],tbl090]-y_2d[i1[3],0])*np.ones(10),'k--',ms=4,label='BL thickness')
axes[4].plot(np.linspace(np.min(-uv_2d[i1[4],:]),np.max(-uv_2d[i1[4],:]),10),(y_2d[i1
    [4],tbl100]-y_2d[i1[4],0])*np.ones(10),'k--',ms=4,label='BL thickness')
axes[5].plot(np.linspace(np.min(-uv_2d[i1[5],:]),np.max(-uv_2d[i1[5],:]),10),(y_2d[i1
```

```
    [5],tbl110]-y_2d[i1[5],0])*np.ones(10),'k--',ms=4,label='BL thickness')
axes[6].plot(np.linspace(np.min(-uv_2d[i1[6],:]),np.max(-uv_2d[i1[6],:]),10),(y_2d[i1
    [6],tbl120]-y_2d[i1[6],0])*np.ones(10),'k--',ms=4,label='BL thickness')
axes[7].plot(np.linspace(np.min(-uv_2d[i1[7],:]),np.max(-uv_2d[i1[7],:]),10),(y_2d[i1
    [7],tbl130]-y_2d[i1[7],0])*np.ones(10),'k--',ms=4,label='BL thickness')
for ax,xloc in zip(axes,xx):
    ax.xaxis.set_minor_locator(AutoMinorLocator())
    ax.yaxis.set_minor_locator(AutoMinorLocator())
    ax.legend(title='At X = '+str(xloc)+' m',title_fontsize=14,fontsize=14,loc='best')
    ax.tick_params(labelsize='medium')
    ax.grid(True, linestyle='-.')
    ax.set_ylim([0,0.25])
    if (ax==axes[0] or ax==axes[4]):
        ax.set_ylabel("$y-y_{wall}$")
    if (ax==axes[4] or ax==axes[5] or ax==axes[6] or ax==axes[7]):
        ax.set_xlabel("$-<\overline{u'v'}>$")
fig17.suptitle("Turbulent Shear Stress -$<\overline{u'v'}>$ (Reference [203])",fontsize
    =22)

fig17a,axes = plt.subplots(1,1,constrained_layout=True)
a1 = axes.contourf(x_2d,y_2d,-uv_model_2d,cmap='jet',levels=50)
cb1 = fig17a.colorbar(a1,ax=axes)
cb1.ax.tick_params(labelsize=22)
a1.ax.tick_params(labelsize=16)
plt.axis([0.6,1.5,0,1])
axes.set_xlabel('X (m)',fontsize=18)
axes.set_ylabel('Y (m)',fontsize=18)
axes.set_title("Modeled Turbulent Shear Stress -$<\overline{u'v'}>$ (Reference [203])",
    fontsize=24)

fig17b,axes = plt.subplots(1,1,constrained_layout=True)
a1 = axes.contourf(x_2d,y_2d,-uv_2d,cmap='jet',levels=50)
cb1 = fig17b.colorbar(a1,ax=axes)
cb1.ax.tick_params(labelsize=22)
a1.ax.tick_params(labelsize=16)
plt.axis([0.6,1.5,0,1])
axes.set_xlabel('X (m)',fontsize=18)
axes.set_ylabel('Y (m)',fontsize=18)
axes.set_title("Resolved Turbulent Shear Stress -$<\overline{u'v'}>$ (Reference [203])",
    fontsize=24)

fig17c,axes = plt.subplots(2,4,constrained_layout=True)
axes = axes.reshape(-1)
axes[0].plot(-uv1_model_2d[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0],'b-o',ms=4,label='
    Modeled')
axes[1].plot(-uv1_model_2d[i1[1],:],y_2d[i1[1],:]-y_2d[i1[1],0],'b-o',ms=4,label='
    Modeled')
axes[2].plot(-uv1_model_2d[i1[2],:],y_2d[i1[2],:]-y_2d[i1[2],0],'b-o',ms=4,label='
    Modeled')
axes[3].plot(-uv1_model_2d[i1[3],:],y_2d[i1[3],:]-y_2d[i1[3],0],'b-o',ms=4,label='
    Modeled')
axes[4].plot(-uv1_model_2d[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0],'b-o',ms=4,label='
    Modeled')
axes[5].plot(-uv1_model_2d[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0],'b-o',ms=4,label='
    Modeled')
axes[6].plot(-uv1_model_2d[i1[6],:],y_2d[i1[6],:]-y_2d[i1[6],0],'b-o',ms=4,label='
    Modeled')
axes[7].plot(-uv1_model_2d[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0],'b-o',ms=4,label='
    Modeled')
axes[0].plot(-uv1_2d[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0],'r-o',ms=4,label='Resolved')
axes[1].plot(-uv1_2d[i1[1],:],y_2d[i1[1],:]-y_2d[i1[1],0],'r-o',ms=4,label='Resolved')
axes[2].plot(-uv1_2d[i1[2],:],y_2d[i1[2],:]-y_2d[i1[2],0],'r-o',ms=4,label='Resolved')
axes[3].plot(-uv1_2d[i1[3],:],y_2d[i1[3],:]-y_2d[i1[3],0],'r-o',ms=4,label='Resolved')
axes[4].plot(-uv1_2d[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0],'r-o',ms=4,label='Resolved')
axes[5].plot(-uv1_2d[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0],'r-o',ms=4,label='Resolved')
axes[6].plot(-uv1_2d[i1[6],:],y_2d[i1[6],:]-y_2d[i1[6],0],'r-o',ms=4,label='Resolved')
```

```
axes[7].plot(-uv1_2d[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0],'r-o',ms=4,label='Resolved')
axes[1].plot(-x065_off[:,6],x065_off[:,1]-y_2d[i1[1],0],'k-*',ms=4,label='Experimental')
axes[2].plot(-x080_off[:,6],x080_off[:,1]-y_2d[i1[2],0],'k-*',ms=4,label='Experimental')
axes[3].plot(-x090_off[:,6],x090_off[:,1]-y_2d[i1[3],0],'k-*',ms=4,label='Experimental')
axes[4].plot(-x100_off[:,6],x100_off[:,1]-y_2d[i1[4],0],'k-*',ms=4,label='Experimental')
axes[5].plot(-x110_off[:,6],x110_off[:,1]-y_2d[i1[5],0],'k-*',ms=4,label='Experimental')
axes[6].plot(-x120_off[:,6],x120_off[:,1]-y_2d[i1[6],0],'k-*',ms=4,label='Experimental')
axes[7].plot(-x130_off[:,6],x130_off[:,1]-y_2d[i1[7],0],'k-*',ms=4,label='Experimental')
axes[1].plot(np.linspace(np.min(-uv1_2d[i1[1],:]),np.max(-uv1_2d[i1[1],:]),10),(y_2d[i1
    [1],tbl1065]-y_2d[i1[1],0])*np.ones(10),'k--',ms=4,label='BL thickness')
axes[2].plot(np.linspace(np.min(-uv1_2d[i1[2],:]),np.max(-uv1_2d[i1[2],:]),10),(y_2d[i1
    [2],tbl1080]-y_2d[i1[2],0])*np.ones(10),'k--',ms=4,label='BL thickness')
axes[3].plot(np.linspace(np.min(-uv1_2d[i1[3],:]),np.max(-uv1_2d[i1[3],:]),10),(y_2d[i1
    [3],tbl1090]-y_2d[i1[3],0])*np.ones(10),'k--',ms=4,label='BL thickness')
axes[4].plot(np.linspace(np.min(-uv1_2d[i1[4],:]),np.max(-uv1_2d[i1[4],:]),10),(y_2d[i1
    [4],tbl1100]-y_2d[i1[4],0])*np.ones(10),'k--',ms=4,label='BL thickness')
axes[5].plot(np.linspace(np.min(-uv1_2d[i1[5],:]),np.max(-uv1_2d[i1[5],:]),10),(y_2d[i1
    [5],tbl1110]-y_2d[i1[5],0])*np.ones(10),'k--',ms=4,label='BL thickness')
axes[6].plot(np.linspace(np.min(-uv1_2d[i1[6],:]),np.max(-uv1_2d[i1[6],:]),10),(y_2d[i1
    [6],tbl1120]-y_2d[i1[6],0])*np.ones(10),'k--',ms=4,label='BL thickness')
axes[7].plot(np.linspace(np.min(-uv1_2d[i1[7],:]),np.max(-uv1_2d[i1[7],:]),10),(y_2d[i1
    [7],tbl1130]-y_2d[i1[7],0])*np.ones(10),'k--',ms=4,label='BL thickness')
for ax,xloc in zip(axes,xx):
    ax.xaxis.set_minor_locator(AutoMinorLocator())
    ax.yaxis.set_minor_locator(AutoMinorLocator())
    ax.legend(title='At X = '+str(xloc)+' m',title_fontsize=14,fontsize=14,loc='best')
    ax.tick_params(labelsize='medium')
    ax.grid(True, linestyle='-.')
    ax.set_ylim([0,0.25])
    if (ax==axes[0] or ax==axes[4]):
        ax.set_ylabel("$y-y_{wall}$")
    if (ax==axes[4] or ax==axes[5] or ax==axes[6] or ax==axes[7]):
        ax.set_xlabel("$-<\overline{u'v'}>$")
fig17c.suptitle("Turbulent Shear Stress -$<\overline{u'v'}>$ (Reference [169])",fontsize
    =22)

fig17d,axes = plt.subplots(1,1,constrained_layout=True)
a1 = axes.contourf(x_2d,y_2d,-uv1_model_2d,cmap='jet',levels=50)
cb1 = fig17d.colorbar(a1,ax=axes)
cb1.ax.tick_params(labelsize=22)
a1.ax.tick_params(labelsize=16)
plt.axis([0.6,1.5,0,1])
axes.set_xlabel('X (m)',fontsize=18)
axes.set_ylabel('Y (m)',fontsize=18)
axes.set_title("Modeled Turbulent Shear Stress -$<\overline{u'v'}>$ (Reference [169])",
    fontsize=24)

fig17e,axes = plt.subplots(1,1,constrained_layout=True)
a1 = axes.contourf(x_2d,y_2d,-uv1_2d,cmap='jet',levels=50)
cb1 = fig17e.colorbar(a1,ax=axes)
cb1.ax.tick_params(labelsize=22)
a1.ax.tick_params(labelsize=16)
plt.axis([0.6,1.5,0,1])
axes.set_xlabel('X (m)',fontsize=18)
axes.set_ylabel('Y (m)',fontsize=18)
axes.set_title("Resolved Turbulent Shear Stress -$<\overline{u'v'}>$ (Reference [169])",
    fontsize=24)

# %% Assignment T3 - The Turbulent Viscosity
visc_ratio = visturb_2d/viscos
visc1_ratio = visturb1_2d/viscos

# Visualize Turbulent viscosity ratio
vmin1 = 1
vmax1 = np.max(np.max(visc_ratio))
v1 = np.linspace(vmin1,vmax1,100)
```

```
vmin2 = 1
vmax2 = np.max(np.max(visc1_ratio))
v2 = np.linspace(vmin2,vmax2,100)
fig15,axes = plt.subplots(2,1,constrained_layout=True)
a1 = axes[0].contourf(x_2d,y_2d,visc_ratio,cmap='gist_gray',levels=v1,extend='min')
a2 = axes[1].contourf(x_2d,y_2d,visc1_ratio,cmap='gist_gray',levels=v2,extend='min')
a1.cmap.set_under('black')
a2.cmap.set_under('black')
cb1 = fig15.colorbar(a1,ax=axes[0])
cb2 = fig15.colorbar(a2,ax=axes[1])
cb1.ax.tick_params(labelsize=16)
cb2.ax.tick_params(labelsize=16)
a1.ax.tick_params(labelsize=16)
a2.ax.tick_params(labelsize=16)
axes[0].set_title(r'Turbulent Viscosity Ratio - $\nu_{t}/\nu$ (Paper Data [203])',
    fontsize=22)
axes[1].set_title(r'Turbulent Viscosity Ratio - $\nu_{t}/\nu$ (Journal Data [169])',
    fontsize=22)
for ax in axes:
    ax.axis([0.6,1.5,0,1])
    ax.set_xlabel('X (m)',fontsize=18)
    ax.set_ylabel('Y (m)',fontsize=18)

fig19,axes = plt.subplots(2,4,constrained_layout=True)
axes = axes.reshape(-1)
axes[0].plot(visc_ratio[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0],'b-o',ms=4,label='Paper
    Data [203]')
axes[1].plot(visc_ratio[i1[1],:],y_2d[i1[1],:]-y_2d[i1[1],0],'b-o',ms=4,label='Paper
    Data [203]')
axes[2].plot(visc_ratio[i1[2],:],y_2d[i1[2],:]-y_2d[i1[2],0],'b-o',ms=4,label='Paper
    Data [203]')
axes[3].plot(visc_ratio[i1[3],:],y_2d[i1[3],:]-y_2d[i1[3],0],'b-o',ms=4,label='Paper
    Data [203]')
axes[4].plot(visc_ratio[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0],'b-o',ms=4,label='Paper
    Data [203]')
axes[5].plot(visc_ratio[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0],'b-o',ms=4,label='Paper
    Data [203]')
axes[6].plot(visc_ratio[i1[6],:],y_2d[i1[6],:]-y_2d[i1[6],0],'b-o',ms=4,label='Paper
    Data [203]')
axes[7].plot(visc_ratio[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0],'b-o',ms=4,label='Paper
    Data [203]')
axes[0].plot(visc1_ratio[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0],'r-o',ms=4,label='Journal
    Data [169]')
axes[1].plot(visc1_ratio[i1[1],:],y_2d[i1[1],:]-y_2d[i1[1],0],'r-o',ms=4,label='Journal
    Data [169]')
axes[2].plot(visc1_ratio[i1[2],:],y_2d[i1[2],:]-y_2d[i1[2],0],'r-o',ms=4,label='Journal
    Data [169]')
axes[3].plot(visc1_ratio[i1[3],:],y_2d[i1[3],:]-y_2d[i1[3],0],'r-o',ms=4,label='Journal
    Data [169]')
axes[4].plot(visc1_ratio[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0],'r-o',ms=4,label='Journal
    Data [169]')
axes[5].plot(visc1_ratio[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0],'r-o',ms=4,label='Journal
    Data [169]')
axes[6].plot(visc1_ratio[i1[6],:],y_2d[i1[6],:]-y_2d[i1[6],0],'r-o',ms=4,label='Journal
    Data [169]')
axes[7].plot(visc1_ratio[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0],'r-o',ms=4,label='Journal
    Data [169]')
for ax,xloc in zip(axes,xx):
    ax.xaxis.set_minor_locator(AutoMinorLocator())
    ax.yaxis.set_minor_locator(AutoMinorLocator())
    ax.legend(title='At X = '+str(xloc)+' m',title_fontsize=14,fontsize=14,loc='best')
    ax.tick_params(labelsize='medium')
    ax.grid(True, linestyle='-.')
    if (ax==axes[0] or ax==axes[4]):
        ax.set_ylabel("$y-y_{wall}$")
    if (ax==axes[4] or ax==axes[5] or ax==axes[6] or ax==axes[7]):
```

```
        ax.set_xlabel(r"$\nu_{t}/\nu$")
fig19.suptitle(r'Turbulent Viscosity Ratio - $\nu_{t}/\nu$',fontsize=22)

# term 1,5 (i=1 and j=1)
# term 2,6 (i=1 and j=2)
# term 3,7 (i=2 and j=1)
# term 4,8 (i=2 and j=2)
term1,dummy1 = dphidx_dy(x_2d_new,y_2d_new,visturb_2d*dudx)
dummy2,term2 = dphidx_dy(x_2d_new,y_2d_new,visturb_2d*dudy)
term3,dummy3 = dphidx_dy(x_2d_new,y_2d_new,visturb_2d*dvdx)
dummy4,term4 = dphidx_dy(x_2d_new,y_2d_new,visturb_2d*dvdy)

term5,dummy5 = dphidx_dy(x_2d_new,y_2d_new,uu_2d)
dummy6,term6 = dphidx_dy(x_2d_new,y_2d_new,uv_2d)
term7,dummy7 = dphidx_dy(x_2d_new,y_2d_new,uv_2d)
dummy8,term8 = dphidx_dy(x_2d_new,y_2d_new,vv_2d)

div1_i1 = term1 + term2
div1_i2 = term3 + term4
div2_i1 = -term5 - term6
div2_i2 = -term7 - term8

fig20,axes = plt.subplots(2,4,constrained_layout=True)
axes = axes.reshape(-1)
axes[0].plot(div1_i1[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0],'b-o',ms=4,label=r'Modeled')
axes[1].plot(div1_i1[i1[1],:],y_2d[i1[1],:]-y_2d[i1[1],0],'b-o',ms=4,label=r'Modeled')
axes[2].plot(div1_i1[i1[2],:],y_2d[i1[2],:]-y_2d[i1[2],0],'b-o',ms=4,label=r'Modeled')
axes[3].plot(div1_i1[i1[3],:],y_2d[i1[3],:]-y_2d[i1[3],0],'b-o',ms=4,label=r'Modeled')
axes[4].plot(div1_i1[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0],'b-o',ms=4,label=r'Modeled')
axes[5].plot(div1_i1[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0],'b-o',ms=4,label=r'Modeled')
axes[6].plot(div1_i1[i1[6],:],y_2d[i1[6],:]-y_2d[i1[6],0],'b-o',ms=4,label=r'Modeled')
axes[7].plot(div1_i1[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0],'b-o',ms=4,label=r'Modeled')
axes[0].plot(div2_i1[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0],'r-o',ms=4,label=r"Resolved")
axes[1].plot(div2_i1[i1[1],:],y_2d[i1[1],:]-y_2d[i1[1],0],'r-o',ms=4,label=r"Resolved")
axes[2].plot(div2_i1[i1[2],:],y_2d[i1[2],:]-y_2d[i1[2],0],'r-o',ms=4,label=r"Resolved")
axes[3].plot(div2_i1[i1[3],:],y_2d[i1[3],:]-y_2d[i1[3],0],'r-o',ms=4,label=r"Resolved")
axes[4].plot(div2_i1[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0],'r-o',ms=4,label=r"Resolved")
axes[5].plot(div2_i1[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0],'r-o',ms=4,label=r"Resolved")
axes[6].plot(div2_i1[i1[6],:],y_2d[i1[6],:]-y_2d[i1[6],0],'r-o',ms=4,label=r"Resolved")
axes[7].plot(div2_i1[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0],'r-o',ms=4,label=r"Resolved")
for ax,xloc in zip(axes,xx):
    ax.xaxis.set_minor_locator(AutoMinorLocator())
    ax.yaxis.set_minor_locator(AutoMinorLocator())
    ax.legend(title='At X = '+str(xloc)+' m',title_fontsize=14,fontsize=14,loc='best')
    ax.tick_params(labelsize='medium')
    ax.grid(True, linestyle='-.')
    ax.set_ylim([0,0.2])
    if (ax==axes[0] or ax==axes[4]):
        ax.set_ylabel("$y-y_{wall}$")
fig20.suptitle(r"Modeled ($\frac{\partial}{\partial x_{j}}(\langle \nu_{t} \frac{\
    partial \overline{v_{1}}}{\partial x_{j}} \rangle)$) and Resolved ($- \frac{\partial
     \langle v'_{1}v'_{j} \rangle}{\partial x_{j}}$) Shear Stress",fontsize=22)

fig21,axes = plt.subplots(2,4,constrained_layout=True)
axes = axes.reshape(-1)
axes[0].plot(div1_i2[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0],'b-o',ms=4,label=r'Modeled')
axes[1].plot(div1_i2[i1[1],:],y_2d[i1[1],:]-y_2d[i1[1],0],'b-o',ms=4,label=r'Modeled')
axes[2].plot(div1_i2[i1[2],:],y_2d[i1[2],:]-y_2d[i1[2],0],'b-o',ms=4,label=r'Modeled')
axes[3].plot(div1_i2[i1[3],:],y_2d[i1[3],:]-y_2d[i1[3],0],'b-o',ms=4,label=r'Modeled')
axes[4].plot(div1_i2[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0],'b-o',ms=4,label=r'Modeled')
axes[5].plot(div1_i2[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0],'b-o',ms=4,label=r'Modeled')
axes[6].plot(div1_i2[i1[6],:],y_2d[i1[6],:]-y_2d[i1[6],0],'b-o',ms=4,label=r'Modeled')
axes[7].plot(div1_i2[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0],'b-o',ms=4,label=r'Modeled')
axes[0].plot(div2_i2[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0],'r-o',ms=4,label=r"Resolved")
axes[1].plot(div2_i2[i1[1],:],y_2d[i1[1],:]-y_2d[i1[1],0],'r-o',ms=4,label=r"Resolved")
axes[2].plot(div2_i2[i1[2],:],y_2d[i1[2],:]-y_2d[i1[2],0],'r-o',ms=4,label=r"Resolved")
```

```
axes[3].plot(div2_i2[i1[3],:],y_2d[i1[3],:]-y_2d[i1[3],0],'r-o',ms=4,label=r"Resolved")
axes[4].plot(div2_i2[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0],'r-o',ms=4,label=r"Resolved")
axes[5].plot(div2_i2[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0],'r-o',ms=4,label=r"Resolved")
axes[6].plot(div2_i2[i1[6],:],y_2d[i1[6],:]-y_2d[i1[6],0],'r-o',ms=4,label=r"Resolved")
axes[7].plot(div2_i2[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0],'r-o',ms=4,label=r"Resolved")
for ax,xloc in zip(axes,xx):
    ax.xaxis.set_minor_locator(AutoMinorLocator())
    ax.yaxis.set_minor_locator(AutoMinorLocator())
    ax.legend(title='At X = '+str(xloc)+' m',title_fontsize=14,fontsize=14,loc='best')
    ax.tick_params(labelsize='medium')
    ax.grid(True, linestyle='-.')
    ax.set_ylim([0,0.2])
    if (ax==axes[0] or ax==axes[4]):
        ax.set_ylabel("$y-y_{wall}$")
fig21.suptitle(r"Modeled($\frac{\partial}{\partial x_{j}}(\langle \nu_{t} \frac{\partial
    \overline{v_{2}}}{\partial x_{j}} \rangle)$) and Resolved ($- \frac{\partial \
    langle v'_{2}v'_{j} \rangle}{\partial x_{j}}$) Shear Stress",fontsize=22)

# %% For visualization - location of experimental points
fig22 = plt.figure()
axes = fig22.add_axes([0.1,0.1,0.8,0.8])
axes.plot(x_2d[:,0],y_2d[:,0],lw=4,color='black')
axes.plot(x_2d[0,:],y_2d[0,:],lw=4,color='black')
axes.plot(x_2d[:,-1],y_2d[:,-1],lw=4,color='black')
axes.plot(x_2d[-1,:],y_2d[-1,:],lw=4,color='black')
axes.plot(x065_off[:,0],np.linspace(x065_off[-1,1],0.6,x065_off.shape[0]),'b--',label='X
    = 0.65 m')
axes.plot(x080_off[:,0],np.linspace(x080_off[-1,1],0.6,x080_off.shape[0]),'g--',label='X
    = 0.80 m')
axes.plot(x090_off[:,0],np.linspace(x090_off[-1,1],0.6,x090_off.shape[0]),'r--',label='X
    = 0.90 m')
axes.plot(x100_off[:,0],np.linspace(x100_off[-1,1],0.6,x100_off.shape[0]),'c--',label='X
    = 0.10 m')
axes.plot(x110_off[:,0],np.linspace(x110_off[-1,1],0.6,x110_off.shape[0]),'m--',label='X
    = 0.11 m')
axes.plot(x120_off[:,0],np.linspace(x120_off[-1,1],0.6,x120_off.shape[0]),'y--',label='X
    = 0.12 m')
axes.plot(x130_off[:,0],np.linspace(x130_off[-1,1],0.6,x130_off.shape[0]),'k--',label='X
    = 0.13 m')
axes.legend(fontsize=14,loc='best')
axes.tick_params(axis="x", labelsize=15)
axes.tick_params(axis="y", labelsize=15)
axes.set_xlabel('x [m]',fontsize=20)
axes.set_ylabel('y [m]',fontsize=20)
axes.set_title('Location of Points for Experimental Data',fontsize=24)
```

For Assignment 2b : Section 2.4 to 2.5

```
import scipy.io as sio
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1.inset_locator import inset_axes
from dphidx_dy import dphidx_dy
from IPython import display
import numpy.ma as ma
from matplotlib.ticker import (MultipleLocator, FormatStrFormatter,
                               AutoMinorLocator)
plt.rcParams.update({'font.size': 22})

# makes sure figures are updated when using ipython
display.clear_output()


re =9.36e+5
viscos =1/re

xy_hump_fine = np.loadtxt("xy_hump_fine.dat")
x1=xy_hump_fine[:,0]
y1=xy_hump_fine[:,1]


nim1=int(x1[0])
njm1=int(y1[0])

ni=nim1+1
nj=njm1+1


x=x1[1:]
y=y1[1:]

x_2d=np.reshape(x,(njm1,nim1))
y_2d=np.reshape(y,(njm1,nim1))

x_2d=np.transpose(x_2d)
y_2d=np.transpose(y_2d)

# compute cell centers
xp2d= np.zeros((ni,nj))
yp2d= np.zeros((ni,nj))

for jj in range (0,nj):
   for ii in range (0,ni):

      im1=max(ii-1,0)
      jm1=max(jj-1,0)

      i=min(ii,nim1-1)
      j=min(jj,njm1-1)


      xp2d[ii,jj]=0.25*(x_2d[i,j]+x_2d[im1,j]+x_2d[i,jm1]+x_2d[im1,jm1])
      yp2d[ii,jj]=0.25*(y_2d[i,j]+y_2d[im1,j]+y_2d[i,jm1]+y_2d[im1,jm1])

# read data file
vectz = np.loadtxt("vectz_fine.dat")
ntstep=vectz[0]
ni=int(vectz[1])
nj=int(vectz[2])
nk=int(vectz[3])
n=len(vectz)

```

```
#             write(48,*)uvec(i,j)
#             write(48,*)vvec(i,j)
#             write(48,*)fk2d(i,j)
#             write(48,*)uvec2(i,j)
#             write(48,*)vvec2(i,j)
#             write(48,*)wvec2(i,j)
#             write(48,*)uvvec(i,j)
#             write(48,*)p2d(i,j)
#             write(48,*)rk2d(i,j)
#             write(48,*)vis2d(i,j)
#             write(48,*)dissp2d(i,j)
#             write(48,*)wvec(i,j)
#             write(48,*)vtvec(i,j)
#             write(48,*)tvec(i,j)


nn=14
nst=3
iu=range(nst+1,n,nn)
iv=range(nst+2,n,nn)
ifk=range(nst+3,n,nn)
iuu=range(nst+4,n,nn)
ivv=range(nst+5,n,nn)
iww=range(nst+6,n,nn)
iuv=range(nst+7,n,nn)
ip=range(nst+8,n,nn)
ik=range(nst+9,n,nn)
ivis=range(nst+10,n,nn)
idiss=range(nst+11,n,nn)

u=vectz[iu]/ntstep
v=vectz[iv]/ntstep
fk=vectz[ifk]/ntstep
uu=vectz[iuu]/ntstep
vv=vectz[ivv]/ntstep
ww=vectz[iww]/ntstep
uv=vectz[iuv]/ntstep
p=vectz[ip]/ntstep
k_model=vectz[ik]/ntstep
vis=vectz[ivis]/ntstep
diss=vectz[idiss]/ntstep

# uu is total inst. velocity squared. Hence the resolved turbulent resolved stresses are
    obtained as
uu=uu-u**2
vv=vv-v**2
uv=uv-u*v

p_2d=np.reshape(p,(ni,nj))
u_2d=np.reshape(u,(ni,nj))
v_2d=np.reshape(v,(ni,nj))
fk_2d=np.reshape(fk,(ni,nj))
uu_2d=np.reshape(uu,(ni,nj))
uv_2d=np.reshape(uv,(ni,nj))
vv_2d=np.reshape(vv,(ni,nj))
ww_2d=np.reshape(ww,(ni,nj))
k_model_2d=np.reshape(k_model,(ni,nj))
vis_2d=np.reshape(vis,(ni,nj)) #this is to total viscosity, i.e. vis_tot=vis+vis_turb
diss_2d=np.reshape(diss,(ni,nj)) #this is to total viscosity, i.e. vis_tot=vis+vis_turb

# set fk_2d=1 at upper boundary
fk_2d[:,nj-1]=fk_2d[:,nj-2]

# Plot Mesh for Visualization
a = np.transpose(xp2d)
b = np.transpose(yp2d)
```

```
c = np.transpose(x_2d)
d = np.transpose(y_2d)
fig1a = plt.figure()
axes = fig1a.add_axes([0.1,0.1,0.8,0.8])
#axes.plot(xp2d,yp2d,'r.')
#axes.plot(a,b,'r.')
axes.plot(x_2d,y_2d,'k-')
axes.plot(c,d,'k-')
#axes.plot(x_2d,y_2d,'b.')
#axes.plot(c,d,'b.')
axes.tick_params(axis="x", labelsize=15)
axes.tick_params(axis="y", labelsize=15)
axes.set_xlabel('x [m]',fontsize=20)
axes.set_ylabel('y [m]',fontsize=20)
axes.set_title('Computational mesh in the domain',fontsize=24)

dz=0.2/32
dx = np.zeros([ni,nj])
dy = np.zeros([ni,nj])
dx_dummy = x_2d[1:,1:] - x_2d[:-1,1:]
dy_dummy = y_2d[1:,1:] - y_2d[1:,:-1]
dx[1:-1,1:-1] = dx_dummy
dy[1:-1,1:-1] = dy_dummy
dx[0,:] = dx[1,:]
dx[:,0] = dx[:,1]
dx[-1,:] = dx[-2,:]
dx[:,-1] = dx[:,-2]
dy[0,:] = dy[1,:]
dy[:,0] = dy[:,1]
dy[-1,:] = dy[-2,:]
dy[:,-1] = dy[:,-2]
delta = (dx*dy*dx)**(1/3)
#delta = np.maximum.reduce([dx,dy,dz*np.ones([ni,nj])])

x065_off=np.genfromtxt("x065_off.dat",dtype=None,comments="%")
x066_off=np.genfromtxt("x066_off.dat",dtype=None,comments="%")
x080_off=np.genfromtxt("x080_off.dat",dtype=None,comments="%")
x090_off=np.genfromtxt("x090_off.dat",dtype=None,comments="%")
x100_off=np.genfromtxt("x100_off.dat",dtype=None,comments="%")
x110_off=np.genfromtxt("x110_off.dat",dtype=None,comments="%")
x120_off=np.genfromtxt("x120_off.dat",dtype=None,comments="%")
x130_off=np.genfromtxt("x130_off.dat",dtype=None,comments="%")


# compute the gradient
dudx,dudy=dphidx_dy(x_2d,y_2d,u_2d)


#*************************
# plot u
fig1b,ax1 = plt.subplots()
xx=0.65;
i1 = (np.abs(xx-xp2d[:,1])).argmin()  # find index which closest fits xx
plt.plot(u_2d[i1,:],yp2d[i1,:],'b-')
plt.plot(x065_off[:,2],x065_off[:,1],'bo')
plt.xlabel("$U$")
plt.ylabel("$y$")
plt.title("$x=0.65$")
plt.axis([0, 1.3,0.115,0.2])
# Create inset of width 30% and height 40% of the parent axes' bounding box
# at the lower left corner (loc=3)
# upper left corner (loc=2)
# use borderpad=1, i.e.
# 22 points padding (as 22pt is the default fontsize) to the parent axes
axins1 = inset_axes(ax1, width="40%", height="30%", loc=2, borderpad=1)
plt.plot(u_2d[i1,:],yp2d[i1,:],'b-')
```

```
plt.axis([0, 1.3,0.115,0.13])
# reduce fotnsize
axins1.tick_params(axis = 'both', which = 'major', labelsize = 10)
# Turn ticklabels of insets off
axins1.tick_params(labelleft=False, labelbottom=False)
plt.plot(x065_off[:,2],x065_off[:,1],'bo')
#plt.savefig('u065_hump_python.eps',bbox_inches='tight')

# %% Assignment T4 - Location of Interface
# RANS to LES where fk goes down from 1 to 0.4
C_mu = 0.09
k_res_2d = 0.5*(uu_2d+vv_2d+ww_2d)
k_tot_2d = k_model_2d + k_res_2d
fk_def_2d = k_model_2d / k_tot_2d
Lt = (k_tot_2d**1.5)/diss_2d
fk_ref_2d = (C_mu**-0.5) * (delta/Lt)**(2/3)

switch_i = np.argmax((fk_2d[1:-1,1:-1]<=0.4),axis=1) + 1
switch_wd = np.zeros(len(switch_i))
for i in range (nim1-1):
    switch_wd[i] = yp2d[i+1,switch_i[i]] - yp2d[i+1,0]

fig2,axes = plt.subplots(1,1,constrained_layout=True)
axes.plot(xp2d[1:-1,0],switch_wd,label='D-PANS model')
axes.legend(fontsize=16,loc='best')
axes.xaxis.set_minor_locator(AutoMinorLocator())
axes.yaxis.set_minor_locator(AutoMinorLocator())
axes.tick_params(labelsize='medium')
axes.grid(True, linestyle='-.')
axes.set_xlabel('x (m)')
axes.set_ylabel(r'$y-y_{wall}$ (m)')
fig2.suptitle(r"Location of the switch (from RANS to LES) wrt the lower wall",fontsize
    =22)

xx = np.array([0,0.65,0.8,0.9,1,1.1,1.2,1.3,2])
i1 = list(map(lambda m : (np.abs(m-xp2d[:,1])).argmin(),xx))

fig3,axes = plt.subplots(2,4,constrained_layout=True)
axes = axes.reshape(-1)
axes[0].plot(fk_2d[i1[0],1:],yp2d[i1[0],1:]-yp2d[i1[0],0],'b-o',ms=4,label=r'$f_{k}$ in
    D-PANS')
axes[1].plot(fk_2d[i1[1],1:],yp2d[i1[1],1:]-yp2d[i1[1],0],'b-o',ms=4,label=r'$f_{k}$ in
    D-PANS')
axes[2].plot(fk_2d[i1[2],1:],yp2d[i1[2],1:]-yp2d[i1[2],0],'b-o',ms=4,label=r'$f_{k}$ in
    D-PANS')
axes[3].plot(fk_2d[i1[3],1:],yp2d[i1[3],1:]-yp2d[i1[3],0],'b-o',ms=4,label=r'$f_{k}$ in
    D-PANS')
axes[4].plot(fk_2d[i1[4],1:],yp2d[i1[4],1:]-yp2d[i1[4],0],'b-o',ms=4,label=r'$f_{k}$ in
    D-PANS')
axes[5].plot(fk_2d[i1[5],1:],yp2d[i1[5],1:]-yp2d[i1[5],0],'b-o',ms=4,label=r'$f_{k}$ in
    D-PANS')
axes[6].plot(fk_2d[i1[6],1:],yp2d[i1[6],1:]-yp2d[i1[6],0],'b-o',ms=4,label=r'$f_{k}$ in
    D-PANS')
axes[7].plot(fk_2d[i1[7],1:],yp2d[i1[7],1:]-yp2d[i1[7],0],'b-o',ms=4,label=r'$f_{k}$ in
    D-PANS')
axes[0].plot(fk_def_2d[i1[0],1:],yp2d[i1[0],1:]-yp2d[i1[0],0],'r-o',ms=4,label=r'$f_{k}
    = k/k_{tot}$')
axes[1].plot(fk_def_2d[i1[1],1:],yp2d[i1[1],1:]-yp2d[i1[1],0],'r-o',ms=4,label=r'$f_{k}
    = k/k_{tot}$')
axes[2].plot(fk_def_2d[i1[2],1:],yp2d[i1[2],1:]-yp2d[i1[2],0],'r-o',ms=4,label=r'$f_{k}
    = k/k_{tot}$')
axes[3].plot(fk_def_2d[i1[3],1:],yp2d[i1[3],1:]-yp2d[i1[3],0],'r-o',ms=4,label=r'$f_{k}
    = k/k_{tot}$')
axes[4].plot(fk_def_2d[i1[4],1:],yp2d[i1[4],1:]-yp2d[i1[4],0],'r-o',ms=4,label=r'$f_{k}
    = k/k_{tot}$')
axes[5].plot(fk_def_2d[i1[5],1:],yp2d[i1[5],1:]-yp2d[i1[5],0],'r-o',ms=4,label=r'$f_{k}
```

```
    = k/k_{tot}$')
axes[6].plot(fk_def_2d[i1[6],1:],yp2d[i1[6],1:]-yp2d[i1[6],0],'r-o',ms=4,label=r'$f_{k}
    = k/k_{tot}$')
axes[7].plot(fk_def_2d[i1[7],1:],yp2d[i1[7],1:]-yp2d[i1[7],0],'r-o',ms=4,label=r'$f_{k}
    = k/k_{tot}$')
for ax,xloc in zip(axes,xx):
    ax.plot(0.4*np.ones(20),np.linspace(0,0.2,20)-yp2d[i1[0],0],'k--',label=r'$f_{k}$ =
    0.4')
    ax.xaxis.set_minor_locator(AutoMinorLocator())
    ax.yaxis.set_minor_locator(AutoMinorLocator())
    ax.legend(title='At X = '+str(xloc)+' m',title_fontsize=14,fontsize=14,loc='best')
    ax.tick_params(labelsize='medium')
    ax.grid(True, linestyle='-.')
    ax.set_ylim([0,0.2])
    if (ax==axes[0] or ax==axes[4]):
        ax.set_ylabel("$y-y_{wall}$")
    if (ax==axes[4] or ax==axes[5] or ax==axes[6] or ax==axes[7]):
        ax.set_xlabel("$f_{k}$")
fig3.suptitle(r"$f_{k}$ computed at various locations",fontsize=22)

fig4,axes = plt.subplots(2,4,constrained_layout=True)
axes = axes.reshape(-1)
axes[0].plot(fk_2d[i1[0],1:],yp2d[i1[0],1:]-yp2d[i1[0],0],'b-o',ms=4,label=r'$f_{k}$ in
    D-PANS')
axes[1].plot(fk_2d[i1[1],1:],yp2d[i1[1],1:]-yp2d[i1[1],0],'b-o',ms=4,label=r'$f_{k}$ in
    D-PANS')
axes[2].plot(fk_2d[i1[2],1:],yp2d[i1[2],1:]-yp2d[i1[2],0],'b-o',ms=4,label=r'$f_{k}$ in
    D-PANS')
axes[3].plot(fk_2d[i1[3],1:],yp2d[i1[3],1:]-yp2d[i1[3],0],'b-o',ms=4,label=r'$f_{k}$ in
    D-PANS')
axes[4].plot(fk_2d[i1[4],1:],yp2d[i1[4],1:]-yp2d[i1[4],0],'b-o',ms=4,label=r'$f_{k}$ in
    D-PANS')
axes[5].plot(fk_2d[i1[5],1:],yp2d[i1[5],1:]-yp2d[i1[5],0],'b-o',ms=4,label=r'$f_{k}$ in
    D-PANS')
axes[6].plot(fk_2d[i1[6],1:],yp2d[i1[6],1:]-yp2d[i1[6],0],'b-o',ms=4,label=r'$f_{k}$ in
    D-PANS')
axes[7].plot(fk_2d[i1[7],1:],yp2d[i1[7],1:]-yp2d[i1[7],0],'b-o',ms=4,label=r'$f_{k}$ in
    D-PANS')
axes[0].plot(fk_ref_2d[i1[0],1:],yp2d[i1[0],1:]-yp2d[i1[0],0],'r-o',ms=4,label=r'$f_{k}
    = C_{\mu}^{-0.5}(\frac{\Delta}{L_{t}})^{2/3}$')
axes[1].plot(fk_ref_2d[i1[1],1:],yp2d[i1[1],1:]-yp2d[i1[1],0],'r-o',ms=4,label=r'$f_{k}
    = C_{\mu}^{-0.5}(\frac{\Delta}{L_{t}})^{2/3}$')
axes[2].plot(fk_ref_2d[i1[2],1:],yp2d[i1[2],1:]-yp2d[i1[2],0],'r-o',ms=4,label=r'$f_{k}
    = C_{\mu}^{-0.5}(\frac{\Delta}{L_{t}})^{2/3}$')
axes[3].plot(fk_ref_2d[i1[3],1:],yp2d[i1[3],1:]-yp2d[i1[3],0],'r-o',ms=4,label=r'$f_{k}
    = C_{\mu}^{-0.5}(\frac{\Delta}{L_{t}})^{2/3}$')
axes[4].plot(fk_ref_2d[i1[4],1:],yp2d[i1[4],1:]-yp2d[i1[4],0],'r-o',ms=4,label=r'$f_{k}
    = C_{\mu}^{-0.5}(\frac{\Delta}{L_{t}})^{2/3}$')
axes[5].plot(fk_ref_2d[i1[5],1:],yp2d[i1[5],1:]-yp2d[i1[5],0],'r-o',ms=4,label=r'$f_{k}
    = C_{\mu}^{-0.5}(\frac{\Delta}{L_{t}})^{2/3}$')
axes[6].plot(fk_ref_2d[i1[6],1:],yp2d[i1[6],1:]-yp2d[i1[6],0],'r-o',ms=4,label=r'$f_{k}
    = C_{\mu}^{-0.5}(\frac{\Delta}{L_{t}})^{2/3}$')
axes[7].plot(fk_ref_2d[i1[7],1:],yp2d[i1[7],1:]-yp2d[i1[7],0],'r-o',ms=4,label=r'$f_{k}
    = C_{\mu}^{-0.5}(\frac{\Delta}{L_{t}})^{2/3}$')
for ax,xloc in zip(axes,xx):
    ax.plot(0.4*np.ones(20),np.linspace(0,0.2,20)-yp2d[i1[0],0],'k--',label=r'$f_{k}$ =
    0.4')
    ax.xaxis.set_minor_locator(AutoMinorLocator())
    ax.yaxis.set_minor_locator(AutoMinorLocator())
    ax.legend(title='At X = '+str(xloc)+' m',title_fontsize=14,fontsize=14,loc='best')
    ax.tick_params(labelsize='medium')
    ax.grid(True, linestyle='-.')
    ax.set_ylim([0,0.2])
    ax.set_xlim([0,1])
    if (ax==axes[0] or ax==axes[4]):
        ax.set_ylabel("$y-y_{wall}$")
```

```
    if (ax==axes[4] or ax==axes[5] or ax==axes[6] or ax==axes[7]):
        ax.set_xlabel("$f_{k}$")
fig4.suptitle(r"$f_{k}$ computed at various locations",fontsize=22)

# %% Assignment T5 - Location of Interface in DES and DDES
# SA-DES model
C_DES_SA = 0.65
delta_SA = C_DES_SA * np.maximum.reduce([dx,dy,dz*np.ones([ni,nj])])
d_tilda = np.minimum(yp2d-yp2d[:,0][:,None],delta_SA)
switch_i_SA = (yp2d-yp2d[:,0][:,None]<=delta_SA).argmin(axis=1)
switch_wd_SA = np.zeros(len(switch_i_SA))
for i in range (ni):
    switch_wd_SA[i] = yp2d[i,switch_i_SA[i]] - yp2d[i,0]

# SST-DES model
C_DES_SST = 0.61
beta_star = 0.09
omega_2d = (1/beta_star) * (diss_2d/k_model_2d)
Lt_SST = (1/beta_star) * ((k_model_2d**0.5)/omega_2d)
delta_SST = C_DES_SST * np.maximum.reduce([dx,dy,dz*np.ones([ni,nj])])
term1 = Lt_SST/delta_SST
F_SST = np.maximum(term1,1)
switch_i_SST = (term1<=1).argmin(axis=1)
switch_i_SST = np.where(switch_i_SST==0,109,switch_i_SST)
index_SST = np.where(switch_i_SST!=109)
switch_wd_SST = np.zeros(len(switch_i_SST))
switch_wd_SST2 = np.zeros(len(switch_i_SST))
for i in range (ni):
    switch_wd_SST[i] = yp2d[i,switch_i_SST[i]] - yp2d[i,0]
    switch_wd_SST2[i] = yp2d[i,switch_i_SST[i]]
switch_wd_SST1 = np.ma.masked_where(switch_wd_SST > 0.5, switch_wd_SST)
switch_wd_SST2 = np.ma.masked_where(switch_wd_SST2 > 0.5, switch_wd_SST2)

# DDES model
dist = yp2d[:]- yp2d[:,0][:,None]
term2 = 2*(Lt_SST[:,1:]/dist[:,1:])
term3 = (500*viscos)/(omega_2d[:,1:]*(dist[:,1:]**2))
eta = np.maximum(term2,term3)
F2 = np.tanh(eta**2)
term4 = term1[:,1:] * (1-F2)
switch_i_DDES = (term4<=1).argmin(axis=1)
switch_i_DDES = np.where(switch_i_DDES==0,109,switch_i_DDES)
index_DDES = np.where(switch_i_DDES!=109)
switch_wd_DDES = np.zeros(len(switch_i_SA))
switch_wd_DDES2 = np.zeros(len(switch_i_SA))
for i in range (ni):
    switch_wd_DDES[i] = yp2d[i,switch_i_DDES[i]] - yp2d[i,0]
    switch_wd_DDES2[i] = yp2d[i,switch_i_DDES[i]]
switch_wd_DDES1 = np.ma.masked_where(switch_wd_DDES > 0.5, switch_wd_DDES)
switch_wd_DDES2 = np.ma.masked_where(switch_wd_DDES2 > 0.5, switch_wd_DDES2)

# Length scales
ls_DES = C_DES_SA * np.maximum.reduce([dx,dy,dz*np.ones([ni,nj])])
ls_DPANS = (k_model_2d**1.5)/diss_2d

# finding boundary layer thickness
blt_i = np.zeros(ni)
blt_yc = np.zeros(ni)
blt_wd = np.zeros(ni)
for i in range(1,ni):
    blt_i[i] = np.max(np.where(((vis_2d[i,:]/viscos-1)<=1)==0))
    blt_yc[i] = yp2d[i,int(blt_i[i])]
    blt_wd[i] = blt_yc[i] - yp2d[i,0]

fig5,axes = plt.subplots(1,1,constrained_layout=True)
axes.plot(xp2d[1:-1,0],switch_wd,'b-o',ms=2,label='D-PANS model')
```

```
359 axes.plot(xp2d[1:-1,0],switch_wd_SA[1:-1],'r-o',ms=2,label='SA-DES model')
360 axes.plot(xp2d[1:-1,0],switch_wd_SST1[1:-1],'m-o',ms=2,label='SST-DES model')
361 axes.plot(xp2d[1:-1,0],switch_wd_DDES1[1:-1],'c-o',ms=2,label='DDES model')
362 axes.legend(fontsize=16,loc='best')
363 axes.xaxis.set_minor_locator(AutoMinorLocator())
364 axes.yaxis.set_minor_locator(AutoMinorLocator())
365 axes.tick_params(labelsize='medium')
366 axes.grid(True, linestyle='-.')
367 axes.set_xlabel('x (m)')
368 axes.set_ylabel(r'$y-y_{wall}$ (m)')
369 fig5.suptitle(r"Location of the switch (from RANS to LES)",fontsize=22)
370
371 fig5a,axes = plt.subplots(1,1,constrained_layout=True)
372 axes.plot(xp2d[1:-1,0],switch_wd+yp2d[1:-1,0],'b--',label='D-PANS model')
373 axes.plot(xp2d[1:-1,0],switch_wd_SA[1:-1]+yp2d[1:-1,0],'r--',label='SA-DES model')
374 axes.plot(xp2d[1:-1,0],switch_wd_SST2[1:-1],'m--',label='SST-DES model')
375 axes.plot(xp2d[1:-1,0],switch_wd_DDES2[1:-1],'c--',label='DDES model')
376 axes.plot(xp2d[:,0],yp2d[:,0],'k',lw=4,label='Lower boundary')
377 axes.legend(fontsize=16,loc='best')
378 axes.xaxis.set_minor_locator(AutoMinorLocator())
379 axes.yaxis.set_minor_locator(AutoMinorLocator())
380 axes.tick_params(labelsize='medium')
381 axes.grid(True, linestyle='-.')
382 axes.set_xlabel('x (m)')
383 axes.set_ylabel(r'$y$ (m)')
384 fig5a.suptitle(r"Location of the switch (from RANS to LES)",fontsize=22)
385
386 fig6,axes = plt.subplots(2,4,constrained_layout=True)
387 axes = axes.reshape(-1)
388 axes[0].plot(ls_DES[i1[1],1:],yp2d[i1[1],1:]-yp2d[i1[1],0],'b-o',ms=4,label=r'$l_{DES}$
    ')
389 axes[1].plot(ls_DES[i1[2],1:],yp2d[i1[2],1:]-yp2d[i1[2],0],'b-o',ms=4,label=r'$l_{DES}$
    ')
390 axes[2].plot(ls_DES[i1[3],1:],yp2d[i1[3],1:]-yp2d[i1[3],0],'b-o',ms=4,label=r'$l_{DES}$
    ')
391 axes[3].plot(ls_DES[i1[4],1:],yp2d[i1[4],1:]-yp2d[i1[4],0],'b-o',ms=4,label=r'$l_{DES}$
    ')
392 axes[4].plot(ls_DES[i1[5],1:],yp2d[i1[5],1:]-yp2d[i1[5],0],'b-o',ms=4,label=r'$l_{DES}$
    ')
393 axes[5].plot(ls_DES[i1[6],1:],yp2d[i1[6],1:]-yp2d[i1[6],0],'b-o',ms=4,label=r'$l_{DES}$
    ')
394 axes[6].plot(ls_DES[i1[7],1:],yp2d[i1[7],1:]-yp2d[i1[7],0],'b-o',ms=4,label=r'$l_{DES}$
    ')
395 axes[7].plot(ls_DES[i1[8],1:],yp2d[i1[8],1:]-yp2d[i1[8],0],'b-o',ms=4,label=r'$l_{DES}$
    ')
396 axes[0].plot(ls_DPANS[i1[1],1:],yp2d[i1[1],1:]-yp2d[i1[1],0],'r-o',ms=4,label=r'$l_{D-
    PANS}$')
397 axes[1].plot(ls_DPANS[i1[2],1:],yp2d[i1[2],1:]-yp2d[i1[2],0],'r-o',ms=4,label=r'$l_{D-
    PANS}$')
398 axes[2].plot(ls_DPANS[i1[3],1:],yp2d[i1[3],1:]-yp2d[i1[3],0],'r-o',ms=4,label=r'$l_{D-
    PANS}$')
399 axes[3].plot(ls_DPANS[i1[4],1:],yp2d[i1[4],1:]-yp2d[i1[4],0],'r-o',ms=4,label=r'$l_{D-
    PANS}$')
400 axes[4].plot(ls_DPANS[i1[5],1:],yp2d[i1[5],1:]-yp2d[i1[5],0],'r-o',ms=4,label=r'$l_{D-
    PANS}$')
401 axes[5].plot(ls_DPANS[i1[6],1:],yp2d[i1[6],1:]-yp2d[i1[6],0],'r-o',ms=4,label=r'$l_{D-
    PANS}$')
402 axes[6].plot(ls_DPANS[i1[7],1:],yp2d[i1[7],1:]-yp2d[i1[7],0],'r-o',ms=4,label=r'$l_{D-
    PANS}$')
403 axes[7].plot(ls_DPANS[i1[8],1:],yp2d[i1[8],1:]-yp2d[i1[8],0],'r-o',ms=4,label=r'$l_{D-
    PANS}$')
404 axes[0].plot(np.linspace(np.min(ls_DPANS[i1[1],1:]),np.max(ls_DES[i1[1],1:]),10),blt_wd[
    i1[1]]*np.ones(10),'k--',ms=4,label='BL thickness')
405 axes[1].plot(np.linspace(np.min(ls_DPANS[i1[2],1:]),np.max(ls_DES[i1[2],1:]),10),blt_wd[
    i1[2]]*np.ones(10),'k--',ms=4,label='BL thickness')
406 axes[2].plot(np.linspace(np.min(ls_DPANS[i1[3],1:]),np.max(ls_DES[i1[3],1:]),10),blt_wd[
```

```
    i1[3]]*np.ones(10),'k--',ms=4,label='BL thickness')
axes[3].plot(np.linspace(np.min(ls_DPANS[i1[4],1:]),np.max(ls_DES[i1[4],1:]),10),blt_wd[
    i1[4]]*np.ones(10),'k--',ms=4,label='BL thickness')
axes[4].plot(np.linspace(np.min(ls_DPANS[i1[5],1:]),np.max(ls_DES[i1[5],1:]),10),blt_wd[
    i1[5]]*np.ones(10),'k--',ms=4,label='BL thickness')
axes[5].plot(np.linspace(np.min(ls_DPANS[i1[6],1:]),np.max(ls_DES[i1[6],1:]),10),blt_wd[
    i1[6]]*np.ones(10),'k--',ms=4,label='BL thickness')
axes[6].plot(np.linspace(np.min(ls_DPANS[i1[7],1:]),np.max(ls_DES[i1[7],1:]),10),blt_wd[
    i1[7]]*np.ones(10),'k--',ms=4,label='BL thickness')
axes[7].plot(np.linspace(np.min(ls_DPANS[i1[8],1:]),np.max(ls_DES[i1[8],1:]),10),blt_wd[
    i1[8]]*np.ones(10),'k--',ms=4,label='BL thickness')
for ax,xloc in zip(axes,xx[1:]):
    ax.xaxis.set_minor_locator(AutoMinorLocator())
    ax.yaxis.set_minor_locator(AutoMinorLocator())
    ax.legend(title='At X = '+str(xloc)+' m',title_fontsize=14,fontsize=14,loc='best')
    ax.tick_params(labelsize='medium')
    ax.grid(True, linestyle='-.')
    ax.set_xlim([0,0.015])
    ax.set_ylim([0,0.2])
    if ax==axes[7]:
        ax.set_xlim([0,0.03])
        ax.set_ylim([0,0.3])
    if (ax==axes[0] or ax==axes[4]):
        ax.set_ylabel("$y-y_{wall}$")
    if (ax==axes[4] or ax==axes[5] or ax==axes[6] or ax==axes[7]):
        ax.set_xlabel("Length scale")
fig6.suptitle(r"Length scales computed at various locations",fontsize=22)
```

For Assignment 2b : Section 2.7 to Section 2.8

```
% Diffuser, lada
clc
clear
close all

nk=34;
re =9.36e+5;
viscos =1/re;

% Computational domain
load xy_hump_fine.dat
x1=xy_hump_fine(:,1);
y1=xy_hump_fine(:,2);

nim1=x1(1);
njm1=y1(1);

ni=nim1+1;
nj=njm1+1;

x=x1(2:end);
y=y1(2:end);

x_2d=reshape(x,nim1,njm1);
y_2d=reshape(y,nim1,njm1);

x_2d(:,nj)=x_2d(:,nj-1);
y_2d(:,nj)=y_2d(:,nj-1);
x_2d(ni,:)=x_2d(ni-1,:);
y_2d(ni,:)=y_2d(ni-1,:);

% compute cell centers
xp2d=zeros(ni,nj);
yp2d=zeros(ni,nj);

for jj=1:nj
for find_loc_i=1:ni
    im1=max(find_loc_i-1,1);
    jm1=max(jj-1,1);
    c=min(find_loc_i,nim1);
    j=min(jj,njm1);
   xp2d(find_loc_i,jj)=0.25*(x_2d(c,j)+x_2d(im1,j)+x_2d(c,jm1)+x_2d(im1,jm1));
   yp2d(find_loc_i,jj)=0.25*(y_2d(c,j)+y_2d(im1,j)+y_2d(c,jm1)+y_2d(im1,jm1));
end
end

zmax=0.2;
dz=zmax/(nk-2);
z=0:dz:zmax;

% Loading of all data files as recommended
% file 1
load u1_pans_iddes.mat
u1=u1_pans_iddes;
temp=reshape(u1,nk,nj,ni);
u3d1=permute(temp,[3 2 1]);

load v1_pans_iddes.mat
v1=v1_pans_iddes;
temp=reshape(v1,nk,nj,ni);
v3d1=permute(temp,[3 2 1]);

load w1_pans_iddes.mat
w1=w1_pans_iddes;
```

```
temp=reshape(w1,nk,nj,ni);
w3d1=permute(temp,[3 2 1]);

% file 2
load u2_pans_iddes.mat
u2=u2_pans_iddes;
temp=reshape(u2,nk,nj,ni);
u3d2=permute(temp,[3 2 1]);

load v2_pans_iddes.mat
v2=v2_pans_iddes;
temp=reshape(v2,nk,nj,ni);
v3d2=permute(temp,[3 2 1]);

load w2_pans_iddes.mat
w2=w2_pans_iddes;
temp=reshape(w2,nk,nj,ni);
w3d2=permute(temp,[3 2 1]);

load u3_pans_iddes.mat
u3=u3_pans_iddes;
temp=reshape(u3,nk,nj,ni);
u3d3=permute(temp,[3 2 1]);

load v3_pans_iddes.mat
v3=v3_pans_iddes;
temp=reshape(v3,nk,nj,ni);
v3d3=permute(temp,[3 2 1]);

load w3_pans_iddes.mat
w3=w3_pans_iddes;
temp=reshape(w3,nk,nj,ni);
w3d3=permute(temp,[3 2 1]);

% file 4
load u4_pans_iddes.mat
u4=u4_pans_iddes;
temp=reshape(u4,nk,nj,ni);
u3d4=permute(temp,[3 2 1]);

load v4_pans_iddes.mat
v4=v4_pans_iddes;
temp=reshape(v4,nk,nj,ni);
v3d4=permute(temp,[3 2 1]);

load w4_pans_iddes.mat
w4=w4_pans_iddes;
temp=reshape(w4,nk,nj,ni);
w3d4=permute(temp,[3 2 1]);

% file 5
load u5_pans_iddes.mat
u5=u5_pans_iddes;
temp=reshape(u5,nk,nj,ni);
u3d5=permute(temp,[3 2 1]);

load v5_pans_iddes.mat
v5=v5_pans_iddes;
temp=reshape(v5,nk,nj,ni);
v3d5=permute(temp,[3 2 1]);

load w5_pans_iddes.mat
w5=w5_pans_iddes;
temp=reshape(w5,nk,nj,ni);
w3d5=permute(temp,[3 2 1]);

```

```
% file 6
load u6_pans_iddes.mat
u6=u6_pans_iddes;
temp=reshape(u6,nk,nj,ni);
u3d6=permute(temp,[3 2 1]);

load v6_pans_iddes.mat
v6=v6_pans_iddes;
temp=reshape(v6,nk,nj,ni);
v3d6=permute(temp,[3 2 1]);

load w6_pans_iddes.mat
w6=w6_pans_iddes;
temp=reshape(w6,nk,nj,ni);
w3d6=permute(temp,[3 2 1]);

% file 7
load u7_pans_iddes.mat
u7=u7_pans_iddes;
temp=reshape(u7,nk,nj,ni);
u3d7=permute(temp,[3 2 1]);

load v7_pans_iddes.mat
v7=v7_pans_iddes;
temp=reshape(v7,nk,nj,ni);
v3d7=permute(temp,[3 2 1]);

load w7_pans_iddes.mat
w7=w7_pans_iddes;
temp=reshape(w7,nk,nj,ni);
w3d7=permute(temp,[3 2 1]);

% file 8
load u8_pans_iddes.mat
u8=u8_pans_iddes;
temp=reshape(u8,nk,nj,ni);
u3d8=permute(temp,[3 2 1]);

load v8_pans_iddes.mat
v8=v8_pans_iddes;
temp=reshape(v8,nk,nj,ni);
v3d8=permute(temp,[3 2 1]);

load w8_pans_iddes.mat
w8=w8_pans_iddes;
temp=reshape(w8,nk,nj,ni);
w3d8=permute(temp,[3 2 1]);

% % merge 2 files. This means than new nk = 2*nk
u3d=cat(3,u3d1,u3d2,u3d3,u3d4,u3d5,u3d6,u3d7,u3d8);
v3d=cat(3,v3d1,v3d2,v3d3,v3d4,v3d5,v3d6,v3d7,v3d8);
w3d=cat(3,w3d1,w3d2,w3d3,w3d4,w3d5,w3d6,w3d7,w3d8);
clear u3d1 u3d2 v3d1 v3d2 w3d1 w3d2

% merge 8 files
% u3d=cat(3,u3d1,u3d2,u3d3,u3d4,u3d5,u3d6,u3d7,u3d8);
% v3d=cat(3,v3d1,v3d2,v3d3,v3d4,v3d5,v3d6,v3d7,v3d8);
% w3d=cat(3,w3d1,w3d2,w3d3,w3d4,w3d5,w3d6,w3d7,w3d8);
clear u3d1 u3d2 u3d3 u3d4 u3d5 u3d6 u3d7 u3d8
clear v3d1 v3d2 v3d3 v3d4 v3d5 v3d6 v3d7 v3d8
clear w3d1 w3d2 w3d3 w3d4 w3d5 w3d6 w3d7 w3d8
clear u1 u2 u3 u4 u5 u6 u7 u8
clear v1 v2 v3 v4 v5 v6 v7 v8
clear w1 w2 w3 w4 w5 w6 w7 w8
clear u1_pans_iddes u2_pans_iddes u3_pans_iddes u4_pans_iddes u5_pans_iddes
    u6_pans_iddes u7_pans_iddes u8_pans_iddes
```

```
clear v1_pans_iddes v2_pans_iddes v3_pans_iddes v4_pans_iddes v5_pans_iddes
    v6_pans_iddes v7_pans_iddes v8_pans_iddes
clear w1_pans_iddes w2_pans_iddes w3_pans_iddes w4_pans_iddes w5_pans_iddes
    w6_pans_iddes w7_pans_iddes w8_pans_iddes

nk=size(u3d,3);

% u3d(ni,nj,nk), v3d, w3d are loaded

% i, j, and k correspond roughly to x, y, z (recall that the mesh is curvilinear in x &
    y)

u_2d=mean(u3d,3);
v_2d=mean(v3d,3);

% compute x, y gradients at grid plane k
for k=2:nk-1
   [dudx_2d,dudy_2d] = dphidx_dy(x_2d,y_2d,u3d(:,:,k),ni,nj);
   [dvdx_2d,dvdy_2d] = dphidx_dy(x_2d,y_2d,v3d(:,:,k),ni,nj);
   [dwdx_2d,dwdy_2d] = dphidx_dy(x_2d,y_2d,w3d(:,:,k),ni,nj);

   [d2udx2_2d d2udxdy_2d]=dphidx_dy(x_2d,y_2d,dudx_2d,ni,nj);
   [dummy d2udy2_2d]=dphidx_dy(x_2d,y_2d,dudy_2d,ni,nj);

% corrected
   [d2vdx2_2d d2vdxdy_2d]=dphidx_dy(x_2d,y_2d,dvdx_2d,ni,nj);
   [dummy d2vdy2_2d]=dphidx_dy(x_2d,y_2d,dvdy_2d,ni,nj);

% corrected
   [d2wdx2_2d d2wdxdy_2d]=dphidx_dy(x_2d,y_2d,dwdx_2d,ni,nj);
   [dummy d2wdy2_2d]=dphidx_dy(x_2d,y_2d,dwdy_2d,ni,nj);

   dudx(:,:,k)=dudx_2d;
   dudy(:,:,k)=dudy_2d;

   dvdx(:,:,k)=dvdx_2d;
   dvdy(:,:,k)=dvdy_2d;

   dwdx(:,:,k)=dwdx_2d;
   dwdy(:,:,k)=dwdy_2d;

   d2udx2(:,:,k)=d2udx2_2d;
   d2udy2(:,:,k)=d2udy2_2d;
   d2udxdy(:,:,k)=d2udxdy_2d;

   d2vdx2(:,:,k)=d2vdx2_2d;
   d2vdy2(:,:,k)=d2vdy2_2d;
   d2vdxdy(:,:,k)=d2vdxdy_2d;

   d2wdx2(:,:,k)=d2wdx2_2d;
   d2wdy2(:,:,k)=d2wdy2_2d;
   d2wdxdy(:,:,k)=d2wdxdy_2d;

end


[dummy dummy dudz]=gradient(u3d,1,1,dz);
[dummy dummy dvdz]=gradient(v3d,1,1,dz);
[dummy dummy dwdz]=gradient(w3d,1,1,dz);

dudz=dudz(:,:,1:nk-1);
dvdz=dvdz(:,:,1:nk-1);
dwdz=dwdz(:,:,1:nk-1);

[dummy dummy d2udxdz]=gradient(dudx,1,1,dz);
[dummy dummy d2udydz]=gradient(dudy,1,1,dz);
```

```
[dummy dummy d2udz2]=gradient(dudz,1,1,dz);

[dummy dummy d2vdxdz]=gradient(dvdx,1,1,dz);
[dummy dummy d2vdydz]=gradient(dvdy,1,1,dz);
[dummy dummy d2vdz2]=gradient(dvdz,1,1,dz);

[dummy dummy d2wdxdz]=gradient(dwdx,1,1,dz);
[dummy dummy d2wdydz]=gradient(dwdy,1,1,dz);
[dummy dummy d2wdz2]=gradient(dwdz,1,1,dz);

s11=dudx;
s12=0.5*(dudy+dvdx);
s13=0.5*(dudz+dwdx);
s21=s12;
s22=dvdy;
s23=0.5*(dvdz+dwdy);
s31=s13;
s32=s23;
s33=dwdz;

ss=(2*(s11.^2+s12.^2+s13.^2+s21.^2+s22.^2+s23.^2+s31.^2+s32.^2+s33.^2).^0.5);

u_1_term=(d2udx2+d2udy2+d2udz2).^2;
v_1_term=(d2vdx2+d2vdy2+d2vdz2).^2;
w_1_term=(d2wdx2+d2wdy2+d2wdz2).^2;

u_2_term=(d2udx2+d2udy2+d2udz2 + 2*d2udxdy + 2*d2udxdz + 2*d2udydz).^2;
v_2_term=(d2vdx2+d2vdy2+d2vdz2 + 2*d2vdxdy + 2*d2vdxdz + 2*d2vdydz).^2;
w_2_term=(d2wdx2+d2wdy2+d2wdz2 + 2*d2wdxdy + 2*d2wdxdz + 2*d2wdydz).^2;

termu_b=d2udx2.^2+d2udy2.^2+d2udz2.^2;
termv_b=d2vdx2.^2+d2vdy2.^2+d2vdz2.^2;
termw_b=d2wdx2.^2+d2wdy2.^2+d2wdz2.^2;

usd_1=(u_1_term+v_1_term+w_1_term).^0.5;
usd_2=(u_2_term+v_2_term+w_2_term).^0.5;
usd_b=(termu_b+termv_b+termw_b).^0.5;

L_vk3d_1=0.4*ss./usd_1;
L_vk3d_2=0.4*ss./usd_2;
L_vk3d_b=0.4*ss./usd_b;

L_vk3d_spanwise_1=mean(L_vk3d_1,3);
L_vk3d_spanwise_2=mean(L_vk3d_2,3);

load vectz_fine.dat
vectz=vectz_fine;
ntstep=vectz(1);
ni=vectz(2);
nj=vectz(3);
nk=vectz(4);
n=length(vectz);

nn=14;
nst=3;
iu=nst+2:nn:n;
iv=nst+3:nn:n;
ifk=nst+4:nn:n;
iuu=nst+5:nn:n;
ivv=nst+6:nn:n;
iww=nst+7:nn:n;
iuv=nst+8:nn:n;
ip=nst+9:nn:n;
ik=nst+10:nn:n;
ivis=nst+11:nn:n;
idiss=nst+12:nn:n;
```

```
325
326 u=vectz(iu)/ntstep;
327 v=vectz(iv)/ntstep;
328 fk=vectz(ifk)/ntstep;
329 uu=vectz(iuu)/ntstep;
330 vv=vectz(ivv)/ntstep;
331 ww=vectz(iww)/ntstep;
332 uv=vectz(iuv)/ntstep;
333 p=vectz(ip)/ntstep;
334 k_model=vectz(ik)/ntstep;
335 vis=vectz(ivis)/ntstep;
336 diss=vectz(idiss)/ntstep;
337
338 uu=uu-u.^2;
339 vv=vv-v.^2;
340 uv=uv-u.*v;
341
342 p_2d=reshape(p,nj,ni)';
343 uu_2d=reshape(uu,nj,ni)';
344 vv_2d=reshape(vv,nj,ni)';
345 fk_2d=reshape(fk,nj,ni)';
346 ww_2d=reshape(ww,nj,ni)';
347 uv_2d=reshape(uv,nj,ni)';
348 k_model_2d=reshape(k_model,nj,ni)';
349 diss_2d=reshape(diss,nj,ni)';
350 vis_2d=reshape(vis,nj,ni)';
351 u_2d=reshape(u,nj,ni)';
352 v_2d=reshape(v,nj,ni)';
353
354 k_resolved= 0.5*(uu+vv+ww);
355 k_resolved_2d=reshape(k_resolved,nj,ni)';
356
357 fk_2d(:,nj)=fk_2d(:,nj-1);
358
359 [dudx_m,dudy_m] = dphidx_dy(x_2d,y_2d,u_2d,ni,nj);
360 [dvdx_m,dvdy_m] = dphidx_dy(x_2d,y_2d,v_2d,ni,nj);
361
362 [d2udx2_m d2udxdy_m]=dphidx_dy(x_2d,y_2d,dudx_m,ni,nj);
363 [dummy d2udy2_m]=dphidx_dy(x_2d,y_2d,dudy_m,ni,nj);
364
365 [d2vdx2_m d2vdxdy_m]=dphidx_dy(x_2d,y_2d,dvdx_m,ni,nj);
366 [dummy d2vdy2_m]=dphidx_dy(x_2d,y_2d,dvdy_m,ni,nj);
367
368 s_11_m=dudx_m;
369 s_12_m=0.5*(dudy_m+dvdx_m);
370
371 s_21_m=s_12_m;
372 s_22_m=dvdy_m;
373
374 s_s_m=(2*(s_11_m.^2+s_12_m.^2+s_21_m.^2+s_22_m.^2).^0.5);
375
376 u_m_1=(d2udx2_m+d2udy2_m).^2;
377 v_m_1=(d2vdx2_m+d2vdy2_m).^2;
378
379 u_m_2=(d2udx2_m+2*d2udxdy_m+d2udy2_m).^2;
380 v_m_2=(d2vdx2_m+2*d2vdxdy_m+d2vdy2_m).^2;
381
382 usd_m_1=(u_m_1+v_m_1).^0.5;
383 usd_m_2=(u_m_2+v_m_2).^0.5;
384
385 L_vk_3d_m_1=0.4*s_s_m./usd_m_1;
386 L_vk_3d_m_2=0.4*s_s_m./usd_m_2;
387
388 dx=diff(xp2d);
389 dx=repmat(dx,[1 1 nk-1]);
390 dx(ni,:)=dx(ni-1,:);
```

```

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

dx2=dx.^2;
dy2=dy.^2;
dz2=dz^2;

C_DES = 0.67;
l_s_PANS = (k_model_2d.^1.5)./diss_2d; % length scale PANS
l_s_DES = C_DES*del; % length scale DES

dx_2d = mean(dx,3);
dy_2d = mean(dy,3);
dz = 0.2/32;
del = max(max(dx_2d, dy_2d),dz);

figure(600)
subplot(2,2,1)
exp_locx=0.65;
find_loc_i = find(xp2d(:,1) < exp_locx);
c=find_loc_i(end);
plot(yp2d(c,:),L_vk3d_spanwise_1(c,:),'r','linewidth',1.5)
hold on
plot(yp2d(c,:),L_vk3d_spanwise_2(c,:),'b','linewidth',1.5)
plot(yp2d(c,:),L_vk_3d_m_1(c,:),'k','linewidth',1.5)
plot(yp2d(c,:),L_vk_3d_m_2(c,:),'g','linewidth',1.5)
plot(yp2d(c,:),l_s_PANS(c,:),'c','linewidth',1.5)
plot(yp2d(c,:),l_s_DES(c,:),'m','linewidth',1.5)
ylabel('Length scale','fontsize',15)
xlabel('y','fontsize',15)
title('Location x=0.65')
h1=gca;
set(h1,'fontsize',15)
grid on
hold off

subplot(2,2,2)
exp_locx=0.8;
find_loc_i = find(xp2d(:,1) < exp_locx);
c=find_loc_i(end);
plot(yp2d(c,:),L_vk3d_spanwise_1(c,:),'r','linewidth',1.5)
hold on
plot(yp2d(c,:),L_vk3d_spanwise_2(c,:),'b','linewidth',1.5)
plot(yp2d(c,:),L_vk_3d_m_1(c,:),'k','linewidth',1.5)
plot(yp2d(c,:),L_vk_3d_m_2(c,:),'g','linewidth',1.5)
plot(yp2d(c,:),l_s_PANS(c,:),'c','linewidth',1.5)
plot(yp2d(c,:),l_s_DES(c,:),'m','linewidth',1.5)
ylabel('Length scale','fontsize',15)
xlabel('y','fontsize',15)
title('Location x=0.8')
h1=gca;
set(h1,'fontsize',15)
grid on
hold off

subplot(2,2,3)
exp_locx=0.9;
find_loc_i = find(xp2d(:,1) < exp_locx);
c=find_loc_i(end);
plot(yp2d(c,:),L_vk3d_spanwise_1(c,:),'r','linewidth',1.5)
hold on
plot(yp2d(c,:),L_vk3d_spanwise_2(c,:),'b','linewidth',1.5)
plot(yp2d(c,:),L_vk_3d_m_1(c,:),'k','linewidth',1.5)
plot(yp2d(c,:),L_vk_3d_m_2(c,:),'g','linewidth',1.5)
```

```
plot(yp2d(c,:),l_s_PANS(c,:),'c','linewidth',1.5)
plot(yp2d(c,:),l_s_DES(c,:),'m','linewidth',1.5)
ylabel('Length scale','fontsize',15)
xlabel('y','fontsize',15)
title('Location x=0.9')
h1=gca;
set(h1,'fontsize',15)
grid on
hold off

subplot(2,2,4)
exp_locx=1;
find_loc_i = find(xp2d(:,1) < exp_locx);
c=find_loc_i(end);
plot(yp2d(c,:),L_vk3d_spanwise_1(c,:),'r','linewidth',1.5)
hold on
plot(yp2d(c,:),L_vk3d_spanwise_2(c,:),'b','linewidth',1.5)
plot(yp2d(c,:),L_vk_3d_m_1(c,:),'k','linewidth',1.5)
plot(yp2d(c,:),L_vk_3d_m_2(c,:),'g','linewidth',1.5)
plot(yp2d(c,:),l_s_PANS(c,:),'c','linewidth',1.5)
plot(yp2d(c,:),l_s_DES(c,:),'m','linewidth',1.5)
ylabel('Length scale','fontsize',15)
xlabel('y','fontsize',15)
title('Location x=1')
h1=gca;
set(h1,'fontsize',15)
grid on
hold off

figure(601)
subplot(2,2,1)
exp_locx=1.1;
find_loc_i = find(xp2d(:,1) < exp_locx);
c=find_loc_i(end);
plot(yp2d(c,:),L_vk3d_spanwise_1(c,:),'r','linewidth',1.5)
hold on
plot(yp2d(c,:),L_vk3d_spanwise_2(c,:),'b','linewidth',1.5)
plot(yp2d(c,:),L_vk_3d_m_1(c,:),'k','linewidth',1.5)
plot(yp2d(c,:),L_vk_3d_m_2(c,:),'g','linewidth',1.5)
plot(yp2d(c,:),l_s_PANS(c,:),'c','linewidth',1.5)
plot(yp2d(c,:),l_s_DES(c,:),'m','linewidth',1.5)
ylabel('Length scale','fontsize',15)
xlabel('y','fontsize',15)
title('Location x=1.1')
h1=gca;
set(h1,'fontsize',15)
grid on
hold off

subplot(2,2,2)
exp_locx=1.2;
find_loc_i = find(xp2d(:,1) < exp_locx);
c=find_loc_i(end);
plot(yp2d(c,:),L_vk3d_spanwise_1(c,:),'r','linewidth',1.5)
hold on
plot(yp2d(c,:),L_vk3d_spanwise_2(c,:),'b','linewidth',1.5)
plot(yp2d(c,:),L_vk_3d_m_1(c,:),'k','linewidth',1.5)
plot(yp2d(c,:),L_vk_3d_m_2(c,:),'g','linewidth',1.5)
plot(yp2d(c,:),l_s_PANS(c,:),'c','linewidth',1.5)
plot(yp2d(c,:),l_s_DES(c,:),'m','linewidth',1.5)
ylabel('Length scale','fontsize',15)
xlabel('y','fontsize',15)
title('Location x=1.2')
h1=gca;
set(h1,'fontsize',15)
grid on
```

```
hold off

subplot(2,2,3)
exp_locx=1.3;
find_loc_i = find(xp2d(:,1) < exp_locx);
c=find_loc_i(end);
plot(yp2d(c,:),L_vk3d_spanwise_1(c,:),'r','linewidth',1.5)
hold on
plot(yp2d(c,:),L_vk3d_spanwise_2(c,:),'b','linewidth',1.5)
plot(yp2d(c,:),L_vk_3d_m_1(c,:),'k','linewidth',1.5)
plot(yp2d(c,:),L_vk_3d_m_2(c,:),'g','linewidth',1.5)
plot(yp2d(c,:),l_s_PANS(c,:),'c','linewidth',1.5)
plot(yp2d(c,:),l_s_DES(c,:),'m','linewidth',1.5)
ylabel('Length scale','fontsize',15)
xlabel('y','fontsize',15)
title('Location x=1.3')
legend('$L_{vk, 3D,s, type1}$','$L_{vk, 3D,s, type2}$','$L_{vk, 3D,mean, type1}$','
    fontsize',25,'Orientation','horizontal','location','southoutside','interpreter','
    latex')
h1=gca;
set(h1,'fontsize',15)
grid on
hold off

subplot(2,2,4)
exp_locx=1.4;
find_loc_i = find(xp2d(:,1) < exp_locx);
c=find_loc_i(end);
plot(yp2d(c,:),L_vk_3d_m_2(c,:),'g','linewidth',1.5)
hold on
plot(yp2d(c,:),l_s_PANS(c,:),'c','linewidth',1.5)
plot(yp2d(c,:),l_s_DES(c,:),'m','linewidth',1.5)
plot(yp2d(c,:),L_vk3d_spanwise_1(c,:),'r','linewidth',1.5)
plot(yp2d(c,:),L_vk3d_spanwise_2(c,:),'b','linewidth',1.5)
plot(yp2d(c,:),L_vk_3d_m_1(c,:),'k','linewidth',1.5)
ylabel('Length scale','fontsize',15)
xlabel('y','fontsize',15)
title('Location x=1.4','fontsize',12)
legend('$L_{vk, 3D,mean, type2 }$','$l_{PANS}$','$l_{DES}$','fontsize',25,'Orientation',
    'horizontal','location','southoutside','interpreter','latex')
h1=gca;
set(h1,'fontsize',15)
grid on
hold off

l_RANS = k_resolved_2d.^(1.5)./diss_2d;


figure(602)
subplot(2,2,1)
exp_locx=0.65;
find_loc_i= find(xp2d(:,1) < exp_locx);
c=find_loc_i(end);
plot(yp2d(c,:),L_vk_3d_m_1(c,:),'r','linewidth',1.5)
hold on
plot(yp2d(c,:),L_vk_3d_m_2(c,:),'b','linewidth',1.5)
plot(yp2d(c,:),l_RANS(c,:),'k','linewidth',1.5)
axis([0.1 0.25 0 0.3])
ylabel('Length scale','fontsize',15)
xlabel('y','fontsize',15)
title('Location x=0.65','fontsize',12)
h1=gca;
set(h1,'fontsize',15)
grid on
hold off

```

```
subplot(2,2,2)
exp_locx=0.8;
find_loc_i= find(xp2d(:,1) < exp_locx);
c=find_loc_i(end);
plot(yp2d(c,:),L_vk_3d_m_1(c,:),'r','linewidth',1.5)
hold on
plot(yp2d(c,:),L_vk_3d_m_2(c,:),'b','linewidth',1.5)
plot(yp2d(c,:),l_RANS(c,:),'k','linewidth',1.5)
axis([0.1 0.25 0 0.3])
ylabel('Length scale','fontsize',15)
xlabel('y','fontsize',15)
title('Location x=0.8','fontsize',12)
h1=gca;
set(h1,'fontsize',15)
grid on
hold off

subplot(2,2,3)
exp_locx=0.9;
find_loc_i= find(xp2d(:,1) < exp_locx);
c=find_loc_i(end);
plot(yp2d(c,:),L_vk_3d_m_1(c,:),'r','linewidth',1.5)
hold on
plot(yp2d(c,:),L_vk_3d_m_2(c,:),'b','linewidth',1.5)
plot(yp2d(c,:),l_RANS(c,:),'k','linewidth',1.5)
axis([0.1 0.25 0 0.3])
ylabel('Length scale','fontsize',15)
xlabel('y','fontsize',15)
title('Location x=0.9','fontsize',12)
h1=gca;
set(h1,'fontsize',15)
grid on
hold off

subplot(2,2,4)
exp_locx=1.0;
find_loc_i= find(xp2d(:,1) < exp_locx);
c=find_loc_i(end);
plot(yp2d(c,:),L_vk_3d_m_1(c,:),'r','linewidth',1.5)
hold on
plot(yp2d(c,:),L_vk_3d_m_2(c,:),'b','linewidth',1.5)
plot(yp2d(c,:),l_RANS(c,:),'k','linewidth',1.5)
axis([0.1 0.25 0 0.3])
ylabel('Length scale','fontsize',15)
xlabel('y','fontsize',15)
title('Location x=1.0','fontsize',12)
h1=gca;
set(h1,'fontsize',15)
grid on
hold off

figure(603)
subplot(2,2,1)
exp_locx=1.1;
find_loc_i= find(xp2d(:,1) < exp_locx);
c=find_loc_i(end);
plot(yp2d(c,:),L_vk_3d_m_1(c,:),'r','linewidth',1.5)
hold on
plot(yp2d(c,:),L_vk_3d_m_2(c,:),'b','linewidth',1.5)
plot(yp2d(c,:),l_RANS(c,:),'k','linewidth',1.5)
axis([0.1 0.25 0 0.3])
ylabel('Length scale','fontsize',15)
xlabel('y','fontsize',15)
title('Location x=1.1','fontsize',12)
h1=gca;
set(h1,'fontsize',15)
```

```
grid on
hold off

subplot(2,2,2)
exp_locx=1.2;
find_loc_i= find(xp2d(:,1) < exp_locx);
c=find_loc_i(end);
plot(yp2d(c,:),L_vk_3d_m_1(c,:),'r','linewidth',1.5)
hold on
plot(yp2d(c,:),L_vk_3d_m_2(c,:),'b','linewidth',1.5)
plot(yp2d(c,:),l_RANS(c,:),'k','linewidth',1.5)
axis([0.1 0.25 0 0.3])
ylabel('Length scale','fontsize',15)
xlabel('y','fontsize',15)
title('Location x=1.2','fontsize',12)
h1=gca;
set(h1,'fontsize',15)
grid on
hold off

subplot(2,2,3)
exp_locx=1.3;
find_loc_i= find(xp2d(:,1) < exp_locx);
c=find_loc_i(end);
plot(yp2d(c,:),L_vk_3d_m_1(c,:),'r','linewidth',1.5)
hold on
plot(yp2d(c,:),L_vk_3d_m_2(c,:),'b','linewidth',1.5)
plot(yp2d(c,:),l_RANS(c,:),'k','linewidth',1.5)
axis([0.1 0.25 0 0.3])
ylabel('Length scale','fontsize',15)
xlabel('y','fontsize',15)
title('Location x=1.3','fontsize',12)
legend('$L_{vk, 3D,mean, type1}$','$L_{vk, 3D,mean, type2}$','$l_{RANS}$','fontsize',25,
    'Orientation','horizontal','interpreter','latex','location','southoutside')
h1=gca;
set(h1,'fontsize',15)
grid on
hold off

subplot(2,2,4)
exp_locx=2.0;
find_loc_i= find(xp2d(:,1) < exp_locx);
c=find_loc_i(end);
plot(yp2d(c,:),L_vk_3d_m_1(c,:),'r','linewidth',1.5)
hold on
plot(yp2d(c,:),L_vk_3d_m_2(c,:),'b','linewidth',1.5)
plot(yp2d(c,:),l_RANS(c,:),'k','linewidth',1.5)
axis([0.1 0.25 0 0.3])
ylabel('Length scale','fontsize',15)
xlabel('y','fontsize',15)
title('Location x=2.0','fontsize',12)
legend('$L_{vk, 3D,mean, type1}$','$L_{vk, 3D,mean, type2}$','$l_{RANS}$','fontsize',25,
    'Orientation','horizontal','interpreter','latex','location','southoutside')
h1=gca;
set(h1,'fontsize',15)
grid on
hold off
```