

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	1
1.1 Time history	1
1.2 Mean velocity profile	4
1.3 Modeled and Resolved turbulent shear stresses (S3 and S5)	5
1.4 Turbulent kinetic energy	7
1.5 Location of interfaces in DES and DDES	8
1.6 SAS Turbulent length scales	9
1.7 Estimation of resolution	11
2 Assignment 2b : Recirculating flow - PANS, DES, DDES and SAS	12
2.1 Discretization schemes	12
2.2 Modeled turbulent shear stress	17
2.3 Turbulent viscosity	19
2.4 Location of interface	21
2.5 Location of interface in DES and DDES	23
2.6 SAS model	25
2.7 Different ways to evaluate resolution	27
A Appendix	30

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

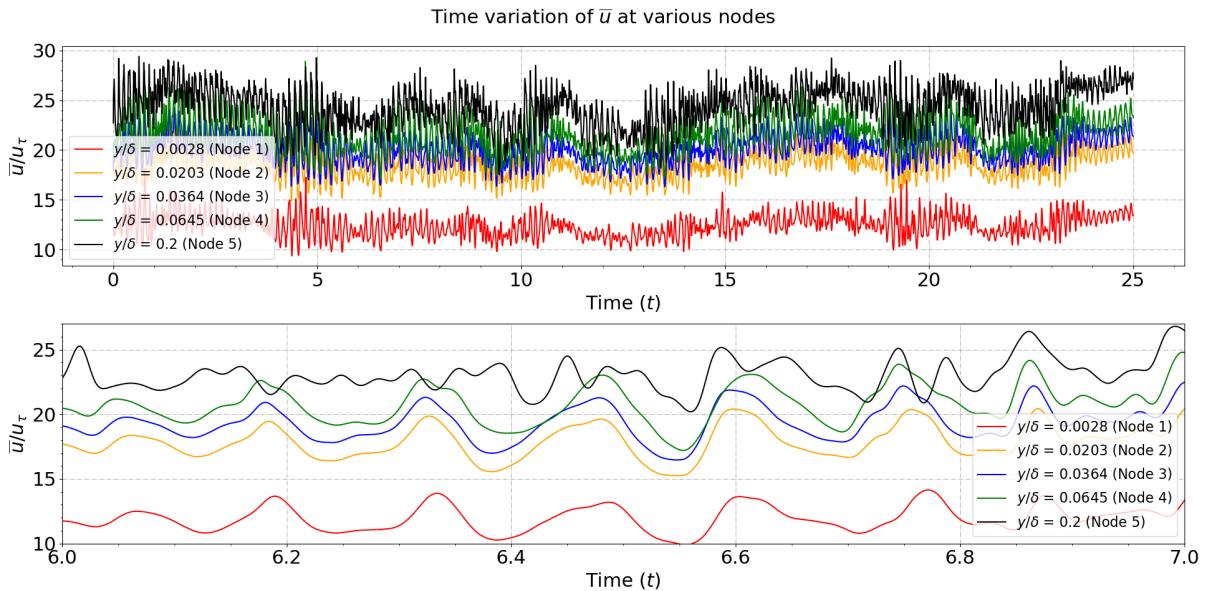


Figure 1: Time History of \bar{u} at various nodes

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 damped 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, \bar{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 \bar{w} , nevertheless we could see that it is slightly smaller at node 1.

Figure 3 shows the auto-correlation calculated at the five node points for the fluctuations \bar{u}' . Auto-correlation is the correlation 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-correlation 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.

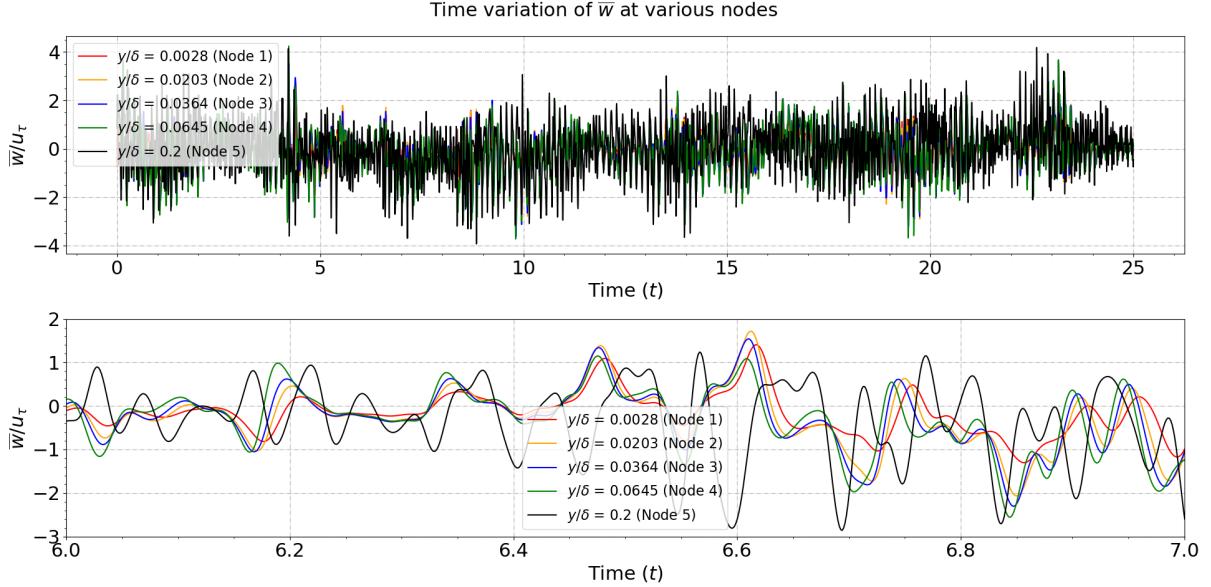


Figure 2: Time History of \bar{w} at various nodes

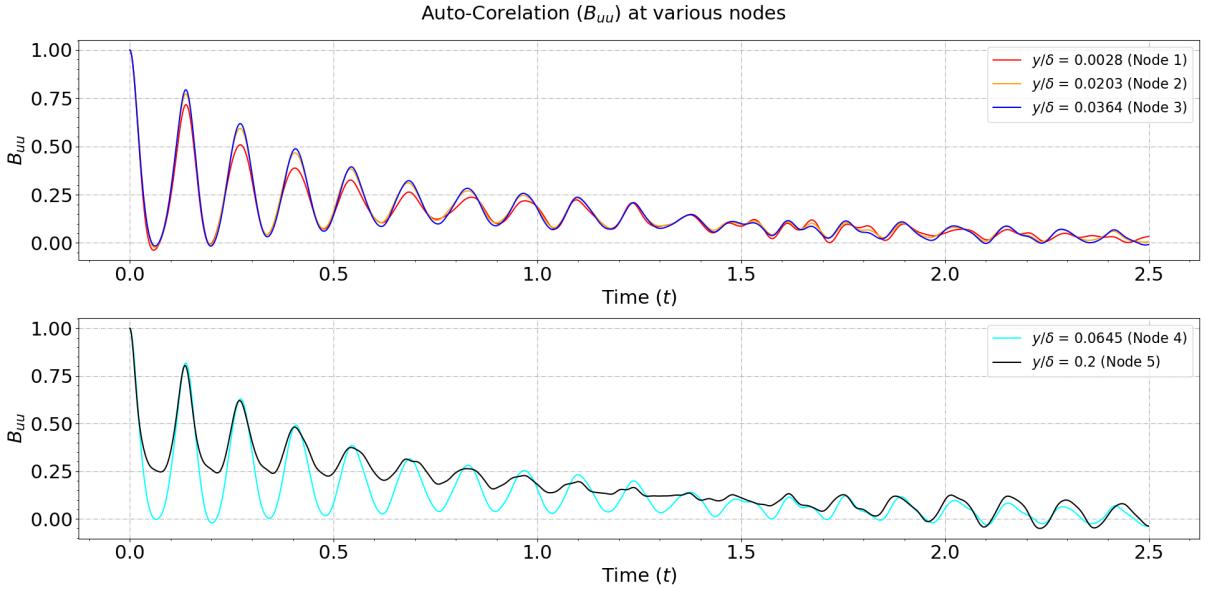


Figure 3: Auto-Correlation at various nodes

Looking at the graph of auto-correlation in figure 3, we see that the auto-correlation 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-correlation should converge to zero ideally. Here, due to numerical noise, it does not converge to zero.

Integral timescale is basically the area under the curve for the auto-correlation .vs. time graph. To calculate the integral time scale, we integrate till the point where the auto-correlation is zero or it starts increasing (see figure 4). The auto-correlation 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

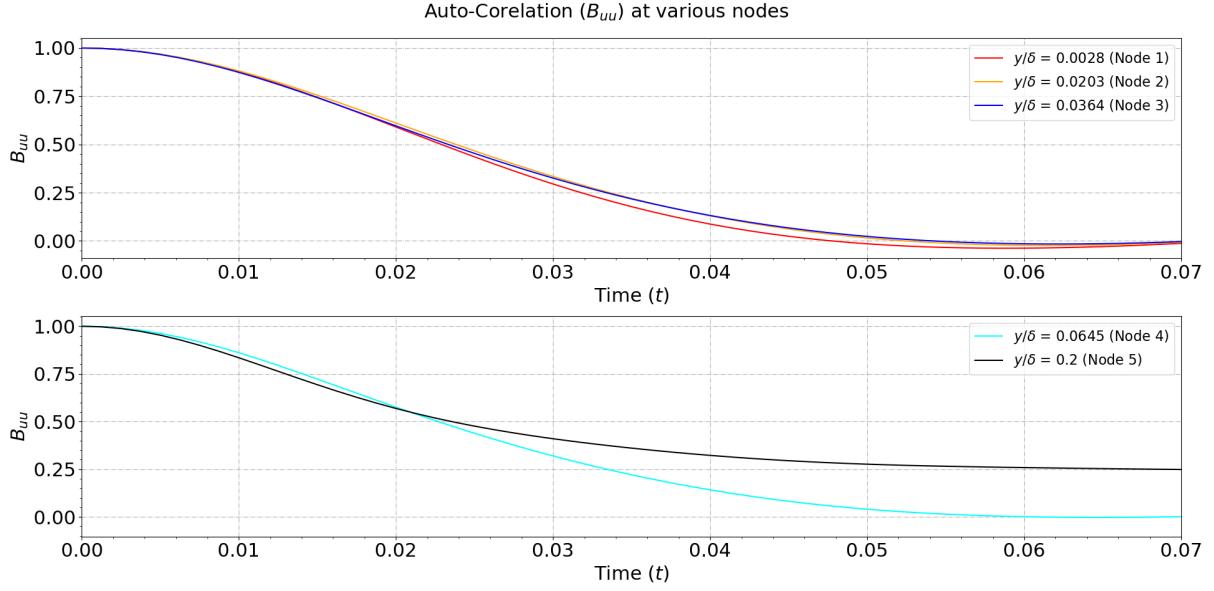


Figure 4: Auto-Correlation at various nodes (Zoomed-in)

turbulent quantity will change more rapidly at that point. At nodes 2-4, the auto-correlation is nearly the same hence the eddies at that points will have a similar time scale. For node 5, the auto-correlation 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-averaging is performed on homogenous x_1 and x_3 directions. The resultant $\langle \bar{v}_1 \rangle$ 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.

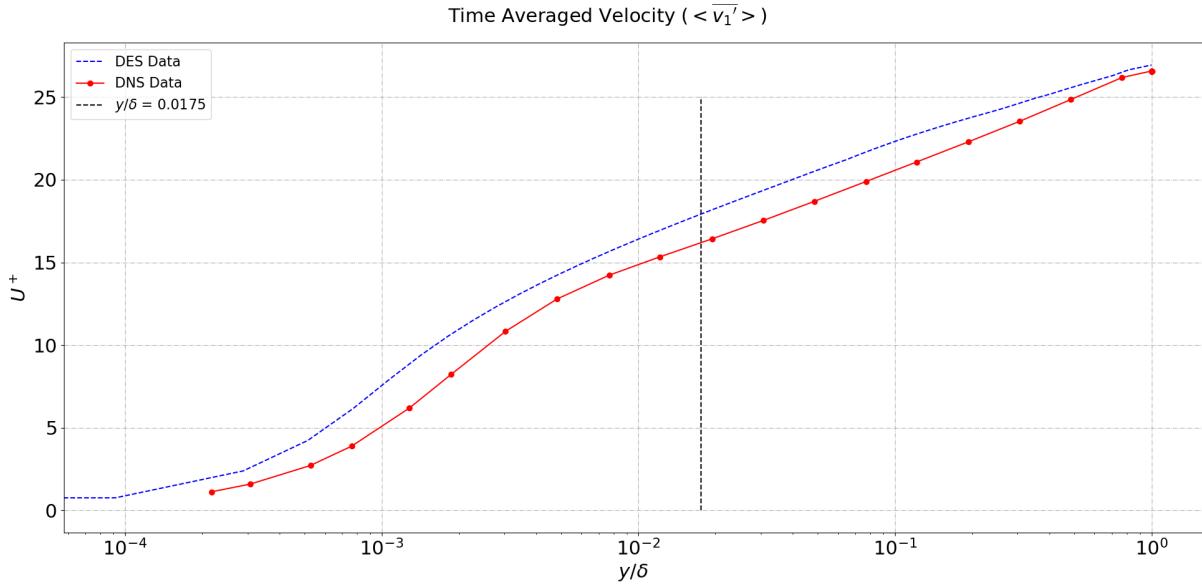


Figure 5: Time averaged velocity $\langle \bar{v}_1 \rangle$

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).

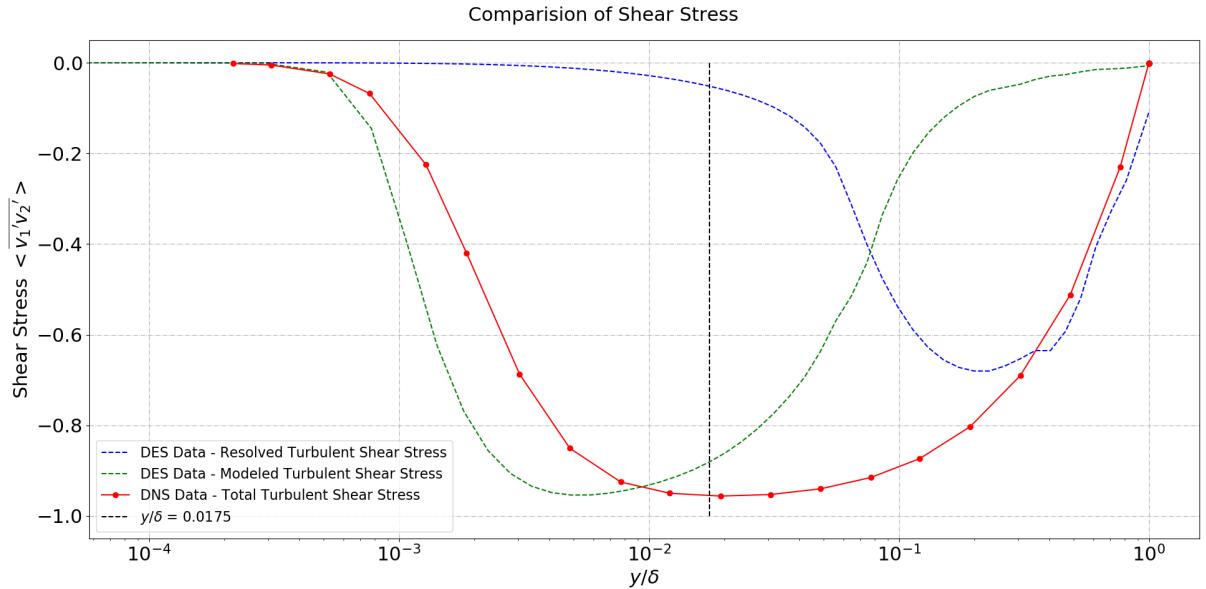


Figure 6: Comparision of modeled and resolved turbulent shear stress

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.

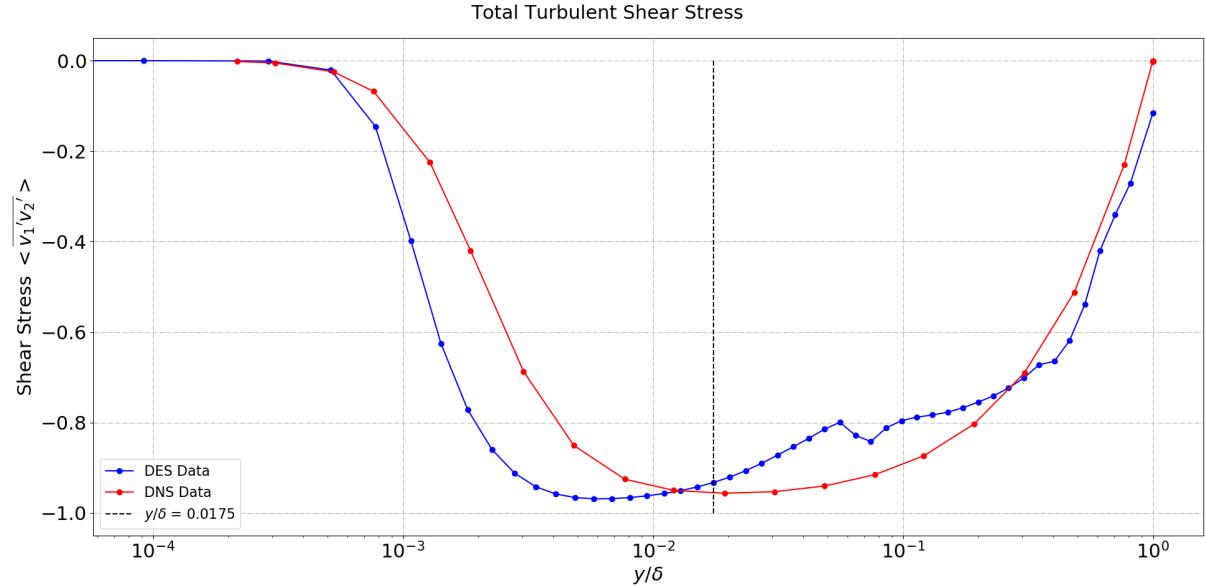


Figure 7: Total turbulent shear stress

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.

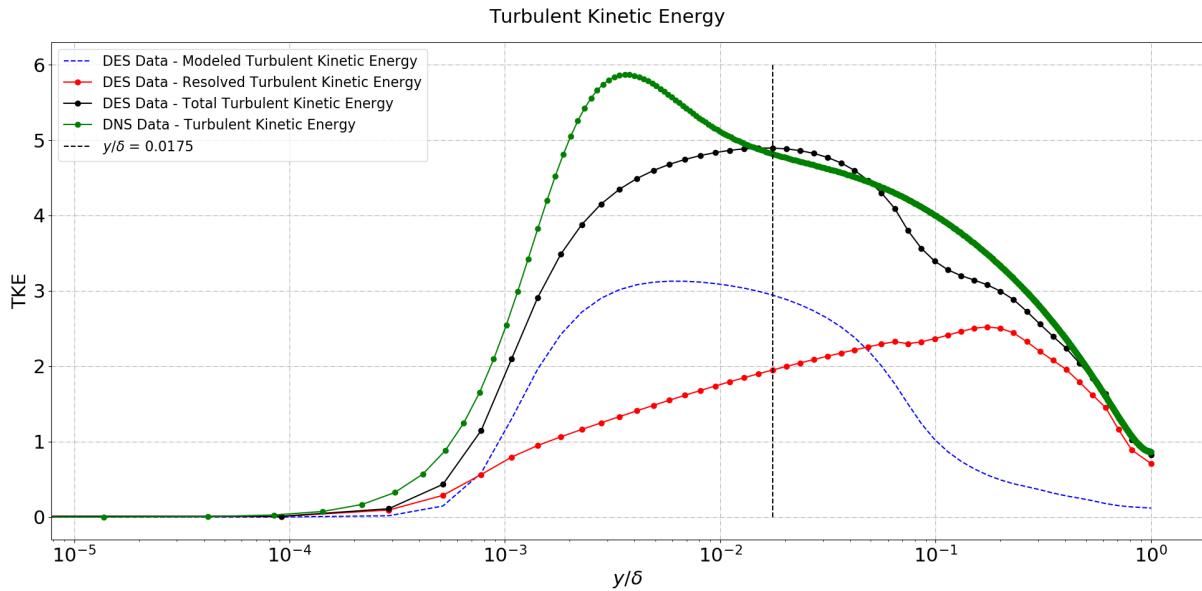


Figure 8: Turbulent Kinetic Energy

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.

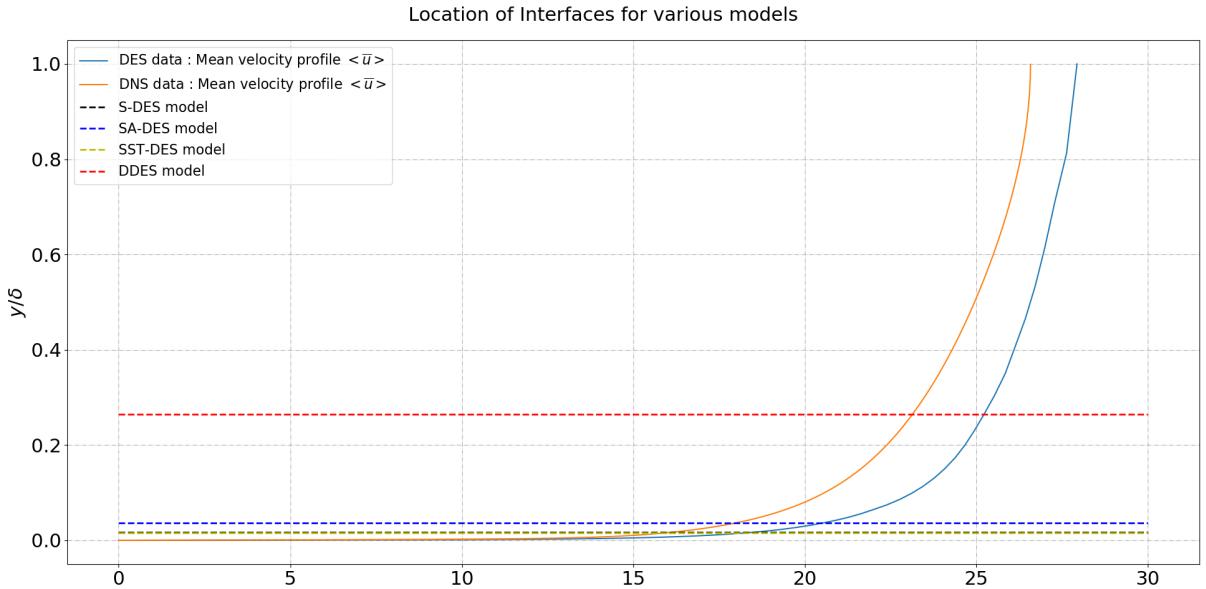


Figure 9: Location of Interface

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 -

$$L_{vK,1D} = \kappa \left| \frac{\partial \langle \bar{v}_1 \rangle / \partial x_2}{\partial^2 \langle \bar{v}_1 \rangle / \partial x_2^2} \right| \quad (1)$$

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

$$\begin{aligned} \partial \bar{v}_1 / \partial x_2 &= \bar{a}_1 + \dots = \mathcal{O}(x_2^0) \\ \partial^2 \bar{v}_1 / \partial x_2^2 &= 1 + \bar{a}_2 + \dots = \mathcal{O}(x_2^0) \\ \text{Hence, } \frac{\partial \bar{v}_1 / \partial x_2}{\partial^2 \bar{v}_1 / \partial x_2^2} &= \mathcal{O}(x_2^0) \end{aligned} \quad (2)$$

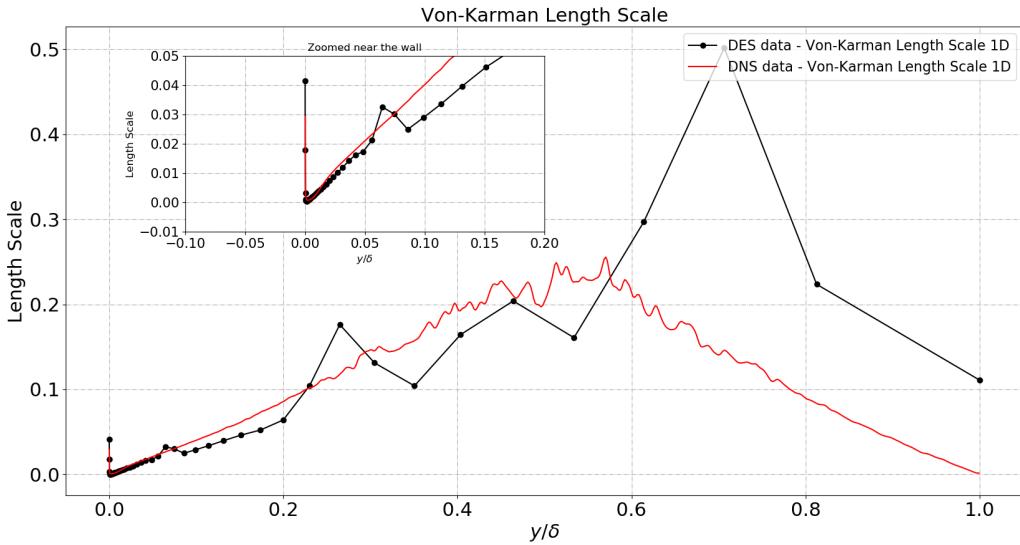


Figure 10: 1D Von-karman length scales

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

$$\begin{aligned} L_{vK,3D,inst} &= \kappa \left| \frac{S}{U''} \right| \\ S &= 2(S_{ij}S_{ij})^{0.5} \\ U'' &= \left(\frac{\partial^2 \bar{v}_i}{\partial x_j \partial x_j} \frac{\partial^2 \bar{v}_i}{\partial x_j \partial x_j} \right)^{0.5} \end{aligned} \quad (3)$$

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'') -

$$U'' = \left(\frac{\partial^2 \bar{v}_i}{\partial x_j \partial x_k} \frac{\partial^2 \bar{v}_i}{\partial x_j \partial x_k} \right)^{0.5} \quad (4)$$

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 slightly higher than method 2 (where we also consider cross-diffusion terms while calculating U'').

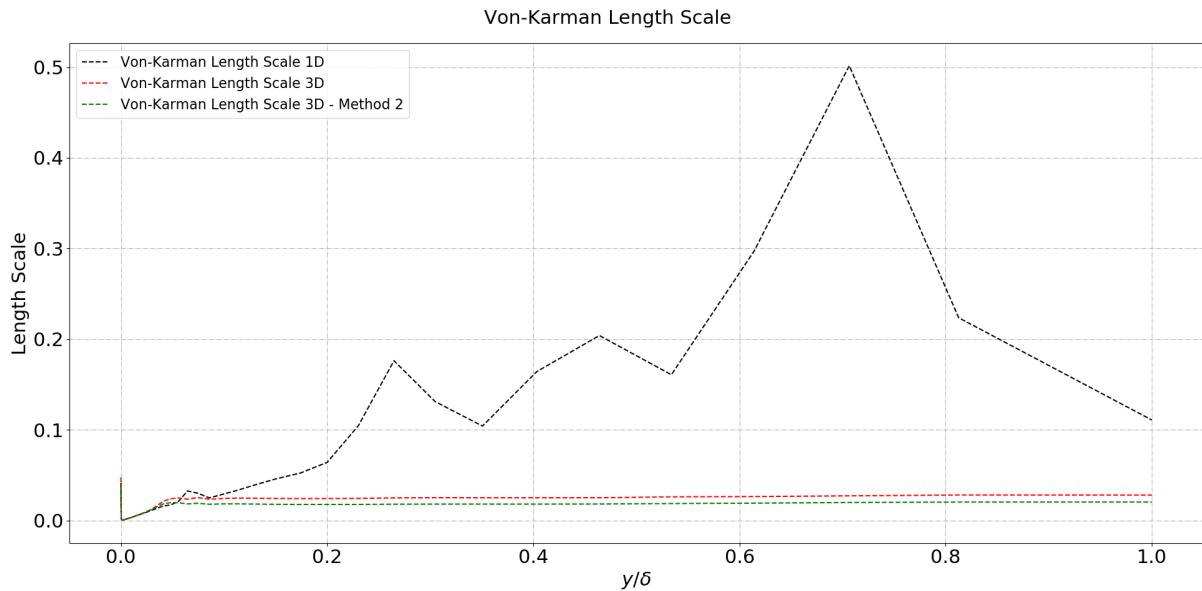


Figure 11: 3D Von-karman length scales

1.7 Estimation of resolution

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

$$\mathcal{A}(n)v = \Delta_x \Delta_y \Delta_z (\langle \bar{v}_i^{*(n)} \bar{v}_i^{*(n)} \rangle)^{0.5}$$

where, $\bar{v}_i^{*(n)} = \frac{-\Delta_n^2}{4} \frac{\partial \bar{v}_i}{\partial x_n \partial x_n}$

(5)

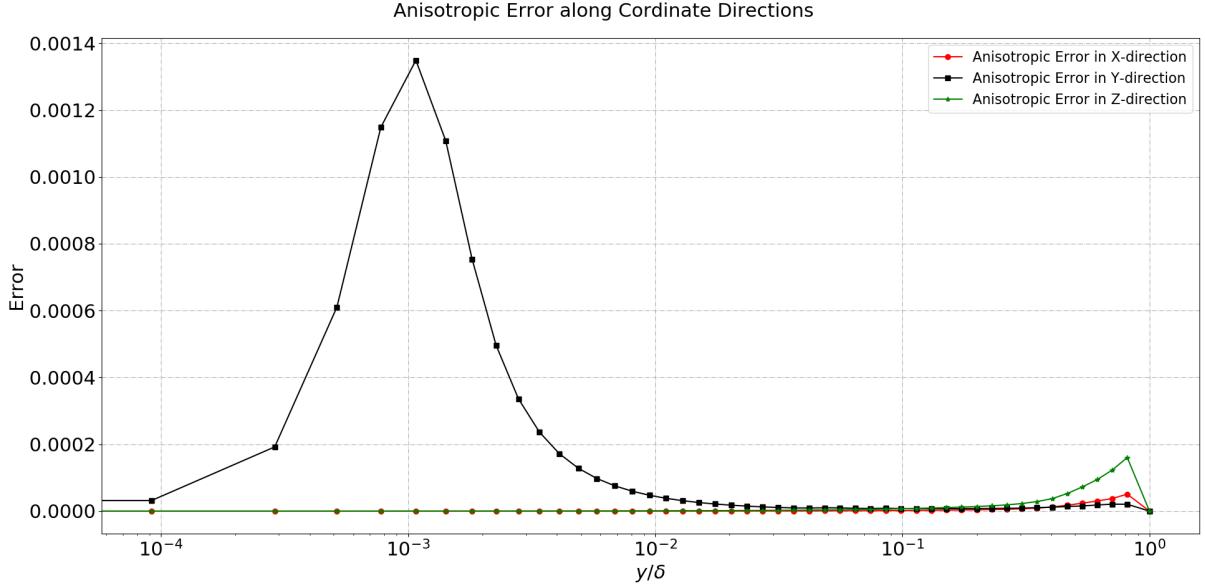


Figure 12: Anisotropic error

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 than 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.

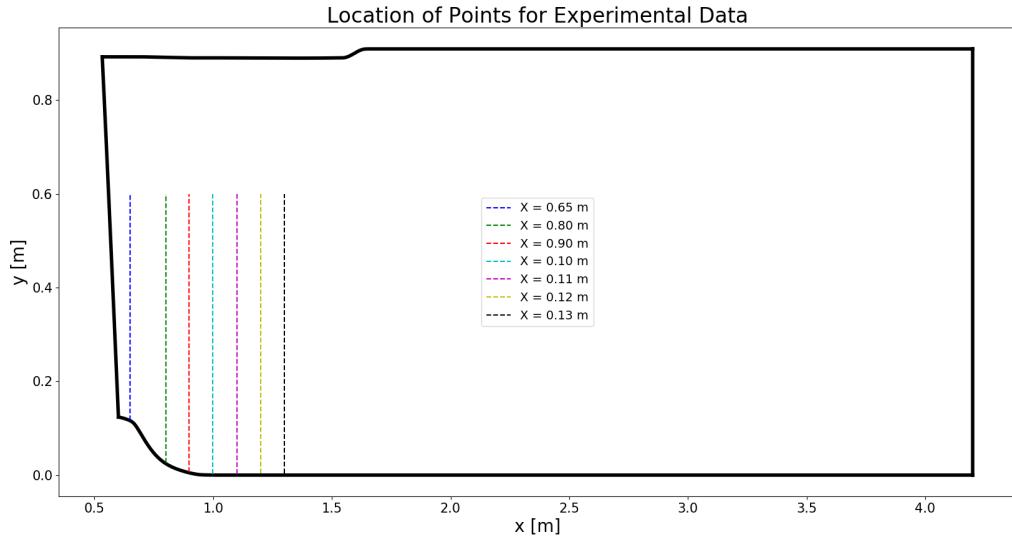


Figure 13: Locations for experimental data

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 $\langle \bar{u} \rangle$ 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 $\langle \bar{v} \rangle$ velocity, which shows a relatively good match (similar trend of velocity profile) with experiments apart from values at $x = 0.65$ m.

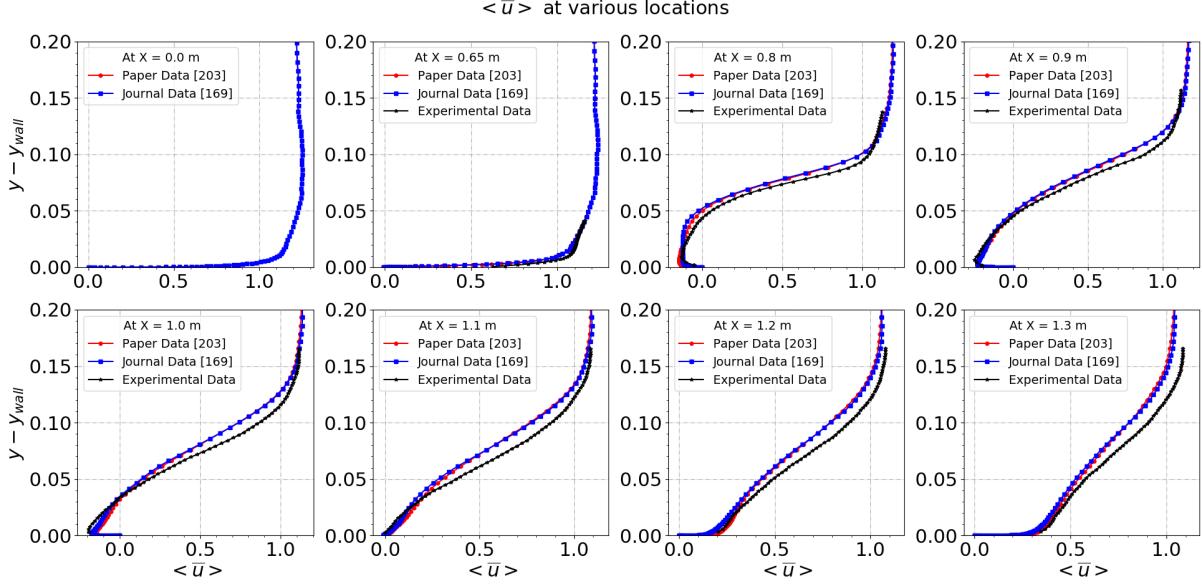


Figure 14: $\langle \bar{u} \rangle$ at various locations

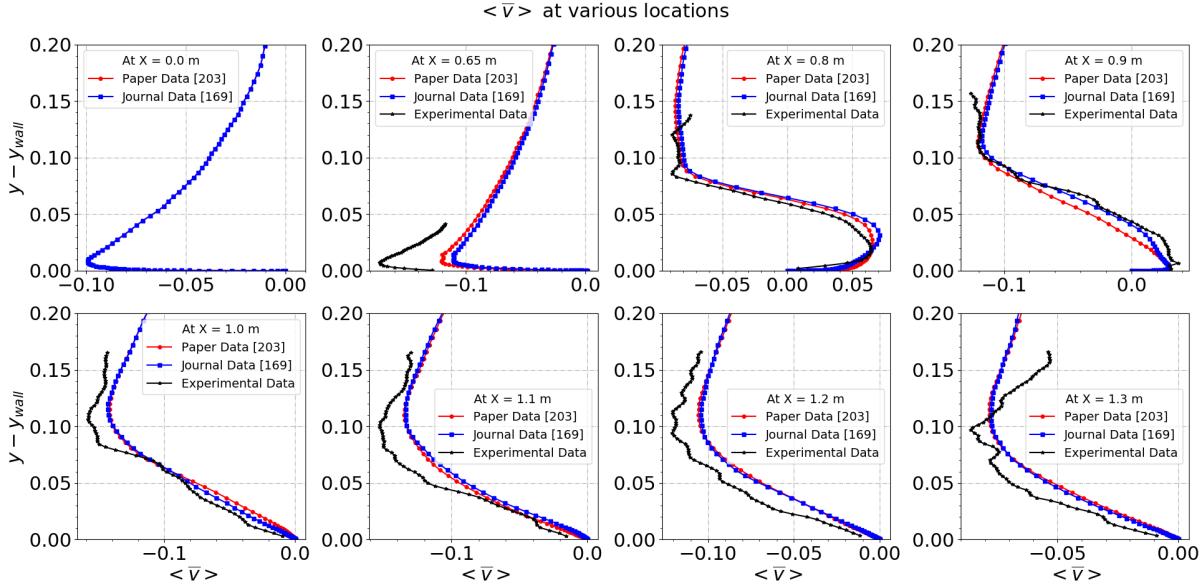


Figure 15: $\langle \bar{v} \rangle$ at various locations

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 $\langle \bar{u} \rangle$ 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 $\langle \bar{v} \rangle$ 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].

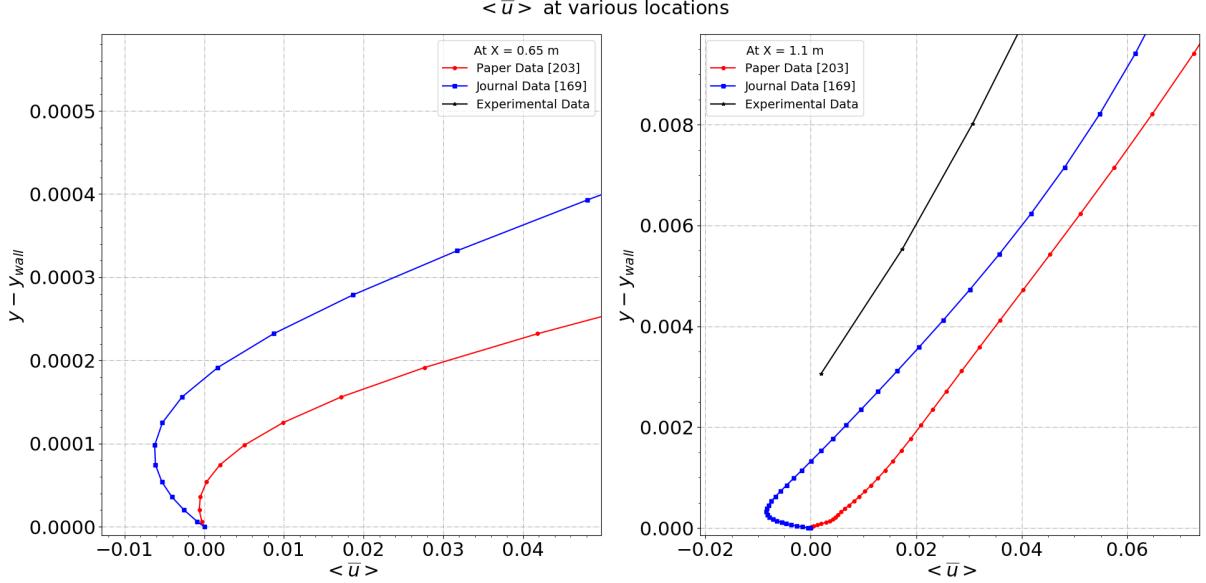


Figure 16: $\langle \bar{u} \rangle$ at $x = 0.65$ m and $x = 1.1$ m

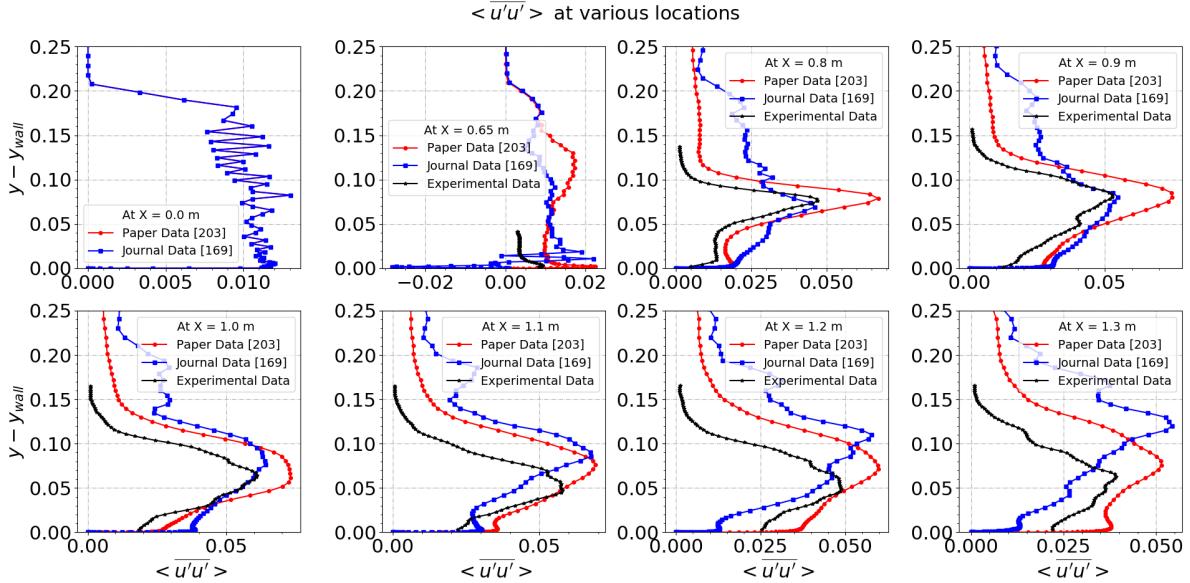
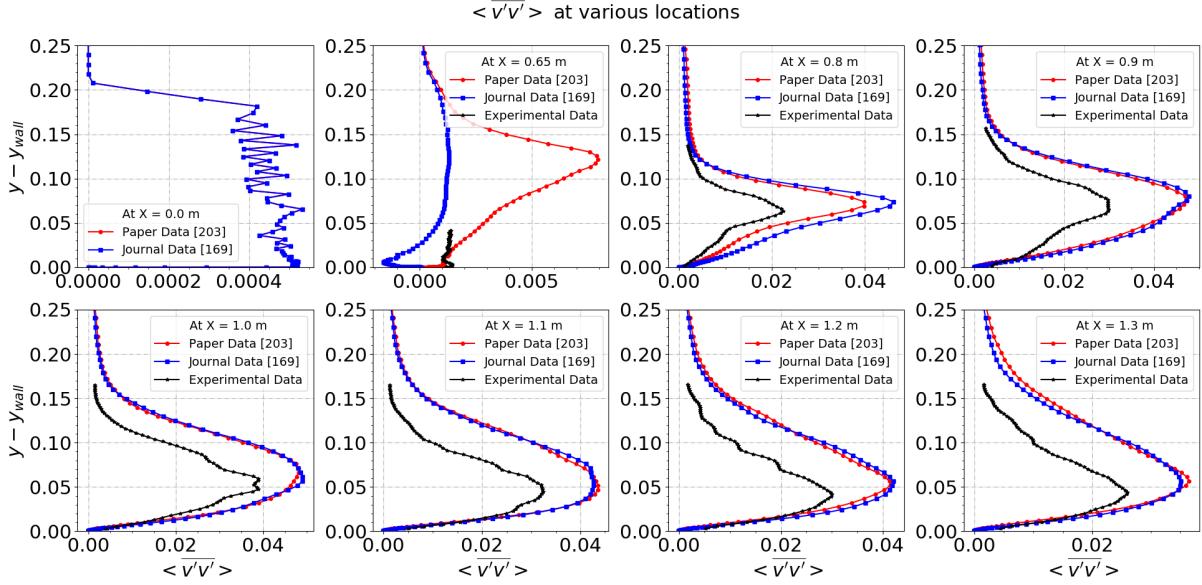
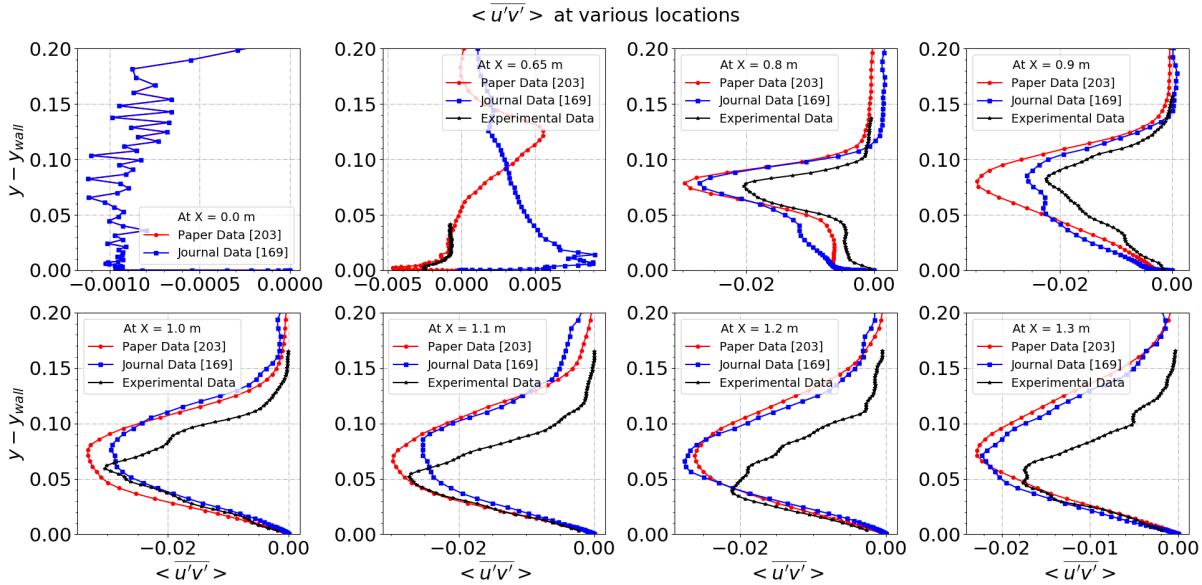


Figure 17: $\langle u'u' \rangle$ at various locations

Figures 17, 18 and 19 shows the turbulent fluctuations $\langle u'u' \rangle$, $\langle v'v' \rangle$ and $\langle u'v' \rangle$ 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 ($\langle u'u' \rangle$) at $x = 0.65$ m as shown in figure 17. We can see the


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

 Figure 19: $\langle u'v' \rangle$ 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 $\langle v'v' \rangle$ 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 $\langle \bar{u}'\bar{v}' \rangle$ 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 hybrid central differencing (95%) and upwind scheme (5%).

2.2 Modeled turbulent shear stress

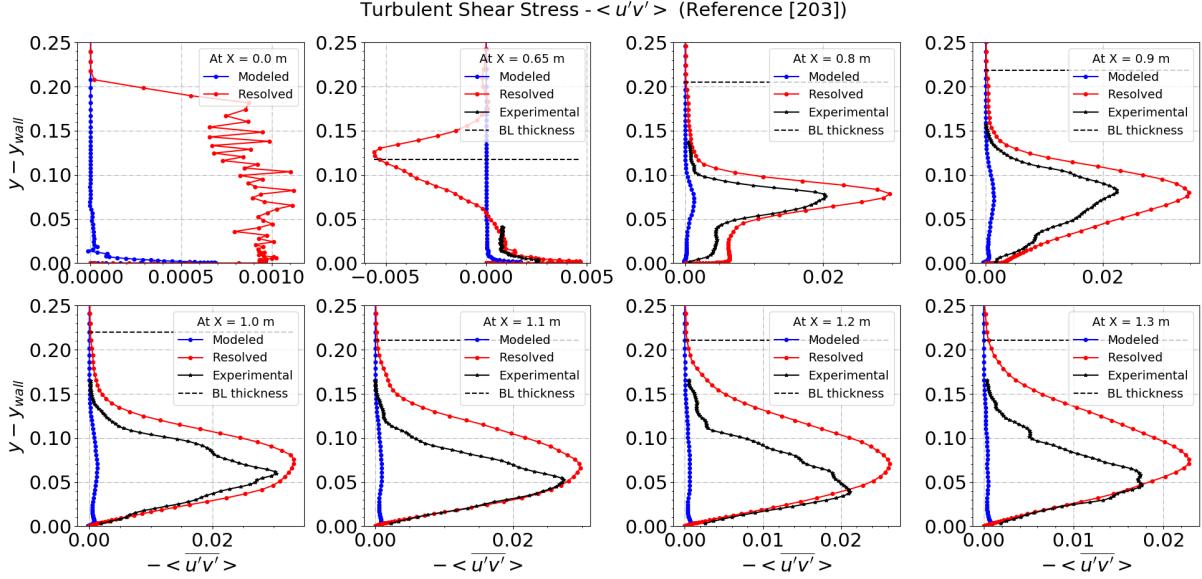


Figure 20: Turbulent shear stress $-\langle u'v' \rangle$ at various locations (Data from reference[203])

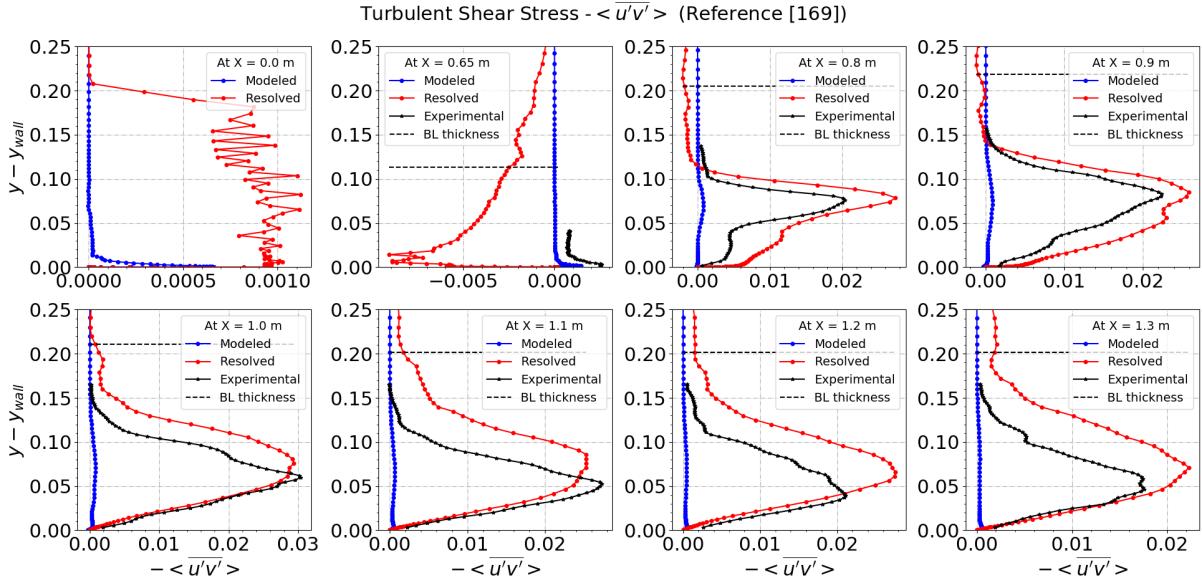


Figure 21: Turbulent shear stress $-\langle u'v' \rangle$ at various locations (Data from reference[169])

In this section, the turbulent shear stresses ($-\langle u'v' \rangle$) 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.

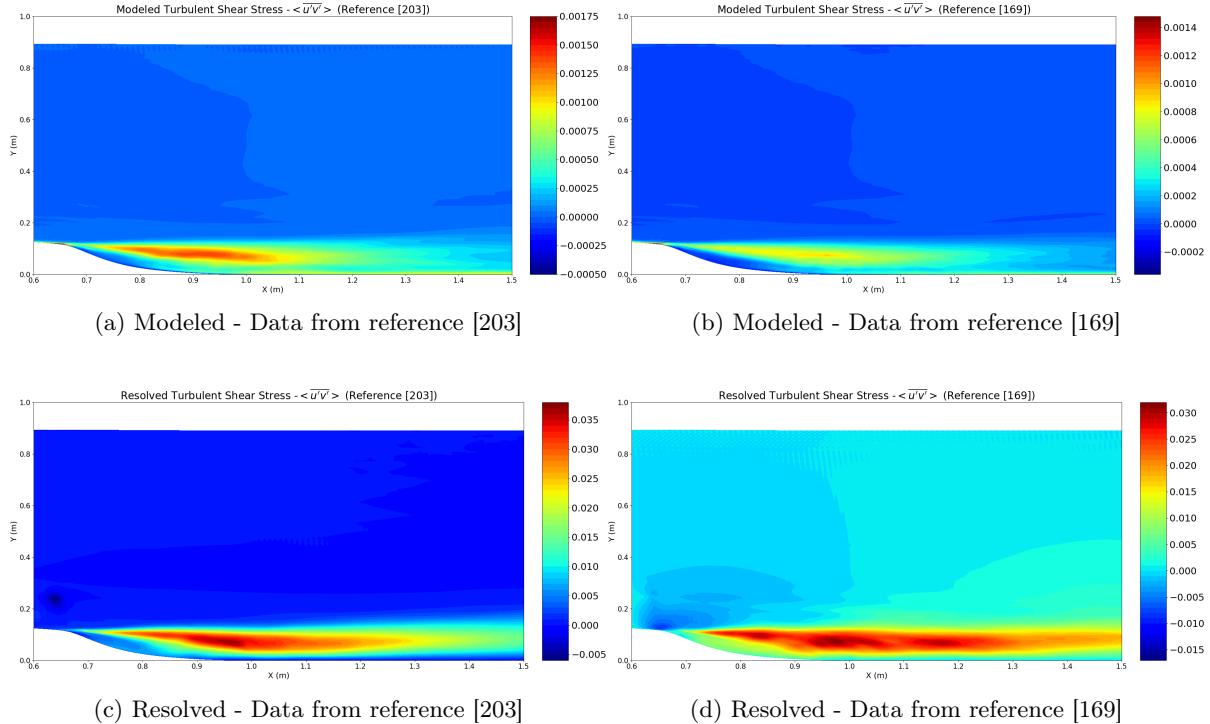


Figure 22: Turbulent Shear Stress ($-\langle u'v' \rangle$)

2.3 Turbulent viscosity

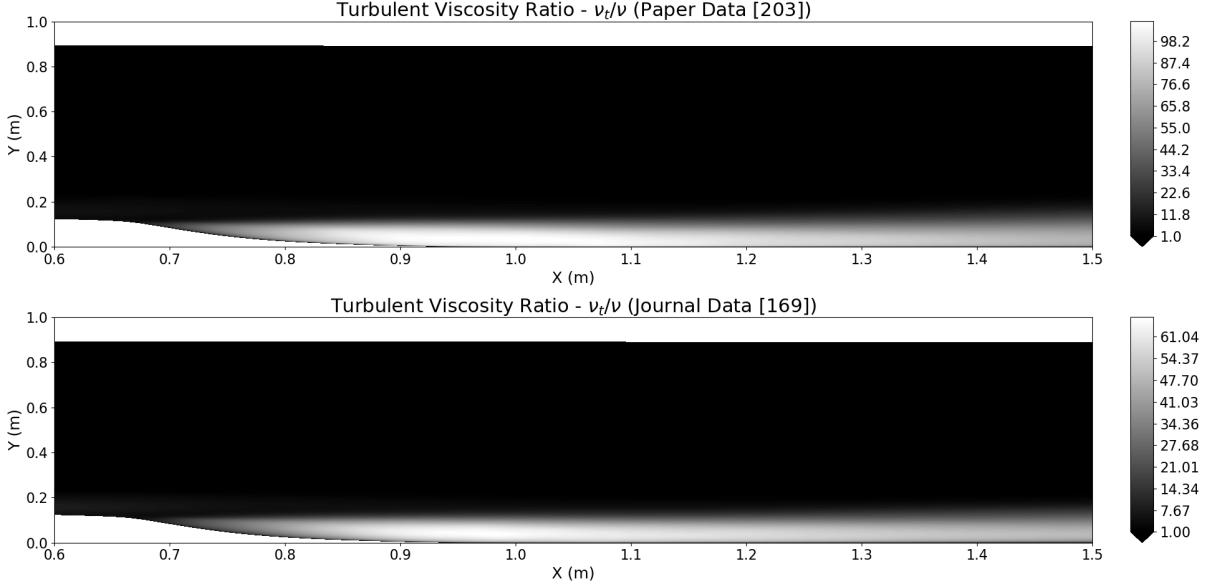


Figure 23: Turbulent viscosity ratio (ν_t/ν)

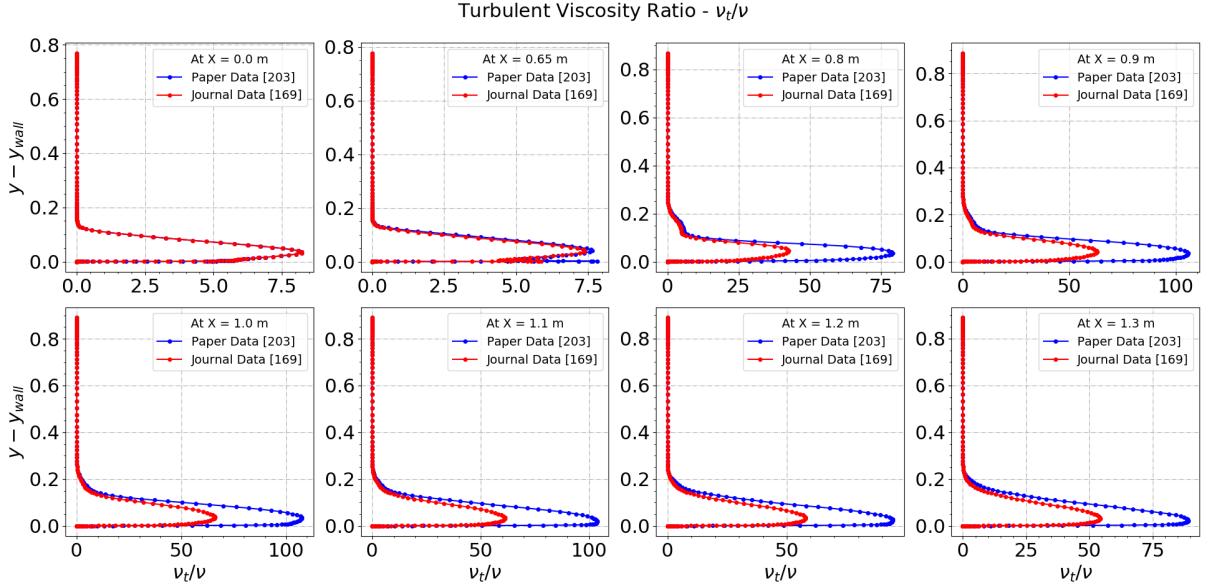


Figure 24: Turbulent viscosity ratio (ν_t/ν)

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 ($\nu_t = C_\mu \frac{k^2}{\epsilon}$).

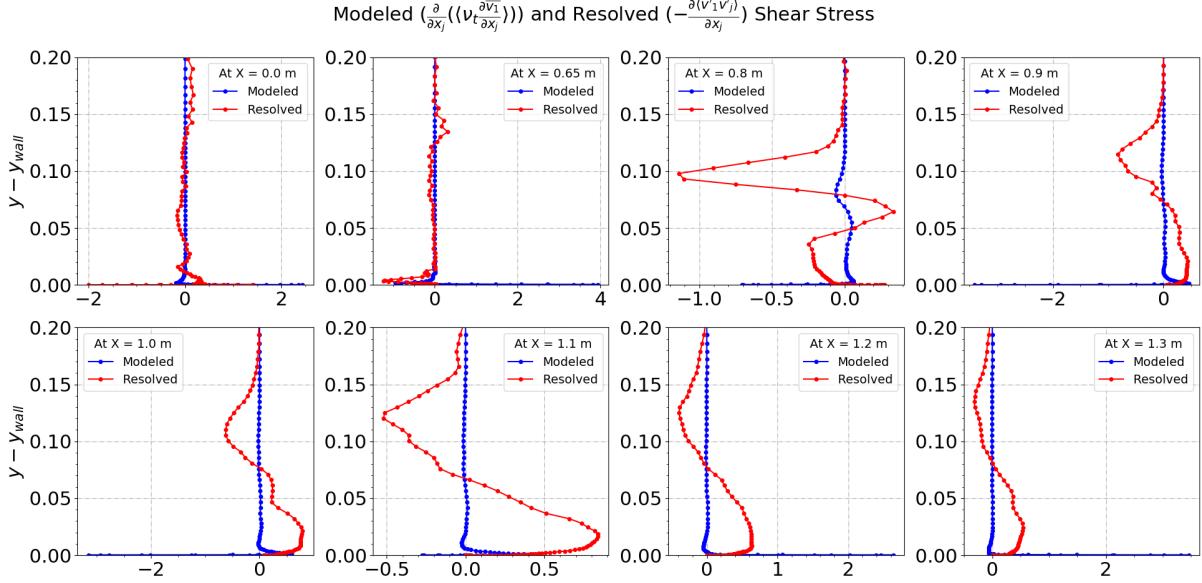


Figure 25: Modeled and resolved term in momentum equation (for $i=1$)

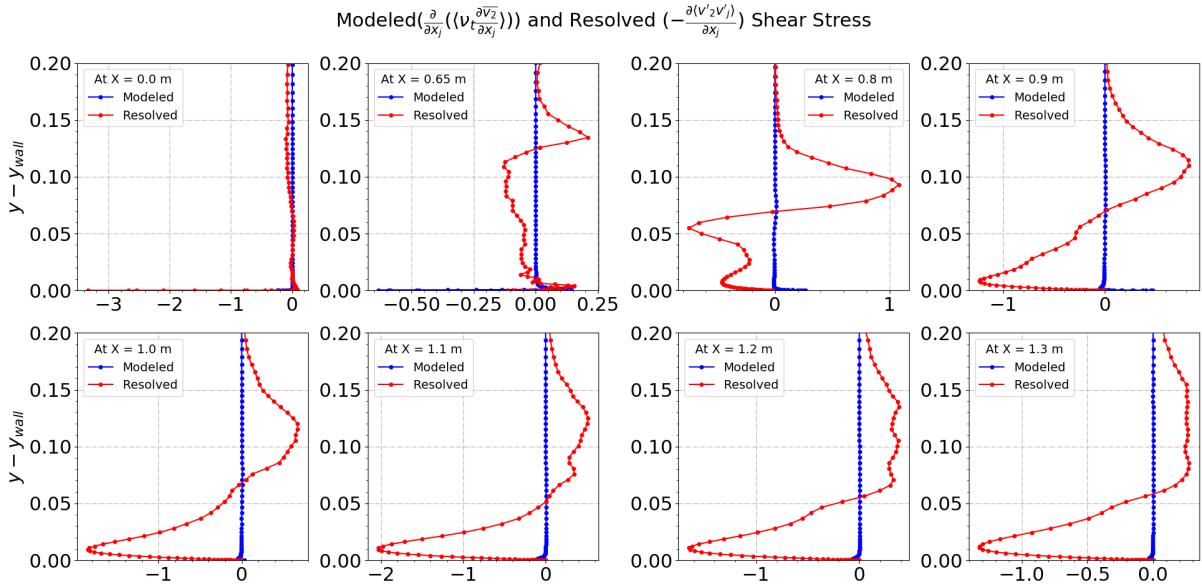


Figure 26: Modeled and resolved term in momentum equation (for $i=2$)

In figure 25 and figure 26, two terms from the momentum equations are plotted - modeled ($\frac{\partial}{\partial x_j} (\langle \nu_t \frac{\partial \bar{v}_1}{\partial x_j} \rangle)$) 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

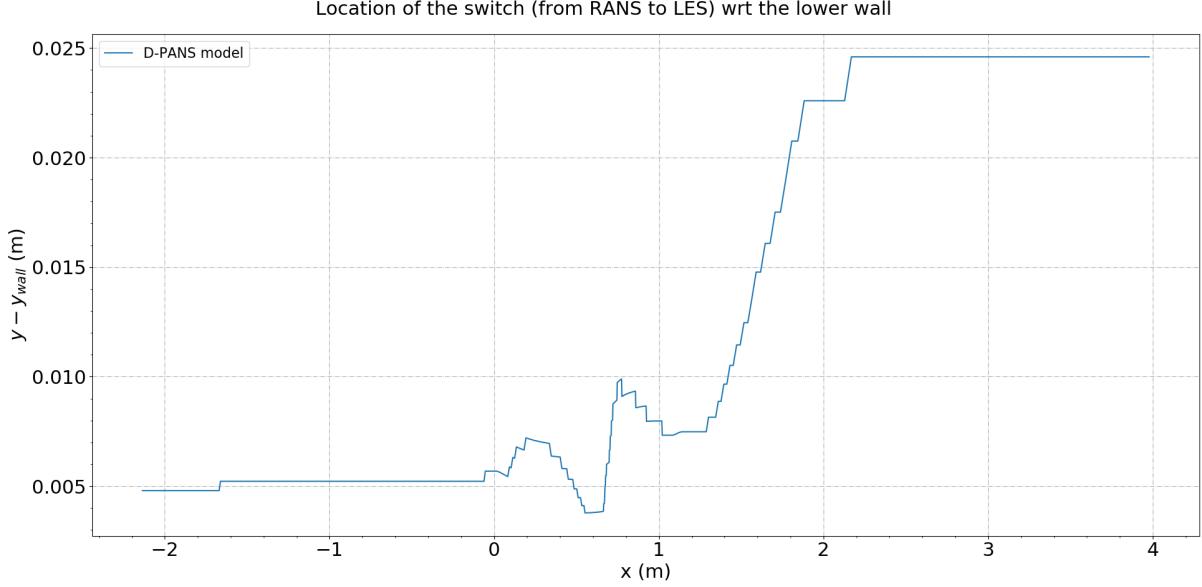


Figure 27: Location of interface for D-PANS model

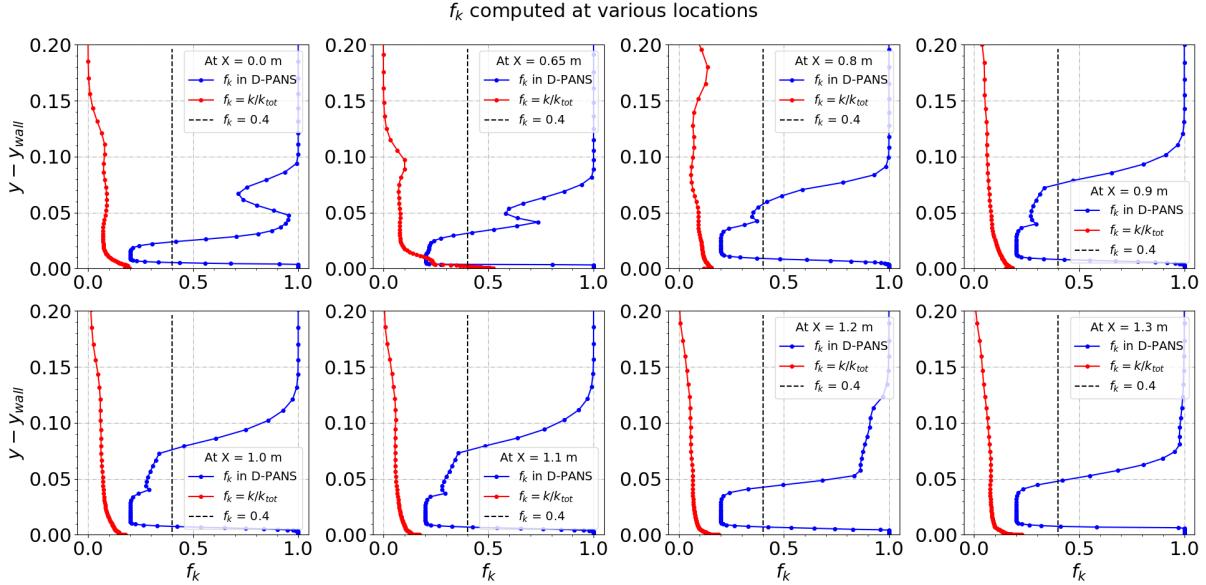


Figure 28: Comparison of f_k obtained using (i)D-PANS and (ii) k/k_{tot}

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 below 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. Krajnović, S. Girimajhi, and Z. Pavlović. 'Near-wall formulation of the Partially Averaged Navier Stokes turbulence model.' AIAA

Journal, 49(12):2627–2636, 2011.

$$\text{Theoretical definition of } f_k = \frac{k}{k_{tot}} \quad (6)$$

$$f_k \text{ as per reference [216]} = C_\mu^{-0.5} \left(\frac{\Delta}{L_t} \right)^{2/3}$$

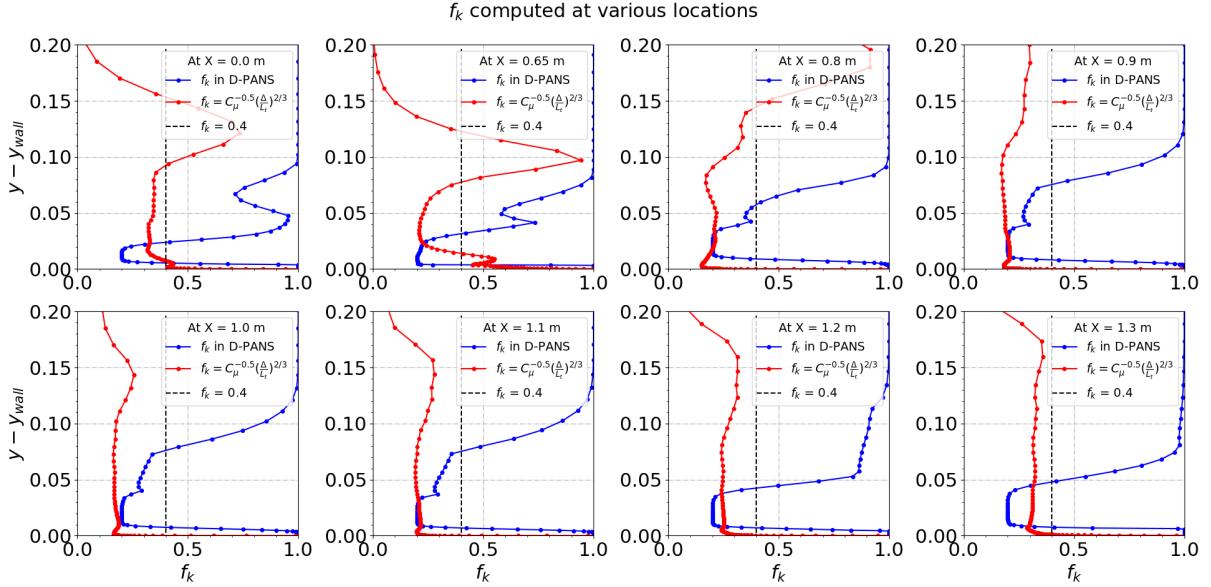


Figure 29: Comparison of f_k obtained using (i)D-PANS and (ii)Reference [216] - Basara

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

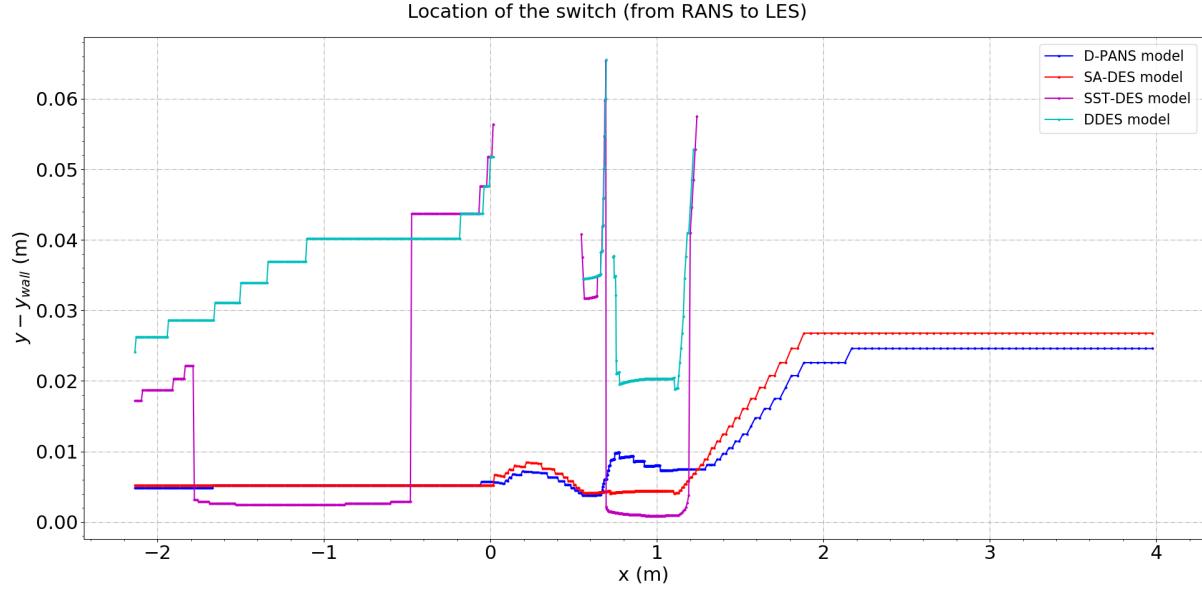


Figure 30: Location of interface for various models

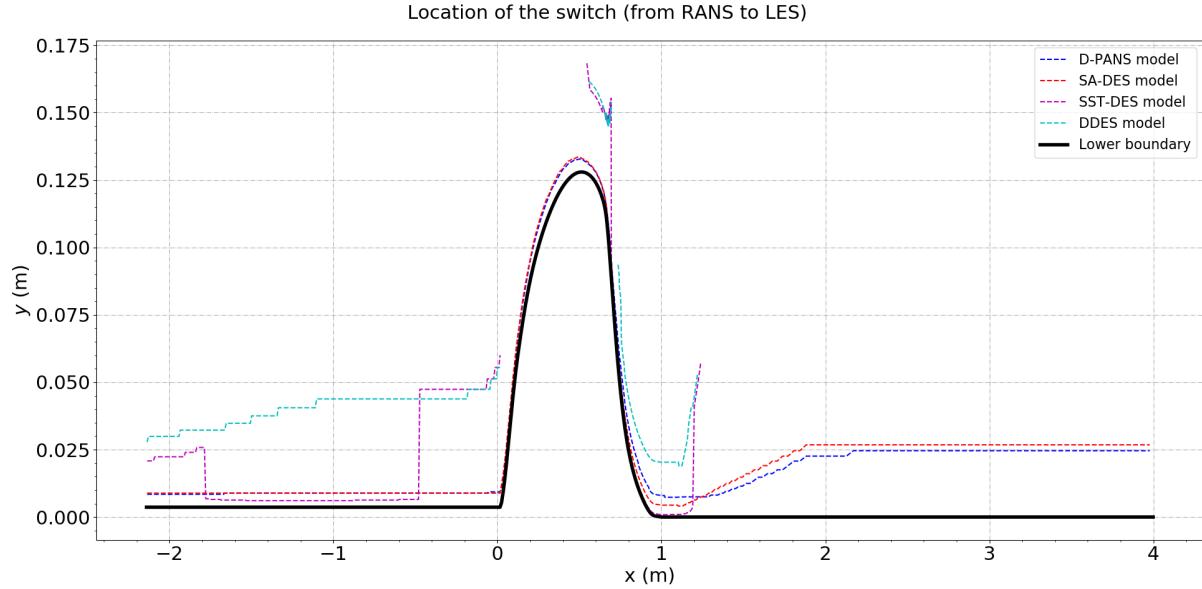


Figure 31: Location of interface for various models - visualisation wrt lower wall

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.

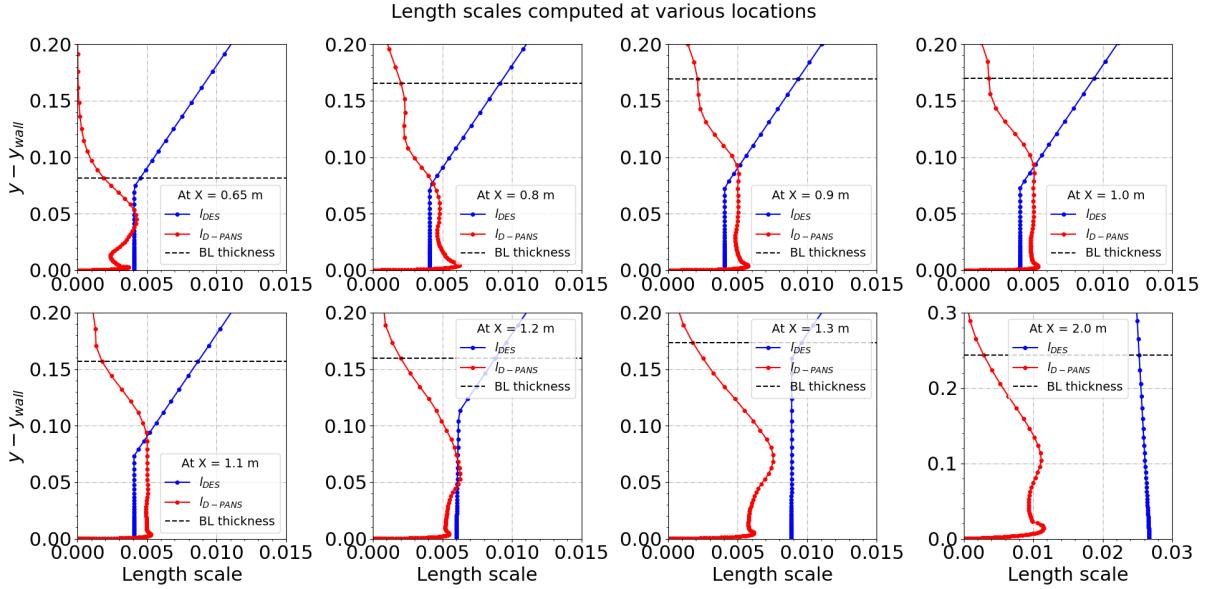


Figure 32: Length scale computed for DES and D-PANS model

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.

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

2.6 SAS model

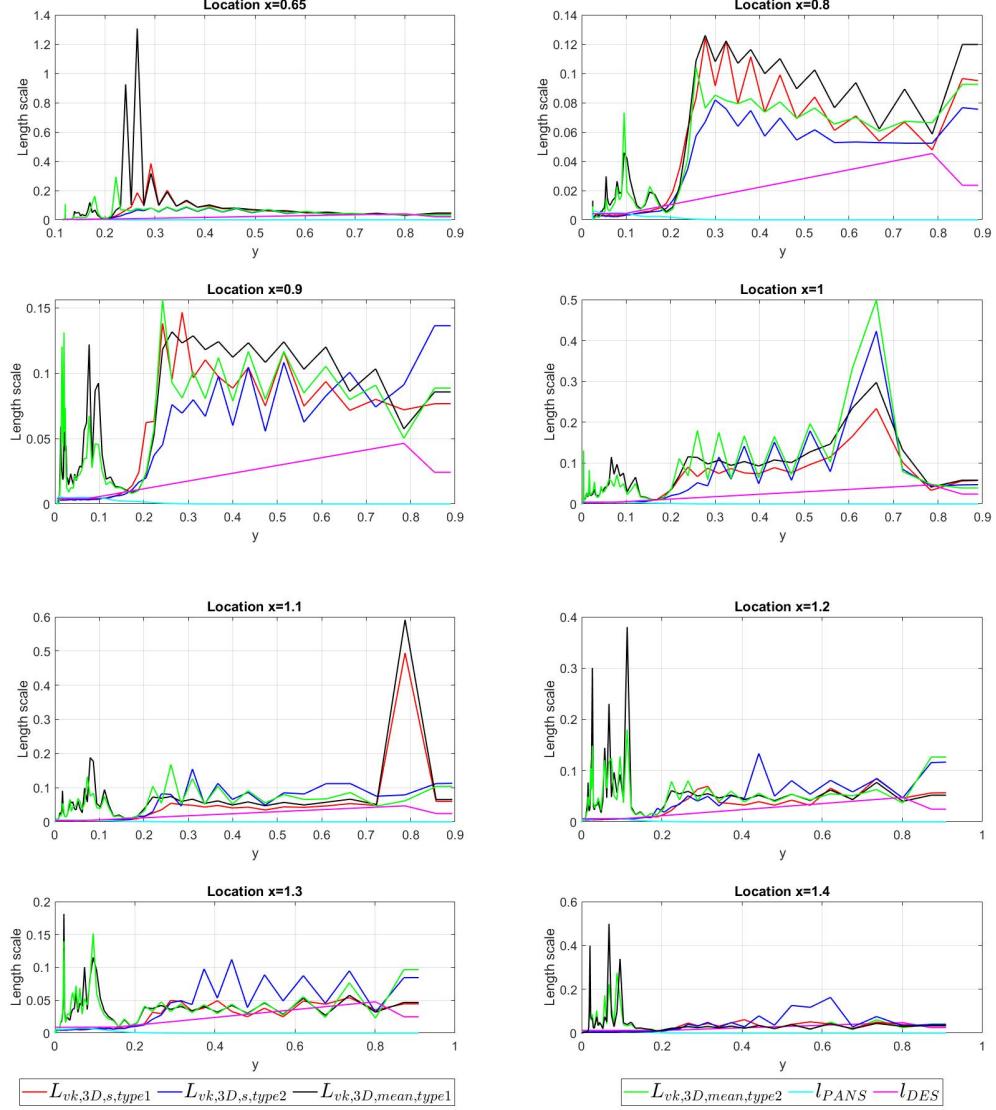


Figure 33: Length scale computed for SAS model

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.

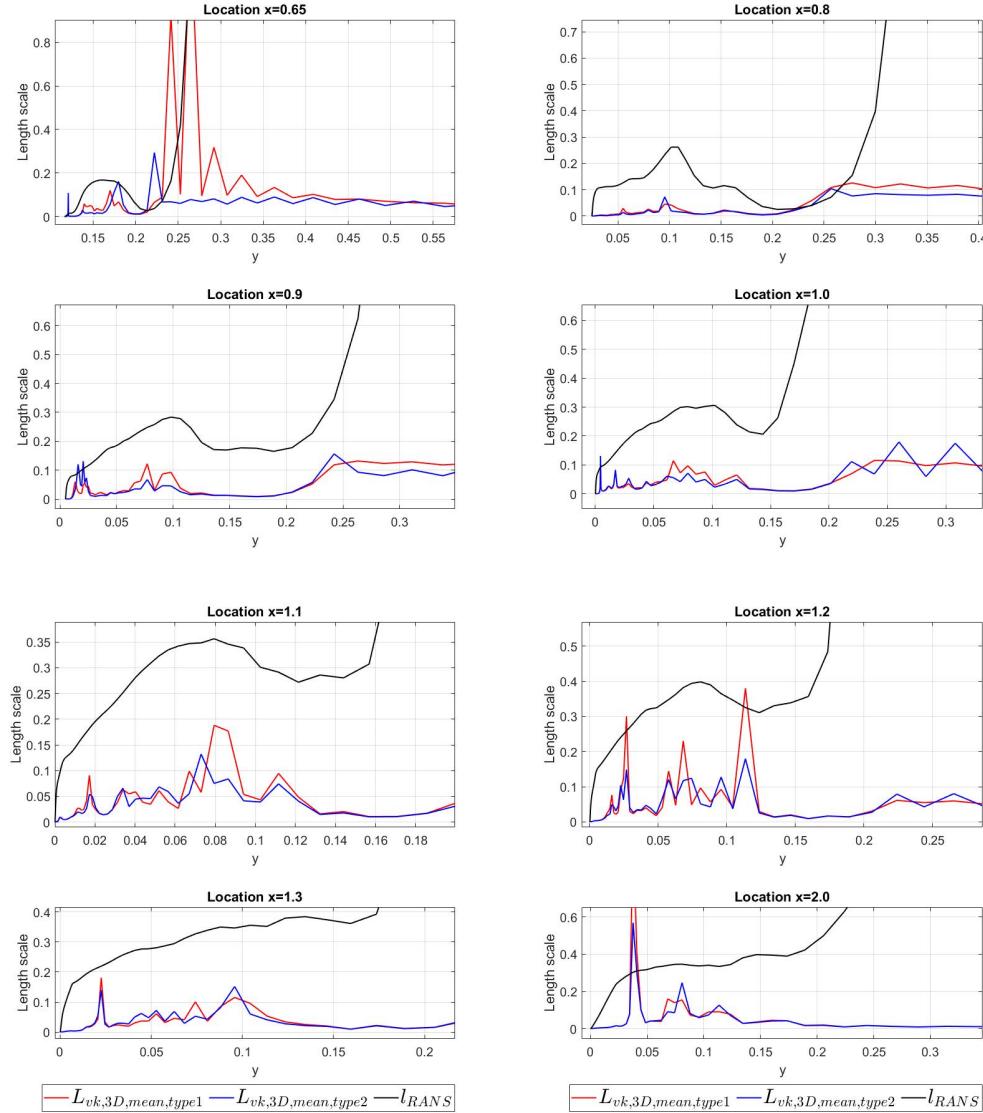


Figure 34: Length scale computed for SAS and RANS model

2.7 Different ways to evaluate resolution

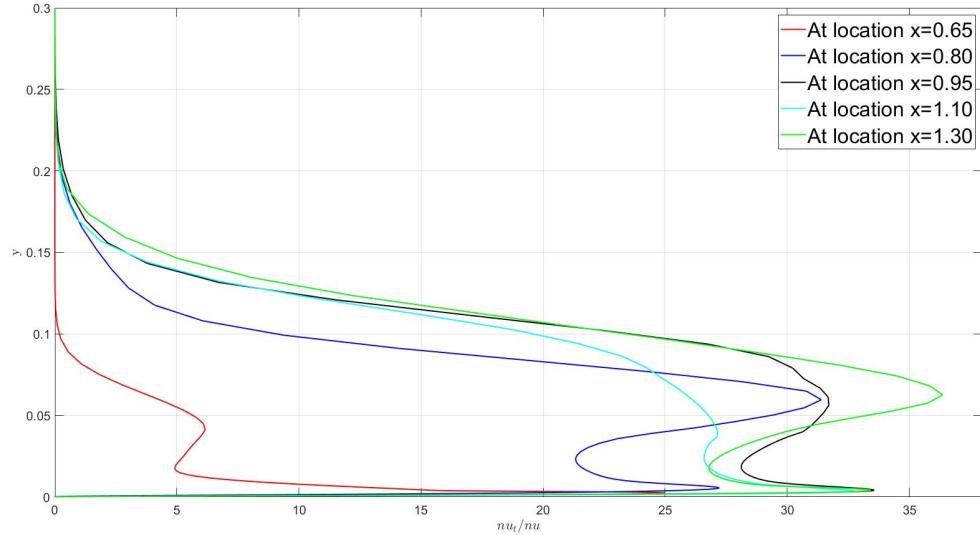


Figure 35: Ratio Between Viscous and Modelled Turbulent Viscosity

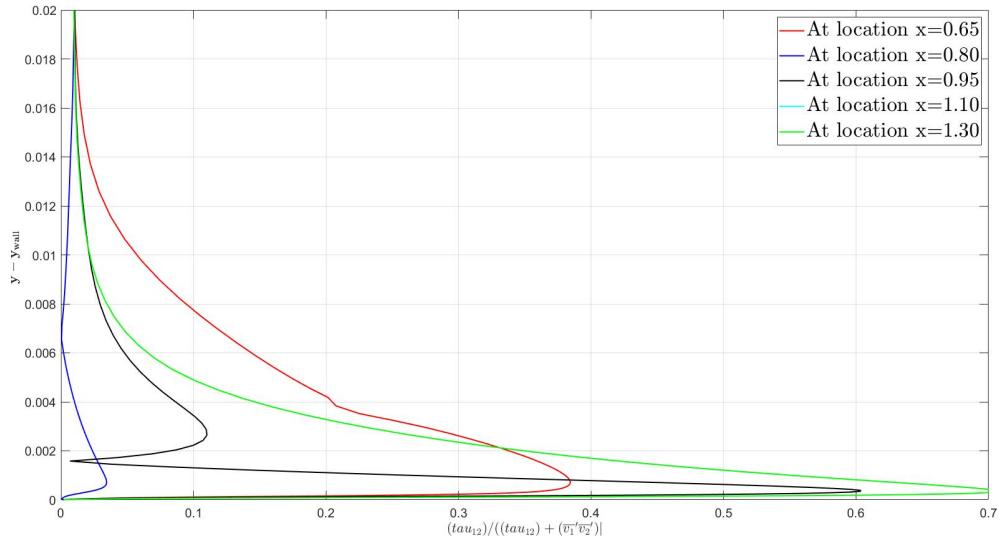


Figure 36: Ratio Between Modeled and Total Shear Stress

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

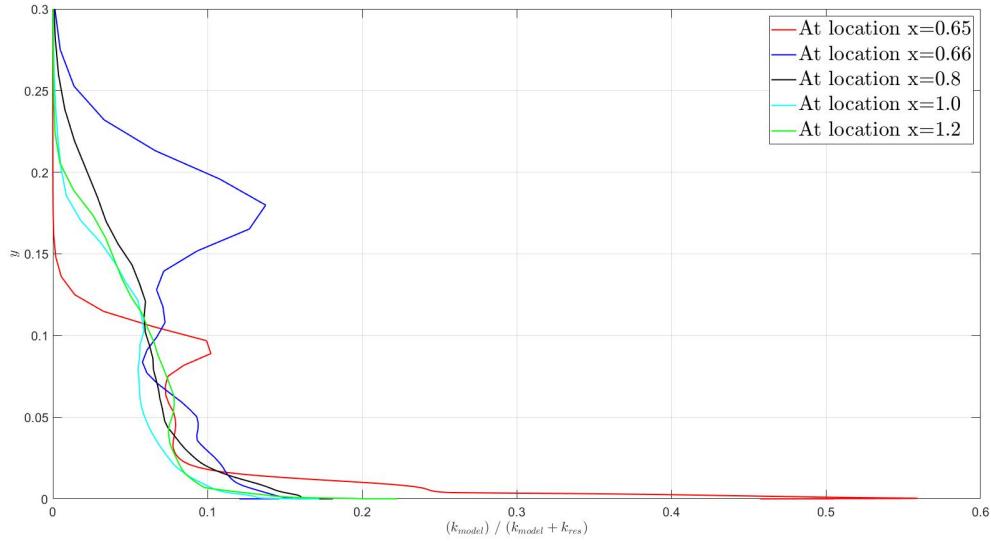


Figure 37: Ratio Between the Modelled and Total Turbulent Kinetic Energy

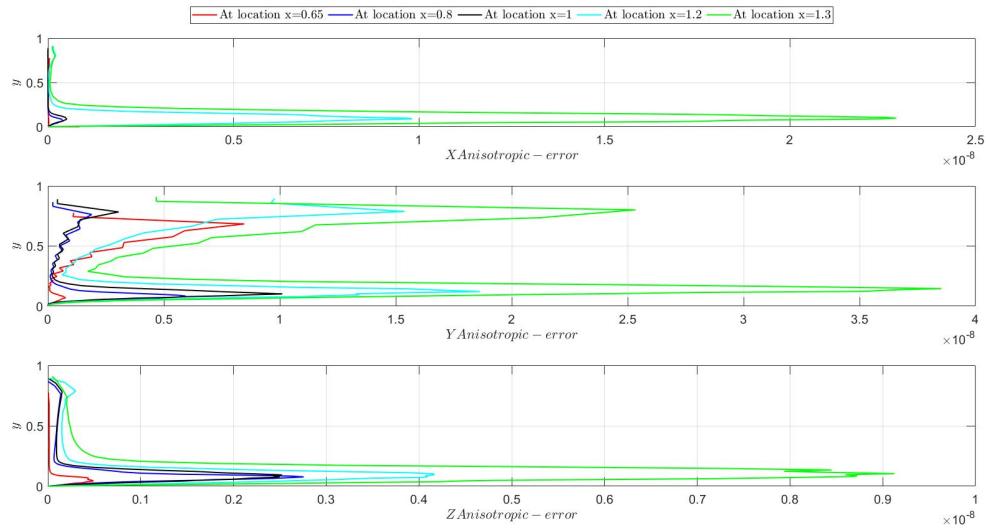


Figure 38: Anisotropic error

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.

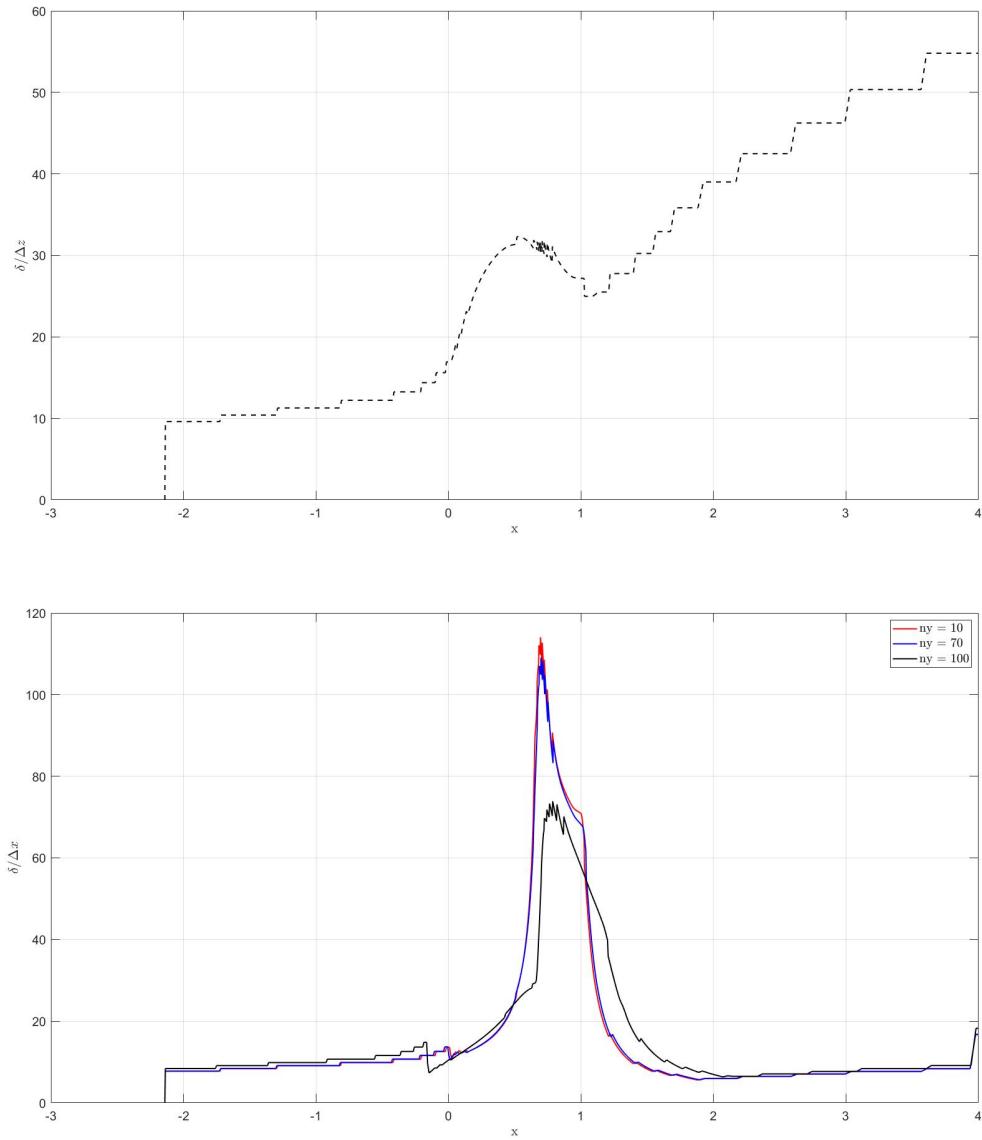


Figure 39: Ratio of boundary layer thickness to cell length

A Appendix

For Assignment 2a - Section 1.1

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from scipy.signal import welch, hann
4 from IPython import display
5 from matplotlib.ticker import (MultipleLocator, FormatStrFormatter,
6                                 AutoMinorLocator)
7 #plt.rcParams.update({'font.size': 20})
8
9 # makes sure figures are updated when using ipython
10 display.clear_output()
11
12 # ***** read u
13 data = np.genfromtxt("u_w_time_5nodes_sdes.dat", dtype=None)
14
15 u1=data[:,0]    #v_1 at point 1
16 u2=data[:,1]    #v_1 at point 2
17 u3=data[:,2]    #v_1 at point 3
18 u4=data[:,3]    #v_1 at point 4
19 u5=data[:,4]    #v_1 at point 5
20
21 w1=data[:,5]    #w_1 at point 1
22 w2=data[:,6]    #w_1 at point 2
23 w3=data[:,7]    #w_1 at point 3
24 w4=data[:,8]    #w_1 at point 4
25 w5=data[:,9]    #w_1 at point 5
26
27 #print("u1=",u1)
28
29 dx=3.2/32
30 dt= 0.25*dx/20
31 t_tot=dt*len(u1)
32
33 t = np.linspace(0,t_tot,len(u1))
34
35 # %%%%%%%%%%%%%% plotting section %%%%%%%%%%%%%%
36 # plot u
37 #fig1 = plt.figure("Figure 1")
38 #plt.plot(t,u1,'b--')
39 #plt.plot(t,u4,'r-')
40 #plt.xlabel("t")
41 #plt.ylabel("u")
42 #plt.savefig('utime_python.eps',bbox_inches='tight')
43
44 ##### ###### ##### ##### ##### # zoom
45 #fig2 = plt.figure("Figure 2")
46 #plt.plot(t,u1,'b--')
47 #plt.plot(t,u4,'r-')
48 #plt.xlabel("t")
49 #plt.ylabel("u")
50 #plt.axis([6, 7, 10, 22])
51 #plt.savefig('utime_zoom_python.eps',bbox_inches='tight')
52
53 # %% Assignment S1 - Time History
54 # Plot Time Variation of V1 at all nodes
55 fig1,axes = plt.subplots(2,1,constrained_layout=True)
56 axes = axes.reshape(-1)
57 axes[0].plot(t,u1,color='red',label=r'$y/\delta = 0.0028$ (Node 1)')
58 axes[0].plot(t,u2,color='orange',label=r'$y/\delta = 0.0203$ (Node 2)')
59 axes[0].plot(t,u3,color='blue',label=r'$y/\delta = 0.0364$ (Node 3)')
60 axes[0].plot(t,u4,color='green',label=r'$y/\delta = 0.0645$ (Node 4)')
61 axes[0].plot(t,u5,color='black',label=r'$y/\delta = 0.2$ (Node 5)')
```

```

62 axes[1].plot(t,u1,color='red',label=r'$y/\delta = 0.0028 \text{ (Node 1)}$')
63 axes[1].plot(t,u2,color='orange',label=r'$y/\delta = 0.0203 \text{ (Node 2)}$')
64 axes[1].plot(t,u3,color='blue',label=r'$y/\delta = 0.0364 \text{ (Node 3)}$')
65 axes[1].plot(t,u4,color='green',label=r'$y/\delta = 0.0645 \text{ (Node 4)}$')
66 axes[1].plot(t,u5,color='black',label=r'$y/\delta = 0.2 \text{ (Node 5)}$')
67 axes[1].set_xlim(6,7)
68 axes[1].set_ylim(10,27)
69 for ax in axes:
70     ax.set_xlabel(r'Time ($t$)')
71     ax.set_ylabel(r'$\overline{u}/u_{\tau}$')
72     ax.legend(fontsize=16,loc='best')
73     ax.tick_params(labelsize='medium')
74     ax.xaxis.set_minor_locator(AutoMinorLocator())
75     ax.yaxis.set_minor_locator(AutoMinorLocator())
76     ax.grid(True, linestyle='-.')
77 fig1.suptitle(r'Time variation of  $\overline{u}$  at various nodes', fontsize=22)
78
79 # Plot Time Variation of V3 at all nodes
80 fig2,axes = plt.subplots(2,1,constrained_layout=True)
81 axes = axes.reshape(-1)
82 axes[0].plot(t,w1,color='red',label=r'$y/\delta = 0.0028 \text{ (Node 1)}$')
83 axes[0].plot(t,w2,color='orange',label=r'$y/\delta = 0.0203 \text{ (Node 2)}$')
84 axes[0].plot(t,w3,color='blue',label=r'$y/\delta = 0.0364 \text{ (Node 3)}$')
85 axes[0].plot(t,w4,color='green',label=r'$y/\delta = 0.0645 \text{ (Node 4)}$')
86 axes[0].plot(t,w5,color='black',label=r'$y/\delta = 0.2 \text{ (Node 5)}$')
87 axes[1].plot(t,w1,color='red',label=r'$y/\delta = 0.0028 \text{ (Node 1)}$')
88 axes[1].plot(t,w2,color='orange',label=r'$y/\delta = 0.0203 \text{ (Node 2)}$')
89 axes[1].plot(t,w3,color='blue',label=r'$y/\delta = 0.0364 \text{ (Node 3)}$')
90 axes[1].plot(t,w4,color='green',label=r'$y/\delta = 0.0645 \text{ (Node 4)}$')
91 axes[1].plot(t,w5,color='black',label=r'$y/\delta = 0.2 \text{ (Node 5)}$')
92 axes[1].set_xlim(6,7)
93 axes[1].set_ylim(-3,2)
94 for ax in axes:
95     ax.set_xlabel(r'Time ($t$)')
96     ax.set_ylabel(r'$\overline{w}/u_{\tau}$')
97     ax.legend(fontsize=16,loc='best')
98     ax.tick_params(labelsize='medium')
99     ax.xaxis.set_minor_locator(AutoMinorLocator())
100    ax.yaxis.set_minor_locator(AutoMinorLocator())
101    ax.grid(True, linestyle='-.')
102 fig2.suptitle(r'Time variation of  $\overline{w}$  at various nodes', fontsize=22)
103
104 # Compute Auto-Correlation
105 u1_fluct = u1 - np.mean(u1)
106 u2_fluct = u2 - np.mean(u2)
107 u3_fluct = u3 - np.mean(u3)
108 u4_fluct = u4 - np.mean(u4)
109 u5_fluct = u5 - np.mean(u5)
110
111 B11_1 = np.correlate(u1_fluct,u1_fluct,'full')
112 nmax = np.argmax(B11_1)
113 magmax = np.max(B11_1)
114 norm_B11_1 = B11_1[nmax:]/magmax
115
116 B11_2 = np.correlate(u2_fluct,u2_fluct,'full')
117 nmax = np.argmax(B11_2)
118 magmax = np.max(B11_2)
119 norm_B11_2 = B11_2[nmax:]/magmax
120
121 B11_3 = np.correlate(u3_fluct,u3_fluct,'full')
122 nmax = np.argmax(B11_3)
123 magmax = np.max(B11_3)
124 norm_B11_3 = B11_3[nmax:]/magmax
125
126 B11_4 = np.correlate(u4_fluct,u4_fluct,'full')
127 nmax = np.argmax(B11_4)

```

```
128 magmax = np.max(B11_4)
129 norm_B11_4 = B11_4[nmax:]/magmax
130
131 B11_5 = np.correlate(u5_fluct,u5_fluct,'full')
132 nmax = np.argmax(B11_5)
133 magmax = np.max(B11_5)
134 norm_B11_5 = B11_5[nmax:]/magmax
135
136 # Plotting Auto-Corelation
137 imax = 2000
138 fig3,axes = plt.subplots(2,1,constrained_layout=True)
139 axes[0].plot(t[0:imax],norm_B11_1[0:imax],color='red',label=r'$y/\delta$ = 0.0028 (Node 1)')
140 axes[0].plot(t[0:imax],norm_B11_2[0:imax],color='orange',label=r'$y/\delta$ = 0.0203 (Node 2)')
141 axes[0].plot(t[0:imax],norm_B11_3[0:imax],color='blue',label=r'$y/\delta$ = 0.0364 (Node 3)')
142 axes[1].plot(t[0:imax],norm_B11_4[0:imax],color='cyan',label=r'$y/\delta$ = 0.0645 (Node 4)')
143 axes[1].plot(t[0:imax],norm_B11_5[0:imax],color='black',label=r'$y/\delta$ = 0.2 (Node 5)')
144 for ax in axes:
145     ax.set_xlabel(r'Time ($t$)')
146     ax.set_ylabel(r'$B_{uu}$')
147     ax.legend(fontsize=16,loc='best')
148     ax.tick_params(labelsize='medium')
149     ax.xaxis.set_minor_locator(AutoMinorLocator())
150     ax.yaxis.set_minor_locator(AutoMinorLocator())
151     ax.grid(True, linestyle='-.')
152     ax.set_xlim([0,0.07])
153 fig3.suptitle(r'Auto-Corelation ($B_{uu}$) at various nodes', fontsize=22)
154
155 # Compute Integral Timescale
156 # All normalised auto-corelations go negative. Find the instance when it goes
157 # negative and integrate till that point..
158 norm_B11_1_M = norm_B11_1[0:np.argmax(norm_B11_1<=0)]
159 norm_B11_2_M = norm_B11_2[0:np.argmax(norm_B11_2<=0)]
160 norm_B11_3_M = norm_B11_3[0:np.argmax(norm_B11_3<=0)]
161 norm_B11_4_M = norm_B11_4[0:np.argmax(norm_B11_4<=0)]
162 norm_B11_5_M = norm_B11_5[0:62]
163 #norm_B11_5_M = norm_B11_5[0:np.argmax(norm_B11_5<=0)]
164
165 int_T_1 = np.trapz(norm_B11_1_M) * dt
166 int_T_2 = np.trapz(norm_B11_2_M) * dt
167 int_T_3 = np.trapz(norm_B11_3_M) * dt
168 int_T_4 = np.trapz(norm_B11_4_M) * dt
169 int_T_5 = np.trapz(norm_B11_5_M) * dt
```

For Assignment 2a - Section 1.2 to Section 1.7

```
1 import scipy.io as sio
2 import numpy as np
3 import matplotlib.pyplot as plt
4 from IPython import display
5 plt.rcParams.update({'font.size': 22})
6
7 # makes sure figures are updated when using ipython
8 display.clear_output()
9
10 dx=0.05
11 dz=0.025
12
13 ni=34
14 nj=49
15 nk=34
16
17 viscos=1./8000.
18
19 ##### read v_1 & transform v_1 to a 3D array (file 1)
20 # V1 Velocity
21 uvw = sio.loadmat('u1_sdes.mat')
22 ttu=uvw['u1_sdes']
23 u3d1= np.reshape(ttu,(nk,nj,ni))
24
25 uvw = sio.loadmat('u2_sdes.mat')
26 ttu=uvw['u2_sdes']
27 u3d2= np.reshape(ttu,(nk,nj,ni))
28
29 uvw = sio.loadmat('u3_sdes.mat')
30 ttu=uvw['u3_sdes']
31 u3d3= np.reshape(ttu,(nk,nj,ni))
32
33 uvw = sio.loadmat('u4_sdes.mat')
34 ttu=uvw['u4_sdes']
35 u3d4= np.reshape(ttu,(nk,nj,ni))
36
37 uvw = sio.loadmat('u5_sdes.mat')
38 ttu=uvw['u5_sdes']
39 u3d5= np.reshape(ttu,(nk,nj,ni))
40
41 uvw = sio.loadmat('u6_sdes.mat')
42 ttu=uvw['u6_sdes']
43 u3d6= np.reshape(ttu,(nk,nj,ni))
44
45 uvw = sio.loadmat('u7_sdes.mat')
46 ttv=uvw['u7_sdes']
47 u3d7= np.reshape(ttv,(nk,nj,ni))
48
49 uvw = sio.loadmat('u8_sdes.mat')
50 ttv=uvw['u8_sdes']
51 u3d8= np.reshape(ttv,(nk,nj,ni))
52
53 # V2 Velocity
54 uvw = sio.loadmat('v1_sdes.mat')
55 ttv=uvw['v1_sdes']
56 v3d1= np.reshape(ttv,(nk,nj,ni))
57
58 uvw = sio.loadmat('v2_sdes.mat')
59 ttv=uvw['v2_sdes']
60 v3d2= np.reshape(ttv,(nk,nj,ni))
61
62 uvw = sio.loadmat('v3_sdes.mat')
63 ttv=uvw['v3_sdes']
64 v3d3= np.reshape(ttv,(nk,nj,ni))
```

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

```

197
198 x=dx*ni
199 z=dz*nk
200
201 umean=np.mean(u3d, axis=(0,2))
202 vmean=np.mean(v3d, axis=(0,2))
203 wmean=np.mean(w3d, axis=(0,2))
204 temean=np.mean(te3d, axis=(0,2))
205 ommean=np.mean(om3d, axis=(0,2))
206
207 # face coordinates
208 yc = np.loadtxt("yc.dat")
209
210 # cell center coordinates
211 y= np.zeros(nj)
212 dy=np.zeros(nj)
213 for j in range (1,nj-1):
214     # dy = cell width
215     dy[j]=yc[j]-yc[j-1]
216     y[j]=0.5*(yc[j]+yc[j-1])
217
218 y[nj-1]=yc[nj-1]
219 tauw=viscos*umean[1]/y[1]
220 ustar=tauw**0.5
221 yplus=y*ustar/viscos
222
223 DNS=np.genfromtxt("LM_Channel_5200_mean_prof.dat", dtype=None, comments="%")
224 y_DNS=DNS[:,0]
225 yplus_DNS=DNS[:,1]
226 u_DNS=DNS[:,2]
227
228 DNS=np.genfromtxt("LM_Channel_5200_vel_fluc_prof.dat", dtype=None, comments="%")
229 u2_DNS=DNS[:,2]
230 v2_DNS=DNS[:,3]
231 w2_DNS=DNS[:,4]
232 uv_DNS=DNS[:,5]
233
234 k_DNS=0.5*(u2_DNS+v2_DNS+w2_DNS)
235
236 # find equi.distant DNS cells in log-scale
237 xx=0.
238 jDNS=[1]*40
239 for i in range (0,40):
240     i1 = (np.abs(10.*xx-yplus_DNS)).argmin()
241     jDNS[i]=int(i1)
242     xx=xx+0.2
243
244 # %% Assignment S2 - Mean Velocity Profile
245 fig1,axes = plt.subplots(1,1,constrained_layout=True)
246 axes.semilogx(y,umean/ustar,'b--',label='DES Data')
247 axes.semilogx(y_DNS[jDNS],u_DNS[jDNS],'r-o',label='DNS Data')
248 axes.semilogx(0.0175*np.ones(20),np.linspace(0,25,20),'k--',label=r'$y/\delta$ = 0.0175')
249 axes.set_xlabel(r"$y/\delta$")
250 axes.set_ylabel(r"$U^+$")
251 axes.legend(fontsize=16,loc='best')
252 axes.tick_params(labelsize='medium')
253 axes.grid(True, linestyle='-.')
254 fig1.suptitle(r'Time Averaged Velocity ($\overline{v_1}'$, fontsize=22)
255
256 # %% Assignment S3 - Resolved Turbulent Shear Stress
257 uv_mean = np.mean((u3d-umean[:,None,None])*(v3d-vmean[:,None,None]),axis=(0,2))
258
259 fig2,axes = plt.subplots(1,1,constrained_layout=True)
260 axes.semilogx(y,uv_mean,'b--',label='DES Data - Resolved Turbulent Shear Stress')
261 axes.semilogx(y_DNS[jDNS],uv_DNS[jDNS],'r-o',label='DNS Data - Total Turbulent Shear

```

```

    Stress')
262 axes.semilogx(0.0175*np.ones(20),np.linspace(-1,0,20),'k--',label=r'$y/\delta$ = 0.0175')
263 axes.set_xlabel(r'$y/\delta$')
264 axes.set_ylabel(r'Shear Stress $\overline{v_1}^{\prime}\overline{v_2}^{\prime}$')
265 axes.legend(fontsize=16,loc='best')
266 axes.tick_params(labelsize='medium')
267 axes.grid(True, linestyle='-.')
268 fig2.suptitle(r'Resolved Turbulent Shear Stress', fontsize=22)
269
270 # %% Assignment S4 - Turbulent Kinetic Energy
271 uu_mean = np.mean((u3d-umean[:,None])**2, axis=(0,2))
272 vv_mean = np.mean((v3d-vmean[:,None])**2, axis=(0,2))
273 ww_mean = np.mean((w3d-wmean[:,None])**2, axis=(0,2))
274
275 #uu_mean = np.mean(u3d*u3d, axis=(0,2)) - (umean*umean)
276 #vv_mean = np.mean(v3d*v3d, axis=(0,2)) - (vmean*vmean)
277 #ww_mean = np.mean(w3d*w3d, axis=(0,2)) - (wmean*wmean)
278
279 ke_res = 0.5*(uu_mean+vv_mean+ww_mean)
280 ke_res_DNS = 0.5*(u2_DNS+v2_DNS+w2_DNS)
281
282 fig3,axes = plt.subplots(1,1,constrained_layout=True)
283 axes.semilogx(y,temean,'b--',label='DES Data - Modeled Turbulent Kinetic Energy')
284 axes.semilogx(y,ke_res,'r-o',label='DES Data - Resolved Turbulent Kinetic Energy')
285 axes.semilogx(y,ke_res+temean,'k-o',label='DES Data - Total Turbulent Kinetic Energy')
286 axes.semilogx(y_DNS,ke_res_DNS,'g-o',label='DNS Data - Turbulent Kinetic Energy')
287 axes.semilogx(0.0175*np.ones(10),np.linspace(0,6,10),'k--',label=r'$y/\delta$ = 0.0175')
288 axes.set_xlabel(r'$y/\delta$')
289 axes.set_ylabel('TKE')
290 axes.legend(fontsize=16,loc='best')
291 axes.tick_params(labelsize='medium')
292 axes.grid(True, linestyle='-.')
293 fig3.suptitle(r'Turbulent Kinetic Energy', fontsize=22)
294
295 # %% Assignment S5 - Modeled Turbulent Shear Stress
296 turb_visc = te3d/om3d
297 dudx,dudy,dudz = np.gradient(u3d,dx,y,dz)
298 dvdx,dvdy,dvdz = np.gradient(v3d,dx,y,dz)
299 dwdx,dwdy,dwdz = np.gradient(w3d,dx,y,dz)
300 tau_12 = -turb_visc * (dudy+dvdx)
301 tau_12_mean = np.mean(tau_12, axis=(0,2))
302 tau_12_mean[0] = 0
303
304 fig4,axes = plt.subplots(1,1,constrained_layout=True)
305 axes.semilogx(y,uv_mean,'b--',label='DES Data - Resolved Turbulent Shear Stress')
306 axes.semilogx(y,tau_12_mean,'g--',label='DES Data - Modeled Turbulent Shear Stress')
307 axes.semilogx(y_DNS[jDNS],uv_DNS[jDNS],'r-o',label='DNS Data - Total Turbulent Shear Stress')
308 axes.semilogx(0.0175*np.ones(20),np.linspace(-1,0,20),'k--',label=r'$y/\delta$ = 0.0175')
309 axes.set_xlabel(r'$y/\delta$')
310 axes.set_ylabel(r'Shear Stress $\overline{v_1}^{\prime}\overline{v_2}^{\prime}$')
311 axes.legend(fontsize=16,loc='best')
312 axes.tick_params(labelsize='medium')
313 axes.grid(True, linestyle='-.')
314 fig4.suptitle(r'Comparision of Shear Stress', fontsize=22)
315
316 fig5,axes = plt.subplots(1,1,constrained_layout=True)
317 axes.semilogx(y,uv_mean+tau_12_mean,'b-o',label='DES Data')
318 axes.semilogx(y_DNS[jDNS],uv_DNS[jDNS],'r-o',label='DNS Data')
319 axes.semilogx(0.0175*np.ones(20),np.linspace(-1,0,20),'k--',label=r'$y/\delta$ = 0.0175')
320 axes.set_xlabel(r'$y/\delta$')

```

```

321 axes.set_ylabel(r'Shear Stress $\overline{v_1}'\prime$\overline{v_2}'\prime$')
322 axes.legend(fontsize=16, loc='best')
323 axes.tick_params(labelsize='medium')
324 axes.grid(True, linestyle='-.')
325 fig5.suptitle(r'Total Turbulent Shear Stress', fontsize=22)
326
327 # %% Assignment S6 - Location of Interface in DES and DDES
328 C_DES_SA = 0.65
329 delta_SA = C_DES_SA * np.maximum(dx*np.ones([49,1]), dy.reshape(49,1), dz*np.ones([49,1]))
330
331 Int_SA = y[(y.reshape(49,1)>=delta_SA).argmax()]
332 print('Switch for SA-DES model occurs at y+ = '+str(np.around(Int_SA,3)))
333
334 C_DES_SST = 0.61
335 beta_star = 0.09
336 delta_SST = C_DES_SST * np.maximum(dx*np.ones([49,1]), dy.reshape(49,1), dz*np.ones([49,1]))
337 Lt = ((temean**0.5)/ommean)/beta_star
338 term1 = Lt.reshape(49,1)/(delta_SST)
339 term1[0] = 1e-10
340
341 Int_SST = y[(np.ones([49,1])>=term1).argmin()]
342 print('Switch for SST-DES model occurs at y+ = '+str(np.around(Int_SST,3)))
343
344 term2 = 2*(Lt[1:]/y[1:])
345 term3 = (500*viscos)/(ommean[1:]*y[1:]**2)
346 eta = np.insert(np.maximum(term2,term3), 0, [1e-10])
347 F2 = np.tanh(eta**2)
348 F_DDES = np.maximum(term1*(1-F2.reshape(49,1)), 1)
349 Int_DDES = y[(np.ones([49,1])>=(term1*(1-F2.reshape(49,1))).argmin()]
350 print('Switch for DDES model occurs at y+ = '+str(np.around(Int_DDES,3)))
351
352 fig6, axes = plt.subplots(1,1, constrained_layout=True)
353 axes.plot(umean, y, label=r'DES data : Mean velocity profile $\overline{u}$')
354 axes.plot(u_DNS, y_DNS, label=r'DNS data : Mean velocity profile $\overline{u}$')
355 axes.plot(np.linspace(0,30,20), 0.0175*np.ones(20), 'k--', lw=2, label=r'S-DES model')
356 axes.plot(np.linspace(0,30,20), Int_SA*np.ones(20), 'b--', lw=2, label=r'SA-DES model')
357 axes.plot(np.linspace(0,30,20), Int_SST*np.ones(20), 'y--', lw=2, label=r'SST-DES model')
358 axes.plot(np.linspace(0,30,20), Int_DDES*np.ones(20), 'r--', lw=2, label=r'DDES model')
359 axes.set_ylabel(r'$y/\delta$')
360 axes.legend(fontsize=16, loc='best')
361 axes.tick_params(labelsize='medium')
362 axes.grid(True, linestyle='-.')
363 fig6.suptitle(r'Location of Interfaces for various models', fontsize=22)
364
365 # %% Assignment S7 - SAS Turbulent Length Scale
366 # 1D Length Scale
367 dua_dy = np.gradient(umean, y)
368 dua2_dy = np.gradient(dua_dy, y)
369 DNSdua_dy = np.gradient(u_DNS, y_DNS)
370 DNSdua2_dy = np.gradient(DNSdua_dy, y_DNS)
371 # Von-karman constant
372 kappa = 0.4
373
374 LVK_1D = kappa * np.abs(dua_dy/dua2_dy)
375 DNSLVK_1D = kappa * np.abs(DNSdua_dy/DNSdua2_dy)
376
377 # 3D Length Scale
378 du2_dx dx, du2_dy dx, du2_dz dx = np.gradient(dudx, dx, y, dz)
379 dv2_dx dx, dv2_dy dx, dv2_dz dx = np.gradient(dvdx, dx, y, dz)
380 dw2_dx dx, dw2_dy dx, dw2_dz dx = np.gradient(dw dx, dx, y, dz)
381
382 du2_dy dx, du2_dy dy, du2_dy dz = np.gradient(dudy, dx, y, dz)
383 dv2_dy dx, dv2_dy dy, dv2_dy dz = np.gradient(dvdy, dx, y, dz)
384 dw2_dy dx, dw2_dy dy, dw2_dy dz = np.gradient(dwdy, dx, y, dz)

```

```

385 du2_dzdx ,du2_dzdy ,du2_dzdz = np.gradient(dudz ,dx ,y ,dz)
386 dv2_dzdx ,dv2_dzdy ,dv2_dzdz = np.gradient(dvdz ,dx ,y ,dz)
387 dw2_dzdx ,dw2_dzdy ,dw2_dzdz = np.gradient(dwdz ,dx ,y ,dz)
388
389 S11 = 0.5*(2*dudx)
390 S12 = 0.5*(dudy+dwdx)
391 S13 = 0.5*(dudz+dwdx)
392 S22 = 0.5*(2*dvdy)
393 S23 = 0.5*(dvdz+dwdy)
394 S33 = 0.5*(2*dwdz)
395
396 S = np.sqrt(2*((S11**2)+(S22**2)+(S33**2)+(2*(S12**2))+(2*(S13**2))+(2*(S23**2))))
397
398 U1 = np.sqrt((du2_dx dx*du2_dx dx)+(du2_dy dy*du2_dy dy)+(du2_dz dz*du2_dz dz) +
399 (dv2_dx dx*dv2_dx dx)+(dv2_dy dy*dv2_dy dy)+(dv2_dz dz*dv2_dz dz) +
400 (dw2_dx dx*dw2_dx dx)+(dw2_dy dy*dw2_dy dy)+(dw2_dz dz*dw2_dz dz))
401 U2 = np.sqrt((du2_dx dx**2)+(du2_dx dy**2)+(du2_dx dz**2)+(du2_dy dx**2)+(du2_dy dy**2) +
402 (du2_dy dz**2)+(du2_dz dx**2)+(du2_dz dy**2)+(du2_dz dz**2)+(dv2_dx dx**2)+(dv2_dy dx**2) +
403 (dv2_dy dy**2)+(dv2_dz dx**2)+(dv2_dz dy**2)+(dv2_dz dz**2)+(dw2_dx dx**2)+(dw2_dy dx**2) +
404 (dw2_dy dy**2)+(dw2_dz dx**2)+(dw2_dz dy**2)+(dw2_dz dz**2)+(dw2_dx dy**2)+(dw2_dy dz**2) +
405 (dw2_dy dz**2)+(dw2_dz dx**2)+(dw2_dz dy**2)+(dw2_dz dz**2))
406
407 LVK_3D = kappa*np.abs(S/U1)
408 LVK_3D_2 = kappa*np.abs(S/U2)
409 delta = (dx*dy*dx)**(1/3)
410
411 fig7, axes = plt.subplots(1,1,constrained_layout=True)
412 axes.plot(y[:,],LVK_1D[:,],'k--',label=r'Von-Karman Length Scale 1D')
413 axes.plot(y[:,],np.mean(LVK_3D[:,],axis=(0,2))[:,],'r--',label=r'Von-Karman Length Scale 3D')
414 axes.plot(y[:,],np.mean(LVK_3D_2[:,],axis=(0,2))[:,],'g--',label=r'Von-Karman Length Scale 3D - Method 2')
415 #axes.plot(y[:,],delta[:])
416 #axes.plot(y[:,],dy[:])
417 axes.set_ylabel(r"Length Scale")
418 axes.set_xlabel(r"$y/\delta$")
419 axes.legend(fontsize=16,loc='best')
420 axes.tick_params(labelsize='medium')
421 axes.grid(True, linestyle='-.')
422 fig7.suptitle(r'Von-Karman Length Scale', fontsize=22)
423
424 fig7a = plt.figure()
425 axes1 = fig7a.add_axes([0.1, 0.1, 0.8, 0.8]) # main axes
426 axes2 = fig7a.add_axes([0.2, 0.55, 0.3, 0.3]) # inset axes
427 # Larger Figure Axes 1
428 axes1.plot(y[:,],LVK_1D[:,],'k-o',label=r'DES data - Von-Karman Length Scale 1D')
429 axes1.plot(y_DNS[4:],DNSLVK_1D[4:],'r-',label=r'DNS data - Von-Karman Length Scale 1D')
430 axes1.set_ylabel(r"Length Scale")
431 axes1.set_xlabel(r"$y/\delta$")
432 axes1.legend(fontsize=16,loc='upper right')
433 axes1.tick_params(labelsize='medium')
434 axes1.grid(True, linestyle='-.')
435 axes1.set_title(r'Von-Karman Length Scale', fontsize=22)
436 # Insert Figure Axes 2
437 axes2.plot(y[:,],LVK_1D[:,],'k-o',label=r'DES data - Von-Karman Length Scale 1D')
438 axes2.plot(y_DNS[4:],DNSLVK_1D[4:],'r-',label=r'DNS data - Von-Karman Length Scale 1D')
439 axes2.set_ylabel(r"Length Scale", fontsize=12)
440 axes2.set_xlabel(r"$y/\delta$", fontsize=12)
441 axes2.tick_params(labelsize='x-small')
442 axes2.grid(True, linestyle='-.')
443 axes2.set_xlim([-0.01,0.05])
444 axes2.set_ylim([-0.1,0.2])
445 axes2.set_title(r'Zoomed near the wall', fontsize=12)
446
447
448

```

```
449 # %% Assignment S8 - Estimate of Resolution
450 ustar_x = -((dx**2)/4) * (du2_dx dx + dv2_dx dx + dw2_dx dx)
451 vstar_x = -((dy[None,:,None]**2)/4) * (du2_dx dx + dv2_dx dx + dw2_dx dx)
452 wstar_x = -((dz**2)/4) * (du2_dx dx + dv2_dx dx + dw2_dx dx)
453
454 ustar_y = -((dx**2)/4) * (du2_dy dy + dv2_dy dy + dw2_dy dy)
455 vstar_y = -((dy[None,:,None]**2)/4) * (du2_dy dy + dv2_dy dy + dw2_dy dy)
456 wstar_y = -((dz**2)/4) * (du2_dy dy + dv2_dy dy + dw2_dy dy)
457
458 ustar_z = -((dx**2)/4) * (du2_dz dz + dv2_dz dz + dw2_dz dz)
459 vstar_z = -((dy[None,:,None]**2)/4) * (du2_dz dz + dv2_dz dz + dw2_dz dz)
460 wstar_z = -((dz**2)/4) * (du2_dz dz + dv2_dz dz + dw2_dz dz)
461
462 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))))
463 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))))
464 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))))
465
466 fig8, axes = plt.subplots(1,1, constrained_layout=True)
467 axes.semilogx(y,Error_x,'r-o',label=r'Anisotropic Error in X-direction')
468 axes.semilogx(y,Error_y,'k-s',label=r'Anisotropic Error in Y-direction')
469 axes.semilogx(y,Error_z,'g-*',label=r'Anisotropic Error in Z-direction')
470 axes.set_xlabel(r"$y/\delta$")
471 axes.set_ylabel(r"Error")
472 axes.legend(fontsize=16, loc='best')
473 axes.tick_params(labelsize='medium')
474 axes.grid(True, linestyle='-.')
475 fig8.suptitle(r'Anisotropic Error along Coordinate Directions', fontsize=22)
```

For Assignment 2b : Section 2.1 to 2.3

```
1 import scipy.io as sio
2 import numpy as np
3 import matplotlib.pyplot as plt
4 from mpl_toolkits.axes_grid1.inset_locator import inset_axes
5 from dphidx_dy import dphidx_dy
6 from matplotlib.ticker import (MultipleLocator, FormatStrFormatter,
7                                AutoMinorLocator)
8 plt.rcParams.update({'font.size': 22})
9
10 re = 9.36e+5
11 viscos = 1/re
12
13 xy_hump_fine = np.loadtxt("xy_hump.dat")
14 x=xy_hump_fine[:,0]
15 y=xy_hump_fine[:,1]
16
17 ni=314
18 nj=122
19
20 nim1=ni-1
21 njm1=nj-1
22
23 # Read Data File (from paper)
24 vectz=np.genfromtxt("vectz_aiaa_paper.dat",comments="#")
25 ntstep=vectz[0]
26 n=len(vectz)
27
28 # Read Data File (from journal)
29 vectz_aiaa = np.genfromtxt("vectz_aiaa_journal.dat",comments="#")
30
31 #           write(48,*)uvec(i,j)
32 #           write(48,*)vvec(i,j)
33 #           write(48,*)dummy(i,j)
34 #           write(48,*)uvec2(i,j)
35 #           write(48,*)vvec2(i,j)
36 #           write(48,*)wvec2(i,j)
37 #           write(48,*)uvvec(i,j)
38 #           write(48,*)p2D(i,j)
39 #           write(48,*)rk2D(i,j)
40 #           write(48,*)vis2D(i,j)
41 #           write(48,*)dissp2D(i,j)
42 #           write(48,*)uvturb(i,j)
43
44
45
46 nn=12
47 nst=0
48 iu=range(nst+1,n,nn)
49 iv=range(nst+2,n,nn)
50 ifk=range(nst+3,n,nn)
51 iuu=range(nst+4,n,nn)
52 ivv=range(nst+5,n,nn)
53 iww=range(nst+6,n,nn)
54 iuv=range(nst+7,n,nn)
55 ip=range(nst+8,n,nn)
56 ik=range(nst+9,n,nn)
57 ivis=range(nst+10,n,nn)
58 idiss=range(nst+11,n,nn)
59 iuv_model=range(nst+12,n,nn)
60
61 # From paper
62 u=vectz[iu]/ntstep
63 v=vectz[iv]/ntstep
64 fk=vectz[ifk]/ntstep
```

```
65 uu=vectz[iuu]/ntstep
66 vv=vectz[ivv]/ntstep
67 ww=vectz[iww]/ntstep
68 uv=vectz[iuv]/ntstep
69 p=vectz[ip]/ntstep
70 k_model=vectz[ik]/ntstep
71 vis=vectz[ivis]/ntstep
72 diss=vectz[idiss]/ntstep
73 uv_model=vectz[iuv_model]/ntstep
74 # From journal
75 u1=vectz_aiaa[iu]/ntstep
76 v1=vectz_aiaa[iv]/ntstep
77 fk1=vectz_aiaa[ifk]/ntstep
78 uu1=vectz_aiaa[iuu]/ntstep
79 vv1=vectz_aiaa[ivv]/ntstep
80 ww1=vectz_aiaa[iww]/ntstep
81 uv1=vectz_aiaa[iuv]/ntstep
82 p1=vectz_aiaa[ip]/ntstep
83 k_model1=vectz_aiaa[ik]/ntstep
84 vis1=vectz_aiaa[ivis]/ntstep
85 diss1=vectz_aiaa[idiss]/ntstep
86 uv1_model=vectz_aiaa[iuv_model]/ntstep
87
88
89 # uu is total inst. velocity squared. Hence the resolved turbulent resolved stresses are
     obtained as
90 # From paper
91 uu=uu-u**2
92 vv=vv-v**2
93 uv=uv-u*v
94 # From journal
95 uu1=uu1-u**2
96 vv1=vv1-v**2
97 uv1=uv1-u*v
98
99 # From paper
100 p_2d=np.reshape(p,(ni,nj))
101 u_2d=np.reshape(u,(ni,nj))
102 v_2d=np.reshape(v,(ni,nj))
103 fk_2d=np.reshape(fk,(ni,nj))
104 uu_2d=np.reshape(uu,(ni,nj))
105 uv_2d=np.reshape(uv,(ni,nj))
106 vv_2d=np.reshape(vv,(ni,nj))
107 ww_2d=np.reshape(ww,(ni,nj))
108 uv_model_2d=np.reshape(uv_model,(ni,nj))
109 k_model_2d=np.reshape(k_model,(ni,nj))
110 vis_2d=np.reshape(vis,(ni,nj)) #this is to total viscosity, i.e. vis_tot=vis+vis_turb
111 diss_2d=np.reshape(diss,(ni,nj)) #this is to total viscosity, i.e. vis_tot=vis+vis_turb
112 x_2d=np.transpose(np.reshape(x,(nj,ni)))
113 y_2d=np.transpose(np.reshape(y,(nj,ni)))
114 # From journal
115 p1_2d=np.reshape(p1,(ni,nj))
116 u1_2d=np.reshape(u1,(ni,nj))
117 v1_2d=np.reshape(v1,(ni,nj))
118 fk1_2d=np.reshape(fk1,(ni,nj))
119 uu1_2d=np.reshape(uu1,(ni,nj))
120 uv1_2d=np.reshape(uv1,(ni,nj))
121 vv1_2d=np.reshape(vv1,(ni,nj))
122 ww1_2d=np.reshape(ww1,(ni,nj))
123 uv1_model_2d=np.reshape(uv1_model,(ni,nj))
124 k_model1_2d=np.reshape(k_model1,(ni,nj))
125 vis1_2d=np.reshape(vis1,(ni,nj)) #this is to total viscosity, i.e. vis_tot=vis+vis_turb
126 diss1_2d=np.reshape(diss1,(ni,nj)) #this is to total viscosity, i.e. vis_tot=vis+
     vis_turb
127
128 # set fk_2d=1 at upper boundary
```

```
129 fk_2d[:,nj-1]=fk_2d[:,nj-2]
130 fk1_2d[:,nj-1]=fk_2d[:,nj-2]
131
132 x065_off=np.genfromtxt("x065_off.dat",comments="#")
133 x066_off=np.genfromtxt("x066_off.dat",comments="#")
134 x080_off=np.genfromtxt("x080_off.dat",comments="#")
135 x090_off=np.genfromtxt("x090_off.dat",comments="#")
136 x100_off=np.genfromtxt("x100_off.dat",comments="#")
137 x110_off=np.genfromtxt("x110_off.dat",comments="#")
138 x120_off=np.genfromtxt("x120_off.dat",comments="#")
139 x130_off=np.genfromtxt("x130_off.dat",comments="#")
140
141 # face coordinates for dphidx_dy function
142 #xf2d = np.zeros((ni,nj))
143 #yf2d = np.zeros((ni,nj))
144 xf2d = 0.5 * (x_2d[1:-2,:] + x_2d[2:-1,:])
145 yf2d = 0.5 * (y_2d[:,1:-2] + y_2d[:,2:-1])
146 #xf2d[] = x_2d[]
147 #yf2d[] = y_2d[]
148
149 # the function dphidx_dy wants x and y arrays to be one cell smaller than u2d. Hence I
     take away the last row and column below
150 x_2d_new=np.delete(x_2d,-1,0)
151 x_2d_new=np.delete(x_2d_new,-1,1)
152 y_2d_new=np.delete(y_2d,-1,0)
153 y_2d_new=np.delete(y_2d_new,-1,1)
154 # compute the gradient
155 dudx,dudy=dphidx_dy(x_2d_new,y_2d_new,u_2d)
156 dvdx,dvdy=dphidx_dy(x_2d_new,y_2d_new,v_2d)
157
158 ****
159 # Mesh for visualization
160 a = np.transpose(x_2d)
161 b = np.transpose(y_2d)
162 fig16 = plt.figure()
163 axes = fig16.add_axes([0.1,0.1,0.8,0.8])
164 axes.plot(x_2d,y_2d,color='black')
165 axes.plot(a,b,color='black')
166 axes.tick_params(axis="x", labelsize=15)
167 axes.tick_params(axis="y", labelsize=15)
168 axes.set_xlabel('x [m]', fontsize=20)
169 axes.set_ylabel('y [m]', fontsize=20)
170 axes.set_title('Computational mesh in the domain', fontsize=24)
171
172 ****
173 # plot u
174 fig1 = plt.figure("Figure 1")
175 plt.clf() #clear the figure
176 xx=0.65;
177 i1 = (np.abs(xx-x_2d[:,1])).argmin() # find index which closest fits xx
178 plt.plot(u_2d[i1,:],y_2d[i1,:]-y_2d[i1,0],'b-')
179 plt.plot(x065_off[:,2],x065_off[:,1]-y_2d[i1,0],'bo')
180 plt.xlabel("$U$")
181 plt.ylabel("$y-y_{wall}$")
182 plt.title("$x=0.65$")
183 plt.axis([0, 1.3, 0, 0.2])
184 #plt.savefig('u065_hump_python.eps')
185
186 ****
187 # plot vv
188 fig2 = plt.figure("Figure 2")
189 plt.clf() #clear the figure
190 xx=0.65;
191 i1 = (np.abs(xx-x_2d[:,1])).argmin() # find index which closest fits xx
192 plt.plot(vv_2d[i1,:],y_2d[i1,:]-y_2d[i1,0],'b-')
193 plt.plot(x065_off[:,5],x065_off[:,1]-y_2d[i1,0],'bo')
```

```

194 plt.xlabel("$\overline{v'v'})$")
195 plt.ylabel("$y-y_{wall}$")
196 plt.title("$x=0.65$")
197 plt.axis([0, 0.01, 0, 0.2])
198 #plt.savefig('vv065_hump_python.eps')
199
200 ****
201 # plot uu
202 fig3 = plt.figure("Figure 3")
203 plt.clf() #clear the figure
204 xx=0.65;
205 i1 = (np.abs(xx-x_2d[:,1])).argmin() # find index which closest fits xx
206 plt.plot(uu_2d[i1,:],y_2d[i1,:]-y_2d[i1,0],'b-')
207 plt.plot(x065_off[:,4],x065_off[:,1]-y_2d[i1,0],'bo')
208 plt.xlabel("$\overline{u'u'})$")
209 plt.ylabel("$y-y_{wall}$")
210 plt.title("$x=0.65$")
211 plt.axis([0, 0.01, 0, 0.2])
212 #plt.savefig('uu065_hump_python.eps')
213
214 ##### contour plot
215 fig4 = plt.figure("Figure 4")
216 plt.clf() #clear the figure
217 plt.contourf(x_2d,y_2d,uu_2d, 50)
218 plt.xlabel("$x$")
219 plt.ylabel("$y$")
220 plt.clim(0,0.05)
221 plt.axis([0.6,1.5,0,1])
222 plt.title("contour $\overline{u'u'})$")
223 #plt.savefig('piso_python.eps')
224
225 ##### vector plot
226 fig5 = plt.figure("Figure 5")
227 plt.clf() #clear the figure
228 k=6# plot every forth vector
229 ss=3.2 #vector length
230 plt.quiver(x_2d[:,::k,::k],y_2d[:,::k,::k],u_2d[:,::k,::k],v_2d[:,::k,::k],width=0.01)
231 plt.xlabel("$x$")
232 plt.ylabel("$y$")
233 plt.axis([0.6,1.5,0,1])
234 plt.title("vector plot")
235 #plt.savefig('vect_python.eps')
236
237 # %% Assignment T1 - Discretization Schemes
238 # Paper Data [203]
239 ## Plot vv
240 xx = np.array([0,0.65,0.8,0.9,1,1.1,1.2,1.3])
241 i1 = list(map(lambda m : (np.abs(m-x_2d[:,1])).argmin(),xx))
242 fig6,axes = plt.subplots(2,4,constrained_layout=True)
243 axes = axes.reshape(-1)
244 axes[0].plot(vv_2d[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0],'b-o',ms=4,label='x = 0 m')
245 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')
246 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')
247 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')
248 axes[4].plot(vv_2d[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0],'m-+',ms=4,label='x = 1 m')
249 axes[5].plot(vv_2d[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0],'y-*',ms=4,label='x = 1.1 m')
250 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')
251 axes[7].plot(vv_2d[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0],label='x = 1.3 m')
252 for ax in axes:
253     ax.xaxis.set_minor_locator(AutoMinorLocator())
254     ax.yaxis.set_minor_locator(AutoMinorLocator())
255     ax.legend(fontsize=14,loc='best')
256     ax.tick_params(labelsize='medium')
257     ax.grid(True, linestyle='-.')
258     ax.set_ylabel("$y-y_{wall}$")
259     if (ax==axes[4] or ax==axes[5] or ax==axes[6] or ax==axes[7]):
```

```

260         ax.set_xlabel("$\overline{v'v'})$")
261 fig6.suptitle("$\overline{v'v'})$ at various locations", fontsize=22)
262
263 ## Plot uv
264 fig7, axes = plt.subplots(2,4, constrained_layout=True)
265 axes = axes.reshape(-1)
266 axes[0].plot(uv_2d[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0], 'b-o', ms=4, label='x = 0 m')
267 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')
268 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')
269 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')
270 axes[4].plot(uv_2d[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0], 'm+', ms=4, label='x = 1 m')
271 axes[5].plot(uv_2d[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0], 'y-*', ms=4, label='x = 1.1 m')
272 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')
273 axes[7].plot(uv_2d[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0], label='x = 1.3 m')
274 for ax in axes:
275     ax.xaxis.set_minor_locator(AutoMinorLocator())
276     ax.yaxis.set_minor_locator(AutoMinorLocator())
277     ax.legend(fontsize=14, loc='best')
278     ax.tick_params(labelsize='medium')
279     ax.grid(True, linestyle='-.')
280     ax.set_ylabel("$y-y_{wall}$")
281     if (ax==axes[4] or ax==axes[5] or ax==axes[6] or ax==axes[7]):
282         ax.set_xlabel("$\overline{u'v'})$")
283 fig7.suptitle("$\overline{u'v'})$ at various locations", fontsize=22)
284
285 # Journal Data [169]
286 ## Plot vv
287 fig8, axes = plt.subplots(2,4, constrained_layout=True)
288 axes = axes.reshape(-1)
289 axes[0].plot(vv1_2d[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0], 'b-o', ms=4, label='x = 0 m')
290 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')
291 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')
292 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')
293 axes[4].plot(vv1_2d[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0], 'm+', ms=4, label='x = 1 m')
294 axes[5].plot(vv1_2d[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0], 'y-*', ms=4, label='x = 1.1 m')
295 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')
296 axes[7].plot(vv1_2d[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0], label='x = 1.3 m')
297 for ax in axes:
298     ax.xaxis.set_minor_locator(AutoMinorLocator())
299     ax.yaxis.set_minor_locator(AutoMinorLocator())
300     ax.legend(fontsize=14, loc='best')
301     ax.tick_params(labelsize='medium')
302     ax.grid(True, linestyle='-.')
303     ax.set_ylabel("$y-y_{wall}$")
304     if (ax==axes[4] or ax==axes[5] or ax==axes[6] or ax==axes[7]):
305         ax.set_xlabel("$\overline{v'v'})$")
306 fig8.suptitle("$\overline{v'v'})$ at various locations", fontsize=22)
307
308 ## Plot uv
309 fig9, axes = plt.subplots(2,4, constrained_layout=True)
310 axes = axes.reshape(-1)
311 axes[0].plot(uv1_2d[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0], 'b-o', ms=4, label='x = 0 m')
312 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')
313 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')
314 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')
315 axes[4].plot(uv1_2d[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0], 'm+', ms=4, label='x = 1 m')
316 axes[5].plot(uv1_2d[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0], 'y-*', ms=4, label='x = 1.1 m')
317 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')
318 axes[7].plot(uv1_2d[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0], label='x = 1.3 m')
319 for ax in axes:
320     ax.xaxis.set_minor_locator(AutoMinorLocator())
321     ax.yaxis.set_minor_locator(AutoMinorLocator())
322     ax.legend(fontsize=14, loc='best')
323     ax.tick_params(labelsize='medium')
324     ax.grid(True, linestyle='-.')
325     ax.set_ylabel("$y-y_{wall}$")
```

```

326     if (ax==axes[4] or ax==axes[5] or ax==axes[6] or ax==axes[7]):
327         ax.set_xlabel("$\overline{u'}$")
328 fig9.suptitle("$\overline{v'}$ at various locations", fontsize=22)
329
330 # Comparision of both dataset
331 ## Plot uu
332 fig10, axes = plt.subplots(2, 4, constrained_layout=True)
333 axes = axes.reshape(-1)
334 axes[0].plot(uu_2d[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0], 'r-o', ms=4, label='Paper Data [203]')
335 axes[1].plot(uu_2d[i1[1],:],y_2d[i1[1],:]-y_2d[i1[1],0], 'r-o', ms=4, label='Paper Data [203]')
336 axes[2].plot(uu_2d[i1[2],:],y_2d[i1[2],:]-y_2d[i1[2],0], 'r-o', ms=4, label='Paper Data [203]')
337 axes[3].plot(uu_2d[i1[3],:],y_2d[i1[3],:]-y_2d[i1[3],0], 'r-o', ms=4, label='Paper Data [203]')
338 axes[4].plot(uu_2d[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0], 'r-o', ms=4, label='Paper Data [203]')
339 axes[5].plot(uu_2d[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0], 'r-o', ms=4, label='Paper Data [203]')
340 axes[6].plot(uu_2d[i1[6],:],y_2d[i1[6],:]-y_2d[i1[6],0], 'r-o', ms=4, label='Paper Data [203]')
341 axes[7].plot(uu_2d[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0], 'r-o', ms=4, label='Paper Data [203]')
342 axes[0].plot(uu1_2d[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0], 'b-s', ms=4, label='Journal Data [169]')
343 axes[1].plot(uu1_2d[i1[1],:],y_2d[i1[1],:]-y_2d[i1[1],0], 'b-s', ms=4, label='Journal Data [169]')
344 axes[2].plot(uu1_2d[i1[2],:],y_2d[i1[2],:]-y_2d[i1[2],0], 'b-s', ms=4, label='Journal Data [169]')
345 axes[3].plot(uu1_2d[i1[3],:],y_2d[i1[3],:]-y_2d[i1[3],0], 'b-s', ms=4, label='Journal Data [169]')
346 axes[4].plot(uu1_2d[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0], 'b-s', ms=4, label='Journal Data [169]')
347 axes[5].plot(uu1_2d[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0], 'b-s', ms=4, label='Journal Data [169]')
348 axes[6].plot(uu1_2d[i1[6],:],y_2d[i1[6],:]-y_2d[i1[6],0], 'b-s', ms=4, label='Journal Data [169]')
349 axes[7].plot(uu1_2d[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0], 'b-s', ms=4, label='Journal Data [169]')
350 axes[1].plot(x065_off[:,4],x065_off[:,1]-y_2d[i1[1],0], 'k-*', ms=4, label='Experimental Data')
351 axes[2].plot(x080_off[:,4],x080_off[:,1]-y_2d[i1[2],0], 'k-*', ms=4, label='Experimental Data')
352 axes[3].plot(x090_off[:,4],x090_off[:,1]-y_2d[i1[3],0], 'k-*', ms=4, label='Experimental Data')
353 axes[4].plot(x100_off[:,4],x100_off[:,1]-y_2d[i1[4],0], 'k-*', ms=4, label='Experimental Data')
354 axes[5].plot(x110_off[:,4],x110_off[:,1]-y_2d[i1[5],0], 'k-*', ms=4, label='Experimental Data')
355 axes[6].plot(x120_off[:,4],x120_off[:,1]-y_2d[i1[6],0], 'k-*', ms=4, label='Experimental Data')
356 axes[7].plot(x130_off[:,4],x130_off[:,1]-y_2d[i1[7],0], 'k-*', ms=4, label='Experimental Data')
357 for ax,xloc in zip(axes,xx):
358     ax.xaxis.set_minor_locator(AutoMinorLocator())
359     ax.yaxis.set_minor_locator(AutoMinorLocator())
360     ax.legend(title='At X = '+str(xloc)+ ' m', title_fontsize=14, font_size=14, loc='best')
361     ax.tick_params(labelsize='medium')
362     ax.grid(True, linestyle='-.')
363     ax.set_xlim([0,0.25])
364     if (ax==axes[4] or ax==axes[5] or ax==axes[6] or ax==axes[7]):
365         ax.set_xlabel("$\overline{u'}$")
366     if (ax==axes[0] or ax==axes[4]):
367         ax.set_ylabel("$y - \{wall\}$")
368 fig10.suptitle("$\overline{u'}$ at various locations", font_size=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_xlim([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_xlim([0,0.2])
442     if (ax==axes[4] or ax==axes[5] or ax==axes[6] or ax==axes[7]):
443         ax.set_xlabel("$\overline{u'}$ at $x = " + str(xloc) + " m$")
444     if (ax==axes[0] or ax==axes[4]):
445         ax.set_ylabel("y - y_{wall}")
446 fig12.suptitle("$\overline{u'}$ 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]')

```

```

453 axes[2].plot(u_2d[i1[2],:],y_2d[i1[2],:]-y_2d[i1[2],0],'r-o',ms=4,label='Paper Data [203]')
454 axes[3].plot(u_2d[i1[3],:],y_2d[i1[3],:]-y_2d[i1[3],0],'r-o',ms=4,label='Paper Data [203]')
455 axes[4].plot(u_2d[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0],'r-o',ms=4,label='Paper Data [203]')
456 axes[5].plot(u_2d[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0],'r-o',ms=4,label='Paper Data [203]')
457 axes[6].plot(u_2d[i1[6],:],y_2d[i1[6],:]-y_2d[i1[6],0],'r-o',ms=4,label='Paper Data [203]')
458 axes[7].plot(u_2d[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0],'r-o',ms=4,label='Paper Data [203]')
459 axes[0].plot(u1_2d[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0],'b-s',ms=4,label='Journal Data [169]')
460 axes[1].plot(u1_2d[i1[1],:],y_2d[i1[1],:]-y_2d[i1[1],0],'b-s',ms=4,label='Journal Data [169]')
461 axes[2].plot(u1_2d[i1[2],:],y_2d[i1[2],:]-y_2d[i1[2],0],'b-s',ms=4,label='Journal Data [169]')
462 axes[3].plot(u1_2d[i1[3],:],y_2d[i1[3],:]-y_2d[i1[3],0],'b-s',ms=4,label='Journal Data [169]')
463 axes[4].plot(u1_2d[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0],'b-s',ms=4,label='Journal Data [169]')
464 axes[5].plot(u1_2d[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0],'b-s',ms=4,label='Journal Data [169]')
465 axes[6].plot(u1_2d[i1[6],:],y_2d[i1[6],:]-y_2d[i1[6],0],'b-s',ms=4,label='Journal Data [169]')
466 axes[7].plot(u1_2d[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0],'b-s',ms=4,label='Journal Data [169]')
467 axes[1].plot(x065_off[:,2],x065_off[:,1]-y_2d[i1[1],0],'k-*',ms=4,label='Experimental Data')
468 axes[2].plot(x080_off[:,2],x080_off[:,1]-y_2d[i1[2],0],'k-*',ms=4,label='Experimental Data')
469 axes[3].plot(x090_off[:,2],x090_off[:,1]-y_2d[i1[3],0],'k-*',ms=4,label='Experimental Data')
470 axes[4].plot(x100_off[:,2],x100_off[:,1]-y_2d[i1[4],0],'k-*',ms=4,label='Experimental Data')
471 axes[5].plot(x110_off[:,2],x110_off[:,1]-y_2d[i1[5],0],'k-*',ms=4,label='Experimental Data')
472 axes[6].plot(x120_off[:,2],x120_off[:,1]-y_2d[i1[6],0],'k-*',ms=4,label='Experimental Data')
473 axes[7].plot(x130_off[:,2],x130_off[:,1]-y_2d[i1[7],0],'k-*',ms=4,label='Experimental Data')
474 for ax,xloc in zip(axes,xx):
475     ax.xaxis.set_minor_locator(AutoMinorLocator())
476     ax.yaxis.set_minor_locator(AutoMinorLocator())
477     ax.legend(title='At X = '+str(xloc)+ ' m',title_fontsize=14,fontsize=14,loc='best')
478     ax.tick_params(labelsize='medium')
479     ax.grid(True, linestyle='-.')
480     ax.set_ylim([0,0.15])
481     if (ax==axes[4] or ax==axes[5] or ax==axes[6] or ax==axes[7]):
482         ax.set_xlabel("$\overline{u}$")
483     if (ax==axes[0] or ax==axes[4]):
484         ax.set_ylabel("$y - y_{wall}$")
485 fig13.suptitle("$\overline{u}$ at various locations", fontsize=22)
486
487 ## Plot v
488 fig14,axes = plt.subplots(2,4,constrained_layout=True)
489 axes = axes.reshape(-1)
490 axes[0].plot(v_2d[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0],'r-o',ms=4,label='Paper Data [203]')
491 axes[1].plot(v_2d[i1[1],:],y_2d[i1[1],:]-y_2d[i1[1],0],'r-o',ms=4,label='Paper Data [203]')
492 axes[2].plot(v_2d[i1[2],:],y_2d[i1[2],:]-y_2d[i1[2],0],'r-o',ms=4,label='Paper Data [203]')
493 axes[3].plot(v_2d[i1[3],:],y_2d[i1[3],:]-y_2d[i1[3],0],'r-o',ms=4,label='Paper Data [203]')

```

```

494 axes[4].plot(v_2d[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0],'r-o',ms=4,label='Paper Data [203]')
495 axes[5].plot(v_2d[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0],'r-o',ms=4,label='Paper Data [203]')
496 axes[6].plot(v_2d[i1[6],:],y_2d[i1[6],:]-y_2d[i1[6],0],'r-o',ms=4,label='Paper Data [203]')
497 axes[7].plot(v_2d[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0],'r-o',ms=4,label='Paper Data [203]')
498 axes[0].plot(v1_2d[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0],'b-s',ms=4,label='Journal Data [169]')
499 axes[1].plot(v1_2d[i1[1],:],y_2d[i1[1],:]-y_2d[i1[1],0],'b-s',ms=4,label='Journal Data [169]')
500 axes[2].plot(v1_2d[i1[2],:],y_2d[i1[2],:]-y_2d[i1[2],0],'b-s',ms=4,label='Journal Data [169]')
501 axes[3].plot(v1_2d[i1[3],:],y_2d[i1[3],:]-y_2d[i1[3],0],'b-s',ms=4,label='Journal Data [169]')
502 axes[4].plot(v1_2d[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0],'b-s',ms=4,label='Journal Data [169]')
503 axes[5].plot(v1_2d[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0],'b-s',ms=4,label='Journal Data [169]')
504 axes[6].plot(v1_2d[i1[6],:],y_2d[i1[6],:]-y_2d[i1[6],0],'b-s',ms=4,label='Journal Data [169]')
505 axes[7].plot(v1_2d[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0],'b-s',ms=4,label='Journal Data [169]')
506 axes[1].plot(x065_off[:,3],x065_off[:,1]-y_2d[i1[1],0],'k-*',ms=4,label='Experimental Data')
507 axes[2].plot(x080_off[:,3],x080_off[:,1]-y_2d[i1[2],0],'k-*',ms=4,label='Experimental Data')
508 axes[3].plot(x090_off[:,3],x090_off[:,1]-y_2d[i1[3],0],'k-*',ms=4,label='Experimental Data')
509 axes[4].plot(x100_off[:,3],x100_off[:,1]-y_2d[i1[4],0],'k-*',ms=4,label='Experimental Data')
510 axes[5].plot(x110_off[:,3],x110_off[:,1]-y_2d[i1[5],0],'k-*',ms=4,label='Experimental Data')
511 axes[6].plot(x120_off[:,3],x120_off[:,1]-y_2d[i1[6],0],'k-*',ms=4,label='Experimental Data')
512 axes[7].plot(x130_off[:,3],x130_off[:,1]-y_2d[i1[7],0],'k-*',ms=4,label='Experimental Data')
513 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_xlim([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 - $\overline{y}$")
514 fig14.suptitle("$\overline{v}$ at various locations", fontsize=22)
515
516 # %% Assignment T2 - Modeled Turbulent Shear Stress
517 visturb_2d = vis_2d - viscous
518 visturb1_2d = vis1_2d - viscous
519 uv_model_2d_calc = -visturb_2d * (dudy+dvdx)
520
521 # Finding boundary layer thickness at the measuring points (Paper Data)
522 tbl065 = np.max(np.where(((vis_2d[i1[1],:]/viscos-1)<=1) == 0))
523 tbl080 = np.max(np.where(((vis_2d[i1[2],:]/viscos-1)<=1) == 0))
524 tbl090 = np.max(np.where(((vis_2d[i1[3],:]/viscos-1)<=1) == 0))
525 tbl100 = np.max(np.where(((vis_2d[i1[4],:]/viscos-1)<=1) == 0))
526 tbl110 = np.max(np.where(((vis_2d[i1[5],:]/viscos-1)<=1) == 0))
527 tbl120 = np.max(np.where(((vis_2d[i1[6],:]/viscos-1)<=1) == 0))
528 tbl130 = np.max(np.where(((vis_2d[i1[7],:]/viscos-1)<=1) == 0))
529
530 blt060 = y_2d[i1[1],tbl065]

```

```

541 blt080 = y_2d[i1[2],tbl080]
542 blt090 = y_2d[i1[3],tbl090]
543 blt100 = y_2d[i1[4],tbl100]
544 blt110 = y_2d[i1[5],tbl110]
545 blt120 = y_2d[i1[6],tbl120]
546 blt130 = y_2d[i1[7],tbl130]
547
548 # Finding boundary layer thickness at the measuring points (Journal Data)
549 tbl1065 = np.max(np.where(((vis1_2d[i1[1],:]/viscos-1)<=1) == 0))
550 tbl1080 = np.max(np.where(((vis1_2d[i1[2],:]/viscos-1)<=1) == 0))
551 tbl1090 = np.max(np.where(((vis1_2d[i1[3],:]/viscos-1)<=1) == 0))
552 tbl1100 = np.max(np.where(((vis1_2d[i1[4],:]/viscos-1)<=1) == 0))
553 tbl1110 = np.max(np.where(((vis1_2d[i1[5],:]/viscos-1)<=1) == 0))
554 tbl1120 = np.max(np.where(((vis1_2d[i1[6],:]/viscos-1)<=1) == 0))
555 tbl1130 = np.max(np.where(((vis1_2d[i1[7],:]/viscos-1)<=1) == 0))
556
557 blt1060 = y_2d[i1[1],tbl1065]
558 blt1080 = y_2d[i1[2],tbl1080]
559 blt1090 = y_2d[i1[3],tbl1090]
560 blt1100 = y_2d[i1[4],tbl1100]
561 blt1110 = y_2d[i1[5],tbl1110]
562 blt1120 = y_2d[i1[6],tbl1120]
563 blt1130 = y_2d[i1[7],tbl1130]
564
565 fig17, axes = plt.subplots(2,4,constrained_layout=True)
566 axes = axes.reshape(-1)
567 axes[0].plot(-uv_model_2d[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0],'b-o',ms=4,label='Modeled')
568 axes[1].plot(-uv_model_2d[i1[1],:],y_2d[i1[1],:]-y_2d[i1[1],0],'b-o',ms=4,label='Modeled')
569 axes[2].plot(-uv_model_2d[i1[2],:],y_2d[i1[2],:]-y_2d[i1[2],0],'b-o',ms=4,label='Modeled')
570 axes[3].plot(-uv_model_2d[i1[3],:],y_2d[i1[3],:]-y_2d[i1[3],0],'b-o',ms=4,label='Modeled')
571 axes[4].plot(-uv_model_2d[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0],'b-o',ms=4,label='Modeled')
572 axes[5].plot(-uv_model_2d[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0],'b-o',ms=4,label='Modeled')
573 axes[6].plot(-uv_model_2d[i1[6],:],y_2d[i1[6],:]-y_2d[i1[6],0],'b-o',ms=4,label='Modeled')
574 axes[7].plot(-uv_model_2d[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0],'b-o',ms=4,label='Modeled')
575 axes[0].plot(-uv_2d[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0],'r-o',ms=4,label='Resolved')
576 axes[1].plot(-uv_2d[i1[1],:],y_2d[i1[1],:]-y_2d[i1[1],0],'r-o',ms=4,label='Resolved')
577 axes[2].plot(-uv_2d[i1[2],:],y_2d[i1[2],:]-y_2d[i1[2],0],'r-o',ms=4,label='Resolved')
578 axes[3].plot(-uv_2d[i1[3],:],y_2d[i1[3],:]-y_2d[i1[3],0],'r-o',ms=4,label='Resolved')
579 axes[4].plot(-uv_2d[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0],'r-o',ms=4,label='Resolved')
580 axes[5].plot(-uv_2d[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0],'r-o',ms=4,label='Resolved')
581 axes[6].plot(-uv_2d[i1[6],:],y_2d[i1[6],:]-y_2d[i1[6],0],'r-o',ms=4,label='Resolved')
582 axes[7].plot(-uv_2d[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0],'r-o',ms=4,label='Resolved')
583 axes[1].plot(-x065_off[:,6],x065_off[:,1]-y_2d[i1[1],0],'k-*',ms=4,label='Experimental')
584 axes[2].plot(-x080_off[:,6],x080_off[:,1]-y_2d[i1[2],0],'k-*',ms=4,label='Experimental')
585 axes[3].plot(-x090_off[:,6],x090_off[:,1]-y_2d[i1[3],0],'k-*',ms=4,label='Experimental')
586 axes[4].plot(-x100_off[:,6],x100_off[:,1]-y_2d[i1[4],0],'k-*',ms=4,label='Experimental')
587 axes[5].plot(-x110_off[:,6],x110_off[:,1]-y_2d[i1[5],0],'k-*',ms=4,label='Experimental')
588 axes[6].plot(-x120_off[:,6],x120_off[:,1]-y_2d[i1[6],0],'k-*',ms=4,label='Experimental')
589 axes[7].plot(-x130_off[:,6],x130_off[:,1]-y_2d[i1[7],0],'k-*',ms=4,label='Experimental')
590 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')
591 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')
592 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')
593 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')
594 axes[5].plot(np.linspace(np.min(-uv_2d[i1[5],:]),np.max(-uv_2d[i1[5],:]),10),(y_2d[i1

```

```

595 [5],tbl110]-y_2d[i1[5],0])*np.ones(10,'k--',ms=4,label='BL thickness')
596 axes[6].plot(np.linspace(np.min(-uv_2d[i1[6],:]),np.max(-uv_2d[i1[6],:]),10),(y_2d[i1
597 [6],tbl120]-y_2d[i1[6],0])*np.ones(10,'k--',ms=4,label='BL thickness')
598 axes[7].plot(np.linspace(np.min(-uv_2d[i1[7],:]),np.max(-uv_2d[i1[7],:]),10),(y_2d[i1
599 [7],tbl130]-y_2d[i1[7],0])*np.ones(10,'k--',ms=4,label='BL thickness')
600 for ax,xloc in zip(axes,xx):
601     ax.xaxis.set_minor_locator(AutoMinorLocator())
602     ax.yaxis.set_minor_locator(AutoMinorLocator())
603     ax.legend(title='At X = '+str(xloc)+ ' m',title_fontsize=14,fontsize=14,loc='best')
604     ax.tick_params(labelsize='medium')
605     ax.grid(True, linestyle='-.')
606     ax.set_ylim([0,0.25])
607     if (ax==axes[0] or ax==axes[4]):
608         ax.set_ylabel("$y-y_{wall}$")
609     if (ax==axes[4] or ax==axes[5] or ax==axes[6] or ax==axes[7]):
610         ax.set_xlabel("$-<\overline{u'v'}>$")
611 fig17.suptitle("Turbulent Shear Stress -$<\overline{u'v'}>$ (Reference [203])",fontsize
612 =22)
613
614 fig17a,axes = plt.subplots(1,1,constrained_layout=True)
615 a1 = axes.contourf(x_2d,y_2d,-uv_model_2d,cmap='jet',levels=50)
616 cb1 = fig17a.colorbar(a1,ax=axes)
617 cb1.ax.tick_params(labelsize=22)
618 a1.ax.tick_params(labelsize=16)
619 plt.axis([0.6,1.5,0,1])
620 axes.set_xlabel('X (m)',fontsize=18)
621 axes.set_ylabel('Y (m)',fontsize=18)
622 axes.set_title("Modeled Turbulent Shear Stress -$<\overline{u'v'}>$ (Reference [203])",
623 fontsize=24)
624
625 fig17b,axes = plt.subplots(1,1,constrained_layout=True)
626 a1 = axes.contourf(x_2d,y_2d,-uv_2d,cmap='jet',levels=50)
627 cb1 = fig17b.colorbar(a1,ax=axes)
628 cb1.ax.tick_params(labelsize=22)
629 a1.ax.tick_params(labelsize=16)
630 plt.axis([0.6,1.5,0,1])
631 axes.set_xlabel('X (m)',fontsize=18)
632 axes.set_ylabel('Y (m)',fontsize=18)
633 axes.set_title("Resolved Turbulent Shear Stress -$<\overline{u'v'}>$ (Reference [203])",
634 fontsize=24)
635
636 fig17c,axes = plt.subplots(2,4,constrained_layout=True)
637 axes = axes.reshape(-1)
638 axes[0].plot(-uv1_model_2d[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0],'b-o',ms=4,label=
639 'Modeled')
640 axes[1].plot(-uv1_model_2d[i1[1],:],y_2d[i1[1],:]-y_2d[i1[1],0],'b-o',ms=4,label=
641 'Modeled')
642 axes[2].plot(-uv1_model_2d[i1[2],:],y_2d[i1[2],:]-y_2d[i1[2],0],'b-o',ms=4,label=
643 'Modeled')
644 axes[3].plot(-uv1_model_2d[i1[3],:],y_2d[i1[3],:]-y_2d[i1[3],0],'b-o',ms=4,label=
645 'Modeled')
646 axes[4].plot(-uv1_model_2d[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0],'b-o',ms=4,label=
647 'Modeled')
648 axes[5].plot(-uv1_model_2d[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0],'b-o',ms=4,label=
649 'Modeled')
650 axes[6].plot(-uv1_model_2d[i1[6],:],y_2d[i1[6],:]-y_2d[i1[6],0],'b-o',ms=4,label=
651 'Modeled')
652 axes[7].plot(-uv1_model_2d[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0],'b-o',ms=4,label=
653 'Modeled')
654 axes[0].plot(-uv1_2d[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0],'r-o',ms=4,label='Resolved')
655 axes[1].plot(-uv1_2d[i1[1],:],y_2d[i1[1],:]-y_2d[i1[1],0],'r-o',ms=4,label='Resolved')
656 axes[2].plot(-uv1_2d[i1[2],:],y_2d[i1[2],:]-y_2d[i1[2],0],'r-o',ms=4,label='Resolved')
657 axes[3].plot(-uv1_2d[i1[3],:],y_2d[i1[3],:]-y_2d[i1[3],0],'r-o',ms=4,label='Resolved')
658 axes[4].plot(-uv1_2d[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0],'r-o',ms=4,label='Resolved')
659 axes[5].plot(-uv1_2d[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0],'r-o',ms=4,label='Resolved')
660 axes[6].plot(-uv1_2d[i1[6],:],y_2d[i1[6],:]-y_2d[i1[6],0],'r-o',ms=4,label='Resolved')

```

```

647 axes[7].plot(-uv1_2d[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0],'r-o',ms=4,label='Resolved')
648 axes[1].plot(-x065_off[:,6],x065_off[:,1]-y_2d[i1[1],0],'k-*',ms=4,label='Experimental')
649 axes[2].plot(-x080_off[:,6],x080_off[:,1]-y_2d[i1[2],0],'k-*',ms=4,label='Experimental')
650 axes[3].plot(-x090_off[:,6],x090_off[:,1]-y_2d[i1[3],0],'k-*',ms=4,label='Experimental')
651 axes[4].plot(-x100_off[:,6],x100_off[:,1]-y_2d[i1[4],0],'k-*',ms=4,label='Experimental')
652 axes[5].plot(-x110_off[:,6],x110_off[:,1]-y_2d[i1[5],0],'k-*',ms=4,label='Experimental')
653 axes[6].plot(-x120_off[:,6],x120_off[:,1]-y_2d[i1[6],0],'k-*',ms=4,label='Experimental')
654 axes[7].plot(-x130_off[:,6],x130_off[:,1]-y_2d[i1[7],0],'k-*',ms=4,label='Experimental')
655 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')
656 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')
657 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')
658 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')
659 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')
660 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')
661 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')
662 for ax,xloc in zip(axes,xx):
663     ax.xaxis.set_minor_locator(AutoMinorLocator())
664     ax.yaxis.set_minor_locator(AutoMinorLocator())
665     ax.legend(title='At X = '+str(xloc)+' m',title_fontsize=14,fontsize=14,loc='best')
666     ax.tick_params(labelsize='medium')
667     ax.grid(True, linestyle='-.')
668     ax.set_ylim([0,0.25])
669     if (ax==axes[0] or ax==axes[4]):
670         ax.set_ylabel("$y-y_{wall}$")
671     if (ax==axes[4] or ax==axes[5] or ax==axes[6] or ax==axes[7]):
672         ax.set_xlabel("$-\overline{u'v'}$")
673 fig17c.suptitle("Turbulent Shear Stress -$-\overline{u'v'}$ (Reference [169])",fontsize
   =22)
674
675 fig17d,axes = plt.subplots(1,1,constrained_layout=True)
676 a1 = axes.contourf(x_2d,y_2d,-uv1_model_2d,cmap='jet',levels=50)
677 cb1 = fig17d.colorbar(a1,ax=axes)
678 cb1.ax.tick_params(labelsize=22)
679 a1.ax.tick_params(labelsize=16)
680 plt.axis([0.6,1.5,0,1])
681 axes.set_xlabel('X (m)',fontsize=18)
682 axes.set_ylabel('Y (m)',fontsize=18)
683 axes.set_title("Modeled Turbulent Shear Stress -$-\overline{u'v'}$ (Reference [169])",
   fontsize=24)
684
685 fig17e,axes = plt.subplots(1,1,constrained_layout=True)
686 a1 = axes.contourf(x_2d,y_2d,-uv1_2d,cmap='jet',levels=50)
687 cb1 = fig17e.colorbar(a1,ax=axes)
688 cb1.ax.tick_params(labelsize=22)
689 a1.ax.tick_params(labelsize=16)
690 plt.axis([0.6,1.5,0,1])
691 axes.set_xlabel('X (m)',fontsize=18)
692 axes.set_ylabel('Y (m)',fontsize=18)
693 axes.set_title("Resolved Turbulent Shear Stress -$-\overline{u'v'}$ (Reference [169])",
   fontsize=24)
694
695 # %% Assignment T3 - The Turbulent Viscosity
696 visc_ratio = visturb_2d/vicos
697 visc1_ratio = visturb1_2d/vicos
698
699 # Visualize Turbulent viscosity ratio
700 vmin1 = 1
701 vmax1 = np.max(np.max(visc_ratio))
702 v1 = np.linspace(vmin1,vmax1,100)

```

```

703 vmin2 = 1
704 vmax2 = np.max(np.max(visc1_ratio))
705 v2 = np.linspace(vmin2,vmax2,100)
706 fig15,axes = plt.subplots(2,1,constrained_layout=True)
707 a1 = axes[0].contourf(x_2d,y_2d,visc_ratio,cmap='gist_gray',levels=v1,extend='min')
708 a2 = axes[1].contourf(x_2d,y_2d,visc1_ratio,cmap='gist_gray',levels=v2,extend='min')
709 a1.cmap.set_under('black')
710 a2.cmap.set_under('black')
711 cb1 = fig15.colorbar(a1,ax=axes[0])
712 cb2 = fig15.colorbar(a2,ax=axes[1])
713 cb1.ax.tick_params(labelsize=16)
714 cb2.ax.tick_params(labelsize=16)
715 a1.ax.tick_params(labelsize=16)
716 a2.ax.tick_params(labelsize=16)
717 axes[0].set_title(r'Turbulent Viscosity Ratio - $\nu_{t}/\nu$ (Paper Data [203])',
    fontsize=22)
718 axes[1].set_title(r'Turbulent Viscosity Ratio - $\nu_{t}/\nu$ (Journal Data [169])',
    fontsize=22)
719 for ax in axes:
720     ax.axis([0.6,1.5,0,1])
721     ax.set_xlabel('X (m)',fontsize=18)
722     ax.set_ylabel('Y (m)',fontsize=18)
723
724 fig19,axes = plt.subplots(2,4,constrained_layout=True)
725 axes = axes.reshape(-1)
726 axes[0].plot(visc_ratio[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0],'b-o',ms=4,label='Paper
    Data [203]')
727 axes[1].plot(visc_ratio[i1[1],:],y_2d[i1[1],:]-y_2d[i1[1],0],'b-o',ms=4,label='Paper
    Data [203]')
728 axes[2].plot(visc_ratio[i1[2],:],y_2d[i1[2],:]-y_2d[i1[2],0],'b-o',ms=4,label='Paper
    Data [203]')
729 axes[3].plot(visc_ratio[i1[3],:],y_2d[i1[3],:]-y_2d[i1[3],0],'b-o',ms=4,label='Paper
    Data [203]')
730 axes[4].plot(visc_ratio[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0],'b-o',ms=4,label='Paper
    Data [203]')
731 axes[5].plot(visc_ratio[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0],'b-o',ms=4,label='Paper
    Data [203]')
732 axes[6].plot(visc_ratio[i1[6],:],y_2d[i1[6],:]-y_2d[i1[6],0],'b-o',ms=4,label='Paper
    Data [203]')
733 axes[7].plot(visc_ratio[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0],'b-o',ms=4,label='Paper
    Data [203]')
734 axes[0].plot(visc1_ratio[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0],'r-o',ms=4,label='Journal
    Data [169]')
735 axes[1].plot(visc1_ratio[i1[1],:],y_2d[i1[1],:]-y_2d[i1[1],0],'r-o',ms=4,label='Journal
    Data [169]')
736 axes[2].plot(visc1_ratio[i1[2],:],y_2d[i1[2],:]-y_2d[i1[2],0],'r-o',ms=4,label='Journal
    Data [169]')
737 axes[3].plot(visc1_ratio[i1[3],:],y_2d[i1[3],:]-y_2d[i1[3],0],'r-o',ms=4,label='Journal
    Data [169]')
738 axes[4].plot(visc1_ratio[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0],'r-o',ms=4,label='Journal
    Data [169]')
739 axes[5].plot(visc1_ratio[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0],'r-o',ms=4,label='Journal
    Data [169]')
740 axes[6].plot(visc1_ratio[i1[6],:],y_2d[i1[6],:]-y_2d[i1[6],0],'r-o',ms=4,label='Journal
    Data [169]')
741 axes[7].plot(visc1_ratio[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0],'r-o',ms=4,label='Journal
    Data [169]')
742 for ax,xloc in zip(axes,xx):
743     ax.xaxis.set_minor_locator(AutoMinorLocator())
744     ax.yaxis.set_minor_locator(AutoMinorLocator())
745     ax.legend(title='At X = '+str(xloc)+ ' m',title_fontsize=14,fontsize=14,loc='best')
746     ax.tick_params(labelsize='medium')
747     ax.grid(True, linestyle='-.')
748     if (ax==axes[0] or ax==axes[4]):
749         ax.set_ylabel("$y-y_{wall}$")
750     if (ax==axes[4] or ax==axes[5] or ax==axes[6] or ax==axes[7]):
```

```

751         ax.set_xlabel(r"\nu_{t}/\nu")
752 fig19.suptitle(r'Turbulent Viscosity Ratio - $\nu_{t}/\nu$', fontsize=22)
753
754 # term 1,5 (i=1 and j=1)
755 # term 2,6 (i=1 and j=2)
756 # term 3,7 (i=2 and j=1)
757 # term 4,8 (i=2 and j=2)
758 term1,dummy1 = dphidx_dy(x_2d_new,y_2d_new,visturb_2d*dudx)
759 dummy2,term2 = dphidx_dy(x_2d_new,y_2d_new,visturb_2d*dudy)
760 term3,dummy3 = dphidx_dy(x_2d_new,y_2d_new,visturb_2d*dvdx)
761 dummy4,term4 = dphidx_dy(x_2d_new,y_2d_new,visturb_2d*dvdy)
762
763 term5,dummy5 = dphidx_dy(x_2d_new,y_2d_new,uu_2d)
764 dummy6,term6 = dphidx_dy(x_2d_new,y_2d_new,uv_2d)
765 term7,dummy7 = dphidx_dy(x_2d_new,y_2d_new,uv_2d)
766 dummy8,term8 = dphidx_dy(x_2d_new,y_2d_new,vv_2d)
767
768 div1_i1 = term1 + term2
769 div1_i2 = term3 + term4
770 div2_i1 = -term5 - term6
771 div2_i2 = -term7 - term8
772
773 fig20,axes = plt.subplots(2,4,constrained_layout=True)
774 axes = axes.reshape(-1)
775 axes[0].plot(div1_i1[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0],'b-o',ms=4,label=r'Modeled')
776 axes[1].plot(div1_i1[i1[1],:],y_2d[i1[1],:]-y_2d[i1[1],0],'b-o',ms=4,label=r'Modeled')
777 axes[2].plot(div1_i1[i1[2],:],y_2d[i1[2],:]-y_2d[i1[2],0],'b-o',ms=4,label=r'Modeled')
778 axes[3].plot(div1_i1[i1[3],:],y_2d[i1[3],:]-y_2d[i1[3],0],'b-o',ms=4,label=r'Modeled')
779 axes[4].plot(div1_i1[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0],'b-o',ms=4,label=r'Modeled')
780 axes[5].plot(div1_i1[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0],'b-o',ms=4,label=r'Modeled')
781 axes[6].plot(div1_i1[i1[6],:],y_2d[i1[6],:]-y_2d[i1[6],0],'b-o',ms=4,label=r'Modeled')
782 axes[7].plot(div1_i1[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0],'b-o',ms=4,label=r'Modeled')
783 axes[0].plot(div2_i1[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0],'r-o',ms=4,label=r'Resolved')
784 axes[1].plot(div2_i1[i1[1],:],y_2d[i1[1],:]-y_2d[i1[1],0],'r-o',ms=4,label=r'Resolved')
785 axes[2].plot(div2_i1[i1[2],:],y_2d[i1[2],:]-y_2d[i1[2],0],'r-o',ms=4,label=r'Resolved')
786 axes[3].plot(div2_i1[i1[3],:],y_2d[i1[3],:]-y_2d[i1[3],0],'r-o',ms=4,label=r'Resolved')
787 axes[4].plot(div2_i1[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0],'r-o',ms=4,label=r'Resolved')
788 axes[5].plot(div2_i1[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0],'r-o',ms=4,label=r'Resolved')
789 axes[6].plot(div2_i1[i1[6],:],y_2d[i1[6],:]-y_2d[i1[6],0],'r-o',ms=4,label=r'Resolved')
790 axes[7].plot(div2_i1[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0],'r-o',ms=4,label=r'Resolved')
791 for ax,xloc in zip(axes,xx):
792     ax.xaxis.set_minor_locator(AutoMinorLocator())
793     ax.yaxis.set_minor_locator(AutoMinorLocator())
794     ax.legend(title='At X = '+str(xloc)+ ' m',title_fontsize=14,fontsize=14,loc='best')
795     ax.tick_params(labelsize='medium')
796     ax.grid(True, linestyle='-.')
797     ax.set_ylim([0,0.2])
798     if (ax==axes[0] or ax==axes[4]):
799         ax.set_ylabel("$y-y_{wall}$")
800 fig20.suptitle(r'Modeled ($\frac{\partial}{\partial x_j}(\langle \nu_{t} \rangle \frac{\partial}{\partial v_i})$) and Resolved ($-\frac{\partial}{\partial x_j}(\langle v_i v_j \rangle)$) Shear Stress", fontsize=22)
801
802 fig21,axes = plt.subplots(2,4,constrained_layout=True)
803 axes = axes.reshape(-1)
804 axes[0].plot(div1_i2[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0],'b-o',ms=4,label=r'Modeled')
805 axes[1].plot(div1_i2[i1[1],:],y_2d[i1[1],:]-y_2d[i1[1],0],'b-o',ms=4,label=r'Modeled')
806 axes[2].plot(div1_i2[i1[2],:],y_2d[i1[2],:]-y_2d[i1[2],0],'b-o',ms=4,label=r'Modeled')
807 axes[3].plot(div1_i2[i1[3],:],y_2d[i1[3],:]-y_2d[i1[3],0],'b-o',ms=4,label=r'Modeled')
808 axes[4].plot(div1_i2[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0],'b-o',ms=4,label=r'Modeled')
809 axes[5].plot(div1_i2[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0],'b-o',ms=4,label=r'Modeled')
810 axes[6].plot(div1_i2[i1[6],:],y_2d[i1[6],:]-y_2d[i1[6],0],'b-o',ms=4,label=r'Modeled')
811 axes[7].plot(div1_i2[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0],'b-o',ms=4,label=r'Modeled')
812 axes[0].plot(div2_i2[i1[0],:],y_2d[i1[0],:]-y_2d[i1[0],0],'r-o',ms=4,label=r'Resolved')
813 axes[1].plot(div2_i2[i1[1],:],y_2d[i1[1],:]-y_2d[i1[1],0],'r-o',ms=4,label=r'Resolved')
814 axes[2].plot(div2_i2[i1[2],:],y_2d[i1[2],:]-y_2d[i1[2],0],'r-o',ms=4,label=r'Resolved')

```

```

815 axes[3].plot(div2_i2[i1[3],:],y_2d[i1[3],:]-y_2d[i1[3],0],'r-o',ms=4,label=r"Resolved")
816 axes[4].plot(div2_i2[i1[4],:],y_2d[i1[4],:]-y_2d[i1[4],0],'r-o',ms=4,label=r"Resolved")
817 axes[5].plot(div2_i2[i1[5],:],y_2d[i1[5],:]-y_2d[i1[5],0],'r-o',ms=4,label=r"Resolved")
818 axes[6].plot(div2_i2[i1[6],:],y_2d[i1[6],:]-y_2d[i1[6],0],'r-o',ms=4,label=r"Resolved")
819 axes[7].plot(div2_i2[i1[7],:],y_2d[i1[7],:]-y_2d[i1[7],0],'r-o',ms=4,label=r"Resolved")
820 for ax,xloc in zip(axes,xx):
821     ax.xaxis.set_minor_locator(AutoMinorLocator())
822     ax.yaxis.set_minor_locator(AutoMinorLocator())
823     ax.legend(title='At X = '+str(xloc)+ ' m',title_fontsize=14,fontsize=14,loc='best')
824     ax.tick_params(labelsize='medium')
825     ax.grid(True, linestyle='-.')
826     ax.set_ylim([0,0.2])
827     if (ax==axes[0] or ax==axes[4]):
828         ax.set_ylabel("$y-y_{wall}$")
829 fig21.suptitle(r"Modeled ($\frac{\partial}{\partial x_j}(\overline{u}_t) \frac{\partial}{\partial x_j v_{2j})$) and Resolved ($-\frac{\partial}{\partial x_j}(\overline{v}_{2j} v_{jj}) \frac{\partial}{\partial x_j u_t)$) Shear Stress", fontsize=22)
830
831 # %% For visualization - location of experimental points
832 fig22 = plt.figure()
833 axes = fig22.add_axes([0.1,0.1,0.8,0.8])
834 axes.plot(x_2d[:,0],y_2d[:,0],lw=4,color='black')
835 axes.plot(x_2d[0,:],y_2d[0,:],lw=4,color='black')
836 axes.plot(x_2d[:, -1],y_2d[:, -1],lw=4,color='black')
837 axes.plot(x_2d[-1,:],y_2d[-1,:],lw=4,color='black')
838 axes.plot(x065_off[:,0],np.linspace(x065_off[-1,1],0.6,x065_off.shape[0]),'b--',label='X = 0.65 m')
839 axes.plot(x080_off[:,0],np.linspace(x080_off[-1,1],0.6,x080_off.shape[0]),'g--',label='X = 0.80 m')
840 axes.plot(x090_off[:,0],np.linspace(x090_off[-1,1],0.6,x090_off.shape[0]),'r--',label='X = 0.90 m')
841 axes.plot(x100_off[:,0],np.linspace(x100_off[-1,1],0.6,x100_off.shape[0]),'c--',label='X = 0.10 m')
842 axes.plot(x110_off[:,0],np.linspace(x110_off[-1,1],0.6,x110_off.shape[0]),'m--',label='X = 0.11 m')
843 axes.plot(x120_off[:,0],np.linspace(x120_off[-1,1],0.6,x120_off.shape[0]),'y--',label='X = 0.12 m')
844 axes.plot(x130_off[:,0],np.linspace(x130_off[-1,1],0.6,x130_off.shape[0]),'k--',label='X = 0.13 m')
845 axes.legend(fontsize=14,loc='best')
846 axes.tick_params(axis="x", labelsize=15)
847 axes.tick_params(axis="y", labelsize=15)
848 axes.set_xlabel('x [m]',fontsize=20)
849 axes.set_ylabel('y [m]',fontsize=20)
850 axes.set_title('Location of Points for Experimental Data',fontsize=24)

```

For Assignment 2b : Section 2.4 to 2.5

```
1 import scipy.io as sio
2 import numpy as np
3 import matplotlib.pyplot as plt
4 from mpl_toolkits.axes_grid1.inset_locator import inset_axes
5 from dphidx_dy import dphidx_dy
6 from IPython import display
7 import numpy.ma as ma
8 from matplotlib.ticker import (MultipleLocator, FormatStrFormatter,
9                                AutoMinorLocator)
10 plt.rcParams.update({'font.size': 22})
11
12 # makes sure figures are updated when using ipython
13 display.clear_output()
14
15
16 re = 9.36e+5
17 viscos = 1/re
18
19 xy_hump_fine = np.loadtxt("xy_hump_fine.dat")
20 x1=xy_hump_fine[:,0]
21 y1=xy_hump_fine[:,1]
22
23
24 nim1=int(x1[0])
25 njm1=int(y1[0])
26
27 ni=nim1+1
28 nj=njm1+1
29
30
31 x=x1[1:]
32 y=y1[1:]
33
34 x_2d=np.reshape(x,(njm1,nim1))
35 y_2d=np.reshape(y,(njm1,nim1))
36
37 x_2d=np.transpose(x_2d)
38 y_2d=np.transpose(y_2d)
39
40 # compute cell centers
41 xp2d= np.zeros((ni,nj))
42 yp2d= np.zeros((ni,nj))
43
44 for jj in range (0,nj):
45     for ii in range (0,ni):
46
47         im1=max(ii-1,0)
48         jm1=max(jj-1,0)
49
50         i=min(ii,nim1-1)
51         j=min(jj,njm1-1)
52
53
54         xp2d[ii,jj]=0.25*(x_2d[i,j]+x_2d[im1,j]+x_2d[i,jm1]+x_2d[im1,jm1])
55         yp2d[ii,jj]=0.25*(y_2d[i,j]+y_2d[im1,j]+y_2d[i,jm1]+y_2d[im1,jm1])
56
57 # read data file
58 vectz = np.loadtxt("vectz_fine.dat")
59 ntstep=vectz[0]
60 ni=int(vectz[1])
61 nj=int(vectz[2])
62 nk=int(vectz[3])
63 n=len(vectz)
```

```
65 #           write(48,*)uvec(i,j)
66 #           write(48,*)vvec(i,j)
67 #           write(48,*)fk2d(i,j)
68 #           write(48,*)uvec2(i,j)
69 #           write(48,*)vvec2(i,j)
70 #           write(48,*)wvec2(i,j)
71 #           write(48,*)uvvec(i,j)
72 #           write(48,*)p2d(i,j)
73 #           write(48,*)rk2d(i,j)
74 #           write(48,*)vis2d(i,j)
75 #           write(48,*)dissp2d(i,j)
76 #           write(48,*)wvec(i,j)
77 #           write(48,*)vtvec(i,j)
78 #           write(48,*)tvec(i,j)

79
80
81 nn=14
82 nst=3
83 iu=range(nst+1,n,nn)
84 iv=range(nst+2,n,nn)
85 ifk=range(nst+3,n,nn)
86 iuu=range(nst+4,n,nn)
87 ivv=range(nst+5,n,nn)
88 iww=range(nst+6,n,nn)
89 iuv=range(nst+7,n,nn)
90 ip=range(nst+8,n,nn)
91 ik=range(nst+9,n,nn)
92 ivis=range(nst+10,n,nn)
93 idiss=range(nst+11,n,nn)

94
95 u=vectz[iu]/ntstep
96 v=vectz[iv]/ntstep
97 fk=vectz[ifk]/ntstep
98 uu=vectz[iuu]/ntstep
99 vv=vectz[ivv]/ntstep
100 ww=vectz[iww]/ntstep
101 uv=vectz[iuv]/ntstep
102 p=vectz[ip]/ntstep
103 k_model=vectz[ik]/ntstep
104 vis=vectz[ivis]/ntstep
105 diss=vectz[idiss]/ntstep

106
107 # uu is total inst. velocity squared. Hence the resolved turbulent resolved stresses are
108 # obtained as
109 uu=uu-u**2
110 vv=vv-v**2
111 uv=uv-u*v

112 p_2d=np.reshape(p,(ni,nj))
113 u_2d=np.reshape(u,(ni,nj))
114 v_2d=np.reshape(v,(ni,nj))
115 fk_2d=np.reshape(fk,(ni,nj))
116 uu_2d=np.reshape(uu,(ni,nj))
117 uv_2d=np.reshape(uv,(ni,nj))
118 vv_2d=np.reshape(vv,(ni,nj))
119 ww_2d=np.reshape(ww,(ni,nj))
120 k_model_2d=np.reshape(k_model,(ni,nj))
121 vis_2d=np.reshape(vis,(ni,nj)) #this is to total viscosity, i.e. vis_tot=vis+vis_turb
122 diss_2d=np.reshape(diss,(ni,nj)) #this is to total viscosity, i.e. vis_tot=vis+vis_turb

123
124 # set fk_2d=1 at upper boundary
125 fk_2d[:,nj-1]=fk_2d[:,nj-2]

126
127 # Plot Mesh for Visualization
128 a = np.transpose(xp2d)
129 b = np.transpose(yp2d)
```

```
130 c = np.transpose(x_2d)
131 d = np.transpose(y_2d)
132 fig1a = plt.figure()
133 axes = fig1a.add_axes([0.1,0.1,0.8,0.8])
134 #axes.plot(xp2d,yp2d,'r.')
135 #axes.plot(a,b,'r.')
136 axes.plot(x_2d,y_2d,'k-')
137 axes.plot(c,d,'k-')
138 #axes.plot(x_2d,y_2d,'b.')
139 #axes.plot(c,d,'b.')
140 axes.tick_params(axis="x", labelsize=15)
141 axes.tick_params(axis="y", labelsize=15)
142 axes.set_xlabel('x [m]', fontsize=20)
143 axes.set_ylabel('y [m]', fontsize=20)
144 axes.set_title('Computational mesh in the domain', fontsize=24)
145
146 dz=0.2/32
147 dx = np.zeros([ni,nj])
148 dy = np.zeros([ni,nj])
149 dx_dummy = x_2d[1:,1:] - x_2d[:-1,1:]
150 dy_dummy = y_2d[1:,1:] - y_2d[1:,:-1]
151 dx[1:-1,1:-1] = dx_dummy
152 dy[1:-1,1:-1] = dy_dummy
153 dx[0,:] = dx[1,:]
154 dx[:,0] = dx[:,1]
155 dx[-1,:] = dx[-2,:]
156 dx[:, -1] = dx[:, -2]
157 dy[0,:] = dy[1,:]
158 dy[:,0] = dy[:,1]
159 dy[-1,:] = dy[-2,:]
160 dy[:, -1] = dy[:, -2]
161 delta = (dx*dy*dx)**(1/3)
162 #delta = np.maximum.reduce([dx,dy,dz*np.ones([ni,nj])])
163
164 x065_off=np.genfromtxt("x065_off.dat",dtype=None,comments="#")
165 x066_off=np.genfromtxt("x066_off.dat",dtype=None,comments="#")
166 x080_off=np.genfromtxt("x080_off.dat",dtype=None,comments="#")
167 x090_off=np.genfromtxt("x090_off.dat",dtype=None,comments="#")
168 x100_off=np.genfromtxt("x100_off.dat",dtype=None,comments="#")
169 x110_off=np.genfromtxt("x110_off.dat",dtype=None,comments="#")
170 x120_off=np.genfromtxt("x120_off.dat",dtype=None,comments="#")
171 x130_off=np.genfromtxt("x130_off.dat",dtype=None,comments="#")
172
173
174 # compute the gradient
175 dudx ,dudy=dphidx_dy(x_2d,y_2d,u_2d)
176
177
178 ****
179 # plot u
180 fig1b,ax1 = plt.subplots()
181 xx=0.65;
182 i1 = (np.abs(xx-xp2d[:,1])).argmin() # find index which closest fits xx
183 plt.plot(u_2d[i1,:],yp2d[i1,:],'b-')
184 plt.plot(x065_off[:,2],x065_off[:,1],'bo')
185 plt.xlabel("$U$")
186 plt.ylabel("$y$")
187 plt.title("$x=0.65$")
188 plt.axis([0, 1.3,0.115,0.2])
189 # Create inset of width 30% and height 40% of the parent axes' bounding box
190 # at the lower left corner (loc=3)
191 # upper left corner (loc=2)
192 # use borderpad=1, i.e.
193 # 22 points padding (as 22pt is the default fontsize) to the parent axes
194 axins1 = inset_axes(ax1, width="40%", height="30%", loc=2, borderpad=1)
195 plt.plot(u_2d[i1,:],yp2d[i1,:],'b-')
```

```

196 plt.axis([0, 1.3, 0.115, 0.13])
197 # reduce fontsize
198 axins1.tick_params(axis = 'both', which = 'major', labelsize = 10)
199 # Turn ticklabels of insets off
200 axins1.tick_params(labelleft=False, labelbottom=False)
201 plt.plot(x065_off[:,2],x065_off[:,1],'bo')
202 #plt.savefig('u065_hump_python.eps',bbox_inches='tight')
203
204 # %% Assignment T4 - Location of Interface
205 # RANS to LES where fk goes down from 1 to 0.4
206 C_mu = 0.09
207 k_res_2d = 0.5*(uu_2d+vv_2d+ww_2d)
208 k_tot_2d = k_model_2d + k_res_2d
209 fk_def_2d = k_model_2d / k_tot_2d
210 Lt = (k_tot_2d**1.5)/diss_2d
211 fk_ref_2d = (C_mu**-0.5) * (delta/Lt)**(2/3)
212
213 switch_i = np.argmax((fk_2d[1:-1,1:-1]<=0.4),axis=1) + 1
214 switch_wd = np.zeros(len(switch_i))
215 for i in range (nim1-1):
216     switch_wd[i] = yp2d[i+1,switch_i[i]] - yp2d[i+1,0]
217
218 fig2,axes = plt.subplots(1,1,constrained_layout=True)
219 axes.plot(xp2d[1:-1,0],switch_wd,label='D-PANS model')
220 axes.legend(fontsize=16,loc='best')
221 axes.xaxis.set_minor_locator(AutoMinorLocator())
222 axes.yaxis.set_minor_locator(AutoMinorLocator())
223 axes.tick_params(labelsize='medium')
224 axes.grid(True, linestyle='-.')
225 axes.set_xlabel('x (m)')
226 axes.set_ylabel(r'$y-y_{wall}$ (m)')
227 fig2.suptitle(r"Location of the switch (from RANS to LES) wrt the lower wall", fontsize =22)
228
229 xx = np.array([0,0.65,0.8,0.9,1,1.1,1.2,1.3,2])
230 i1 = list(map(lambda m : (np.abs(m-xp2d[:,1])).argmin(),xx))
231
232 fig3,axes = plt.subplots(2,4,constrained_layout=True)
233 axes = axes.reshape(-1)
234 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')
235 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')
236 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')
237 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')
238 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')
239 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')
240 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')
241 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')
242 axes[0].plot(fk_def_2d[i1[0],1:],yp2d[i1[0],1:]-yp2d[i1[0],0],'r-o',ms=4,label=r'$f_{\{k\}} = k/k_{tot}$')
243 axes[1].plot(fk_def_2d[i1[1],1:],yp2d[i1[1],1:]-yp2d[i1[1],0],'r-o',ms=4,label=r'$f_{\{k\}} = k/k_{tot}$')
244 axes[2].plot(fk_def_2d[i1[2],1:],yp2d[i1[2],1:]-yp2d[i1[2],0],'r-o',ms=4,label=r'$f_{\{k\}} = k/k_{tot}$')
245 axes[3].plot(fk_def_2d[i1[3],1:],yp2d[i1[3],1:]-yp2d[i1[3],0],'r-o',ms=4,label=r'$f_{\{k\}} = k/k_{tot}$')
246 axes[4].plot(fk_def_2d[i1[4],1:],yp2d[i1[4],1:]-yp2d[i1[4],0],'r-o',ms=4,label=r'$f_{\{k\}} = k/k_{tot}$')
247 axes[5].plot(fk_def_2d[i1[5],1:],yp2d[i1[5],1:]-yp2d[i1[5],0],'r-o',ms=4,label=r'$f_{\{k\}} = k/k_{tot}$')

```

```

= k/k_{tot}$')
248 axes[6].plot(fk_def_2d[i1[6],1:],yp2d[i1[6],1:]-yp2d[i1[6],0],'r-o',ms=4,label=r'$f_{\{k\}}$'
= k/k_{tot}$')
249 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}$')
250 for ax,xloc in zip(axes,xx):
251     ax.plot(0.4*np.ones(20),np.linspace(0,0.2,20)-yp2d[i1[0],0],'k--',label=r'$f_{\{k\}}$' =
0.4')
252     ax.xaxis.set_minor_locator(AutoMinorLocator())
253     ax.yaxis.set_minor_locator(AutoMinorLocator())
254     ax.legend(title='At X = '+str(xloc)+ ' m',title_fontsize=14,fontsize=14,loc='best')
255     ax.tick_params(labelsize='medium')
256     ax.grid(True, linestyle='-.')
257     ax.set ylim([0,0.2])
258     if (ax==axes[0] or ax==axes[4]):
259         ax.set_ylabel("$y-y_{wall}$")
260     if (ax==axes[4] or ax==axes[5] or ax==axes[6] or ax==axes[7]):
261         ax.set xlabel("$f_{\{k\}}$")
262 fig3.suptitle(r"$f_{\{k\}}$ computed at various locations", fontsize=22)
263
264 fig4, axes = plt.subplots(2,4, constrained_layout=True)
265 axes = axes.reshape(-1)
266 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')
267 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')
268 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')
269 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')
270 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')
271 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')
272 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')
273 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')
274 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}$')
275 axes[1].plot(fk_ref_2d[i1[1],1:],yp2d[i1[1],1:]-yp2d[i1[1],0],'r-o',ms=4,label=r'$f_{\{k\}}$'
= C_{\mu}^{-0.5}(\frac{\Delta}{L_{\{t\}}})^{2/3}$')
276 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}$')
277 axes[3].plot(fk_ref_2d[i1[3],1:],yp2d[i1[3],1:]-yp2d[i1[3],0],'r-o',ms=4,label=r'$f_{\{k\}}$'
= C_{\mu}^{-0.5}(\frac{\Delta}{L_{\{t\}}})^{2/3}$')
278 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}$')
279 axes[5].plot(fk_ref_2d[i1[5],1:],yp2d[i1[5],1:]-yp2d[i1[5],0],'r-o',ms=4,label=r'$f_{\{k\}}$'
= C_{\mu}^{-0.5}(\frac{\Delta}{L_{\{t\}}})^{2/3}$')
280 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}$')
281 axes[7].plot(fk_ref_2d[i1[7],1:],yp2d[i1[7],1:]-yp2d[i1[7],0],'r-o',ms=4,label=r'$f_{\{k\}}$'
= C_{\mu}^{-0.5}(\frac{\Delta}{L_{\{t\}}})^{2/3}$')
282 for ax,xloc in zip(axes,xx):
283     ax.plot(0.4*np.ones(20),np.linspace(0,0.2,20)-yp2d[i1[0],0],'k--',label=r'$f_{\{k\}}$' =
0.4')
284     ax.xaxis.set_minor_locator(AutoMinorLocator())
285     ax.yaxis.set_minor_locator(AutoMinorLocator())
286     ax.legend(title='At X = '+str(xloc)+ ' m',title_fontsize=14,fontsize=14,loc='best')
287     ax.tick_params(labelsize='medium')
288     ax.grid(True, linestyle='-.')
289     ax.set ylim([0,0.2])
290     ax.set_xlim([0,1])
291     if (ax==axes[0] or ax==axes[4]):
292         ax.set_ylabel("$y-y_{wall}$")

```

```
293     if (ax==axes[4] or ax==axes[5] or ax==axes[6] or ax==axes[7]):  
294         ax.set_xlabel("$f_{\{k\}}$")  
295 fig4.suptitle(r"$f_{\{k\}}$ computed at various locations", fontsize=22)  
296  
297 # %% Assignment T5 - Location of Interface in DES and DDES  
298 # SA-DES model  
299 C_DES_SA = 0.65  
300 delta_SA = C_DES_SA * np.maximum.reduce([dx,dy,dz*np.ones([ni,nj])])  
301 d_tilda = np.minimum(yp2d-yp2d[:,0][:,None],delta_SA)  
302 switch_i_SA = (yp2d-yp2d[:,0][:,None]<=delta_SA).argmin(axis=1)  
303 switch_wd_SA = np.zeros(len(switch_i_SA))  
304 for i in range(ni):  
305     switch_wd_SA[i] = yp2d[i,switch_i_SA[i]] - yp2d[i,0]  
306  
307 # SST-DES model  
308 C_DES_SST = 0.61  
309 beta_star = 0.09  
310 omega_2d = (1/beta_star) * (diss_2d/k_model_2d)  
311 Lt_SST = (1/beta_star) * ((k_model_2d**0.5)/omega_2d)  
312 delta_SST = C_DES_SST * np.maximum.reduce([dx,dy,dz*np.ones([ni,nj])])  
313 term1 = Lt_SST/delta_SST  
314 F_SST = np.maximum(term1,1)  
315 switch_i_SST = (term1<=1).argmin(axis=1)  
316 switch_i_SST = np.where(switch_i_SST==0,109,switch_i_SST)  
317 index_SST = np.where(switch_i_SST!=109)  
318 switch_wd_SST = np.zeros(len(switch_i_SST))  
319 switch_wd_SST2 = np.zeros(len(switch_i_SST))  
320 for i in range(ni):  
321     switch_wd_SST[i] = yp2d[i,switch_i_SST[i]] - yp2d[i,0]  
322     switch_wd_SST2[i] = yp2d[i,switch_i_SST[i]]  
323 switch_wd_SST1 = np.ma.masked_where(switch_wd_SST > 0.5, switch_wd_SST)  
324 switch_wd_SST2 = np.ma.masked_where(switch_wd_SST2 > 0.5, switch_wd_SST2)  
325  
326 # DDES model  
327 dist = yp2d[:,0]-yp2d[:,0][:,None]  
328 term2 = 2*(Lt_SST[:,1:]/dist[:,1:])  
329 term3 = (500*viscos)/(omega_2d[:,1:]*(dist[:,1:]**2))  
330 eta = np.maximum(term2,term3)  
331 F2 = np.tanh(eta**2)  
332 term4 = term1[:,1:] * (1-F2)  
333 switch_i_DDES = (term4<=1).argmin(axis=1)  
334 switch_i_DDES = np.where(switch_i_DDES==0,109,switch_i_DDES)  
335 index_DDES = np.where(switch_i_DDES!=109)  
336 switch_wd_DDES = np.zeros(len(switch_i_SA))  
337 switch_wd_DDES2 = np.zeros(len(switch_i_SA))  
338 for i in range(ni):  
339     switch_wd_DDES[i] = yp2d[i,switch_i_DDES[i]] - yp2d[i,0]  
340     switch_wd_DDES2[i] = yp2d[i,switch_i_DDES[i]]  
341 switch_wd_DDES1 = np.ma.masked_where(switch_wd_DDES > 0.5, switch_wd_DDES)  
342 switch_wd_DDES2 = np.ma.masked_where(switch_wd_DDES2 > 0.5, switch_wd_DDES2)  
343  
344 # Length scales  
345 ls_DES = C_DES_SA * np.maximum.reduce([dx,dy,dz*np.ones([ni,nj])])  
346 ls_DPANS = (k_model_2d**1.5)/diss_2d  
347  
348 # finding boundary layer thickness  
349 blt_i = np.zeros(ni)  
350 blt_yc = np.zeros(ni)  
351 blt_wd = np.zeros(ni)  
352 for i in range(1,ni):  
353     blt_i[i] = np.max(np.where(((vis_2d[i,:]/viscos-1)<=1)==0))  
354     blt_yc[i] = yp2d[i,int(blt_i[i])]  
355     blt_wd[i] = blt_yc[i] - yp2d[i,0]  
356  
357 fig5,axes = plt.subplots(1,1,constrained_layout=True)  
358 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'$l_{DPANS}$')
397 axes[1].plot(ls_DPANS[i1[2],1:], yp2d[i1[2],1:]-yp2d[i1[2],0], 'r-o', ms=4, label=r'$l_{DPANS}$')
398 axes[2].plot(ls_DPANS[i1[3],1:], yp2d[i1[3],1:]-yp2d[i1[3],0], 'r-o', ms=4, label=r'$l_{DPANS}$')
399 axes[3].plot(ls_DPANS[i1[4],1:], yp2d[i1[4],1:]-yp2d[i1[4],0], 'r-o', ms=4, label=r'$l_{DPANS}$')
400 axes[4].plot(ls_DPANS[i1[5],1:], yp2d[i1[5],1:]-yp2d[i1[5],0], 'r-o', ms=4, label=r'$l_{DPANS}$')
401 axes[5].plot(ls_DPANS[i1[6],1:], yp2d[i1[6],1:]-yp2d[i1[6],0], 'r-o', ms=4, label=r'$l_{DPANS}$')
402 axes[6].plot(ls_DPANS[i1[7],1:], yp2d[i1[7],1:]-yp2d[i1[7],0], 'r-o', ms=4, label=r'$l_{DPANS}$')
403 axes[7].plot(ls_DPANS[i1[8],1:], yp2d[i1[8],1:]-yp2d[i1[8],0], 'r-o', ms=4, label=r'$l_{DPANS}$')
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')

```

```
    i1[3]]*np.ones(10), 'k--', ms=4, label='BL thickness')
407 axes[3].plot(np.linspace(np.min(ls_DPANS[i1[4],1:]),np.max(ls_DES[i1[4],1:]),10),blt_wd[
408     i1[4]]*np.ones(10), 'k--', ms=4, label='BL thickness')
409 axes[4].plot(np.linspace(np.min(ls_DPANS[i1[5],1:]),np.max(ls_DES[i1[5],1:]),10),blt_wd[
410     i1[5]]*np.ones(10), 'k--', ms=4, label='BL thickness')
411 axes[5].plot(np.linspace(np.min(ls_DPANS[i1[6],1:]),np.max(ls_DES[i1[6],1:]),10),blt_wd[
412     i1[6]]*np.ones(10), 'k--', ms=4, label='BL thickness')
413 axes[6].plot(np.linspace(np.min(ls_DPANS[i1[7],1:]),np.max(ls_DES[i1[7],1:]),10),blt_wd[
414     i1[7]]*np.ones(10), 'k--', ms=4, label='BL thickness')
415 axes[7].plot(np.linspace(np.min(ls_DPANS[i1[8],1:]),np.max(ls_DES[i1[8],1:]),10),blt_wd[
416     i1[8]]*np.ones(10), 'k--', ms=4, label='BL thickness')
417 for ax,xloc in zip(axes,xx[1:]):
418     ax.xaxis.set_minor_locator(AutoMinorLocator())
419     ax.yaxis.set_minor_locator(AutoMinorLocator())
420     ax.legend(title='At X = '+str(xloc)+ ' m',title_fontsize=14,fontsize=14,loc='best')
421     ax.tick_params(labelsize='medium')
422     ax.grid(True, linestyle='-.')
423     ax.set_xlim([0,0.015])
424     ax.set_ylim([0,0.2])
425     if ax==axes[7]:
426         ax.set_xlim([0,0.03])
427         ax.set_ylim([0,0.3])
428     if (ax==axes[0] or ax==axes[4]):
429         ax.set_ylabel("$y-y_{wall}$")
430     if (ax==axes[4] or ax==axes[5] or ax==axes[6] or ax==axes[7]):
431         ax.set_xlabel("Length scale")
432 fig6.suptitle(r"Length scales computed at various locations",fontsize=22)
```

For Assignment 2b : Section 2.7 to Section 2.8

```
1 % Diffuser, lada
2 clc
3 clear
4 close all
5
6 nk=34;
7 re = 9.36e+5;
8 viscos =1/re;
9
10 % Computational domain
11 load xy_hump_fine.dat
12 x1=xy_hump_fine(:,1);
13 y1=xy_hump_fine(:,2);
14
15 nim1=x1(1);
16 njm1=y1(1);
17
18 ni=nim1+1;
19 nj=njm1+1;
20
21 x=x1(2:end);
22 y=y1(2:end);
23
24 x_2d=reshape(x,nim1,njm1);
25 y_2d=reshape(y,nim1,njm1);
26
27 x_2d(:,nj)=x_2d(:,nj-1);
28 y_2d(:,nj)=y_2d(:,nj-1);
29 x_2d(ni,:)=x_2d(ni-1,:);
30 y_2d(ni,:)=y_2d(ni-1,:);
31
32 % compute cell centers
33 xp2d=zeros(ni,nj);
34 yp2d=zeros(ni,nj);
35
36 for jj=1:nj
37   for find_loc_i=1:ni
38     im1=max(find_loc_i-1,1);
39     jm1=max(jj-1,1);
40     c=min(find_loc_i,nim1);
41     j=min(jj,njm1);
42     xp2d(find_loc_i,jj)=0.25*(x_2d(c,j)+x_2d(im1,j)+x_2d(c,jm1)+x_2d(im1,jm1));
43     yp2d(find_loc_i,jj)=0.25*(y_2d(c,j)+y_2d(im1,j)+y_2d(c,jm1)+y_2d(im1,jm1));
44   end
45 end
46
47 zmax=0.2;
48 dz=zmax/(nk-2);
49 z=0:dz:zmax;
50
51 % Loading of all data files as recommended
52 % file 1
53 load u1_pans_iddes.mat
54 u1=u1_pans_iddes;
55 temp=reshape(u1,nk,nj,ni);
56 u3d1=permute(temp,[3 2 1]);
57
58 load v1_pans_iddes.mat
59 v1=v1_pans_iddes;
60 temp=reshape(v1,nk,nj,ni);
61 v3d1=permute(temp,[3 2 1]);
62
63 load w1_pans_iddes.mat
64 w1=w1_pans_iddes;
```

```
65 temp=reshape(w1,nk,nj,ni);
66 w3d1=permute(temp,[3 2 1]);
67
68 % file 2
69 load u2_pans_iddes.mat
70 u2=u2_pans_iddes;
71 temp=reshape(u2,nk,nj,ni);
72 u3d2=permute(temp,[3 2 1]);
73
74 load v2_pans_iddes.mat
75 v2=v2_pans_iddes;
76 temp=reshape(v2,nk,nj,ni);
77 v3d2=permute(temp,[3 2 1]);
78
79 load w2_pans_iddes.mat
80 w2=w2_pans_iddes;
81 temp=reshape(w2,nk,nj,ni);
82 w3d2=permute(temp,[3 2 1]);
83
84 load u3_pans_iddes.mat
85 u3=u3_pans_iddes;
86 temp=reshape(u3,nk,nj,ni);
87 u3d3=permute(temp,[3 2 1]);
88
89 load v3_pans_iddes.mat
90 v3=v3_pans_iddes;
91 temp=reshape(v3,nk,nj,ni);
92 v3d3=permute(temp,[3 2 1]);
93
94 load w3_pans_iddes.mat
95 w3=w3_pans_iddes;
96 temp=reshape(w3,nk,nj,ni);
97 w3d3=permute(temp,[3 2 1]);
98
99 % file 4
100 load u4_pans_iddes.mat
101 u4=u4_pans_iddes;
102 temp=reshape(u4,nk,nj,ni);
103 u3d4=permute(temp,[3 2 1]);
104
105 load v4_pans_iddes.mat
106 v4=v4_pans_iddes;
107 temp=reshape(v4,nk,nj,ni);
108 v3d4=permute(temp,[3 2 1]);
109
110 load w4_pans_iddes.mat
111 w4=w4_pans_iddes;
112 temp=reshape(w4,nk,nj,ni);
113 w3d4=permute(temp,[3 2 1]);
114
115 % file 5
116 load u5_pans_iddes.mat
117 u5=u5_pans_iddes;
118 temp=reshape(u5,nk,nj,ni);
119 u3d5=permute(temp,[3 2 1]);
120
121 load v5_pans_iddes.mat
122 v5=v5_pans_iddes;
123 temp=reshape(v5,nk,nj,ni);
124 v3d5=permute(temp,[3 2 1]);
125
126 load w5_pans_iddes.mat
127 w5=w5_pans_iddes;
128 temp=reshape(w5,nk,nj,ni);
129 w3d5=permute(temp,[3 2 1]);
130
```

```
131 % file 6
132 load u6_pans_iddes.mat
133 u6=u6_pans_iddes;
134 temp=reshape(u6,nk,nj,ni);
135 u3d6=permute(temp,[3 2 1]);
136
137 load v6_pans_iddes.mat
138 v6=v6_pans_iddes;
139 temp=reshape(v6,nk,nj,ni);
140 v3d6=permute(temp,[3 2 1]);
141
142 load w6_pans_iddes.mat
143 w6=w6_pans_iddes;
144 temp=reshape(w6,nk,nj,ni);
145 w3d6=permute(temp,[3 2 1]);
146
147 % file 7
148 load u7_pans_iddes.mat
149 u7=u7_pans_iddes;
150 temp=reshape(u7,nk,nj,ni);
151 u3d7=permute(temp,[3 2 1]);
152
153 load v7_pans_iddes.mat
154 v7=v7_pans_iddes;
155 temp=reshape(v7,nk,nj,ni);
156 v3d7=permute(temp,[3 2 1]);
157
158 load w7_pans_iddes.mat
159 w7=w7_pans_iddes;
160 temp=reshape(w7,nk,nj,ni);
161 w3d7=permute(temp,[3 2 1]);
162
163 % file 8
164 load u8_pans_iddes.mat
165 u8=u8_pans_iddes;
166 temp=reshape(u8,nk,nj,ni);
167 u3d8=permute(temp,[3 2 1]);
168
169 load v8_pans_iddes.mat
170 v8=v8_pans_iddes;
171 temp=reshape(v8,nk,nj,ni);
172 v3d8=permute(temp,[3 2 1]);
173
174 load w8_pans_iddes.mat
175 w8=w8_pans_iddes;
176 temp=reshape(w8,nk,nj,ni);
177 w3d8=permute(temp,[3 2 1]);
178
179 % % merge 2 files. This means than new nk = 2*nk
180 u3d=cat(3,u3d1,u3d2,u3d3,u3d4,u3d5,u3d6,u3d7,u3d8);
181 v3d=cat(3,v3d1,v3d2,v3d3,v3d4,v3d5,v3d6,v3d7,v3d8);
182 w3d=cat(3,w3d1,w3d2,w3d3,w3d4,w3d5,w3d6,w3d7,w3d8);
183 clear u3d1 u3d2 v3d1 v3d2 w3d1 w3d2
184
185 % merge 8 files
186 % u3d=cat(3,u3d1,u3d2,u3d3,u3d4,u3d5,u3d6,u3d7,u3d8);
187 % v3d=cat(3,v3d1,v3d2,v3d3,v3d4,v3d5,v3d6,v3d7,v3d8);
188 % w3d=cat(3,w3d1,w3d2,w3d3,w3d4,w3d5,w3d6,w3d7,w3d8);
189 clear u3d1 u3d2 u3d3 u3d4 u3d5 u3d6 u3d7 u3d8
190 clear v3d1 v3d2 v3d3 v3d4 v3d5 v3d6 v3d7 v3d8
191 clear w3d1 w3d2 w3d3 w3d4 w3d5 w3d6 w3d7 w3d8
192 clear u1 u2 u3 u4 u5 u6 u7 u8
193 clear v1 v2 v3 v4 v5 v6 v7 v8
194 clear w1 w2 w3 w4 w5 w6 w7 w8
195 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
```

```
196 clear v1_pans_iddes v2_pans_iddes v3_pans_iddes v4_pans_iddes v5_pans_iddes
197     v6_pans_iddes v7_pans_iddes v8_pans_iddes
198 clear w1_pans_iddes w2_pans_iddes w3_pans_iddes w4_pans_iddes w5_pans_iddes
199     w6_pans_iddes w7_pans_iddes w8_pans_iddes
200
201 nk=size(u3d,3);
202
203 % u3d(ni,nj,nk), v3d, w3d are loaded
204
205 % i, j, and k correspond roughly to x, y, z (recall that the mesh is curvilinear in x &
206 % y)
207
208 % compute x, y gradients at grid plane k
209 for k=2:nk-1
210     [dudx_2d,dudy_2d] = dphidx_dy(x_2d,y_2d,u3d(:,:,k),ni,nj);
211     [dvdx_2d,dvdy_2d] = dphidx_dy(x_2d,y_2d,v3d(:,:,k),ni,nj);
212     [dwdx_2d,dwdy_2d] = dphidx_dy(x_2d,y_2d,w3d(:,:,k),ni,nj);
213
214     [d2udx2_2d d2udxdy_2d]=dphidx_dy(x_2d,y_2d,dudx_2d,ni,nj);
215     [dummy d2udy2_2d]=dphidx_dy(x_2d,y_2d,dudy_2d,ni,nj);
216
217 % corrected
218     [d2vdx2_2d d2vdxdy_2d]=dphidx_dy(x_2d,y_2d,dvdx_2d,ni,nj);
219     [dummy d2vdy2_2d]=dphidx_dy(x_2d,y_2d,dvdy_2d,ni,nj);
220
221 % corrected
222     [d2wdx2_2d d2wdx dy_2d]=dphidx_dy(x_2d,y_2d,dwdx_2d,ni,nj);
223     [dummy d2wdy2_2d]=dphidx_dy(x_2d,y_2d,dwdy_2d,ni,nj);
224
225 dudx(:,:,k)=dudx_2d;
226 dudy(:,:,k)=dudy_2d;
227
228 dvdx(:,:,k)=dvdx_2d;
229 dvdy(:,:,k)=dvdy_2d;
230
231 dwdx(:,:,k)=dwdx_2d;
232 dwdy(:,:,k)=dwdy_2d;
233
234 d2udx2(:,:,k)=d2udx2_2d;
235 d2udy2(:,:,k)=d2udy2_2d;
236 d2udxdy(:,:,k)=d2udxdy_2d;
237
238 d2vdx2(:,:,k)=d2vdx2_2d;
239 d2vdy2(:,:,k)=d2vdy2_2d;
240 d2vdxdy(:,:,k)=d2vdxdy_2d;
241
242 d2wdx2(:,:,k)=d2wdx2_2d;
243 d2wdy2(:,:,k)=d2wdy2_2d;
244 d2wdx dy(:,:,k)=d2wdx dy_2d;
245
246 end
247
248 [dummy dummy dudz]=gradient(u3d,1,1,dz);
249 [dummy dummy dvdz]=gradient(v3d,1,1,dz);
250 [dummy dummy dwdz]=gradient(w3d,1,1,dz);
251
252 dudz=dudz(:,:,1:nk-1);
253 dvdz=dvdz(:,:,1:nk-1);
254 dwdz=dwdz(:,:,1:nk-1);
255
256 [dummy dummy d2udxdz]=gradient(dudx,1,1,dz);
257 [dummy dummy d2udydz]=gradient(dudy,1,1,dz);
```

```

259 [dummy dummy d2udz2]=gradient(dudz,1,1,dz);
260
261 [dummy dummy d2vdxdz]=gradient(dvdx,1,1,dz);
262 [dummy dummy d2vdydz]=gradient(dvdy,1,1,dz);
263 [dummy dummy d2vdz2]=gradient(dvdz,1,1,dz);
264
265 [dummy dummy d2wdx2]=gradient(dwrx,1,1,dz);
266 [dummy dummy d2wdydz]=gradient(dwdy,1,1,dz);
267 [dummy dummy d2wdz2]=gradient(dwdz,1,1,dz);
268
269 s11=dudx;
270 s12=0.5*(dudy+dvdx);
271 s13=0.5*(dudz+dwdx);
272 s21=s12;
273 s22=dvdy;
274 s23=0.5*(dvdz+dwdy);
275 s31=s13;
276 s32=s23;
277 s33=dwdz;
278
279 ss=(2*(s11.^2+s12.^2+s13.^2+s21.^2+s22.^2+s23.^2+s31.^2+s32.^2+s33.^2).^0.5);
280
281 u_1_term=(d2udx2+d2udy2+d2udz2).^2;
282 v_1_term=(d2vdx2+d2vdy2+d2vdz2).^2;
283 w_1_term=(d2wdx2+d2wdy2+d2wdz2).^2;
284
285 u_2_term=(d2udx2+d2udy2+d2udz2 + 2*d2udxdy + 2*d2udxdz + 2*d2udydz).^2;
286 v_2_term=(d2vdx2+d2vdy2+d2vdz2 + 2*d2vdx2 + 2*d2vdx2 + 2*d2vdydz).^2;
287 w_2_term=(d2wdx2+d2wdy2+d2wdz2 + 2*d2wdx2 + 2*d2wdx2 + 2*d2wdydz).^2;
288
289 termu_b=d2udx2.^2+d2udy2.^2+d2udz2.^2;
290 termv_b=d2vdx2.^2+d2vdy2.^2+d2vdz2.^2;
291 termw_b=d2wdx2.^2+d2wdy2.^2+d2wdz2.^2;
292
293 usd_1=(u_1_term+v_1_term+w_1_term).^0.5;
294 usd_2=(u_2_term+v_2_term+w_2_term).^0.5;
295 usd_b=(termu_b+termv_b+termw_b).^0.5;
296
297 L_vk3d_1=0.4*ss./usd_1;
298 L_vk3d_2=0.4*ss./usd_2;
299 L_vk3d_b=0.4*ss./usd_b;
300
301 L_vk3d_spanwise_1=mean(L_vk3d_1,3);
302 L_vk3d_spanwise_2=mean(L_vk3d_2,3);
303
304 load vectz_fine.dat
305 vectz=vectz_fine;
306 nstep=vectz(1);
307 ni=vectz(2);
308 nj=vectz(3);
309 nk=vectz(4);
310 n=length(vectz);
311
312 nn=14;
313 nst=3;
314 iu=nst+2:nn:n;
315 iv=nst+3:nn:n;
316 ifk=nst+4:nn:n;
317 iuu=nst+5:nn:n;
318 ivv=nst+6:nn:n;
319 iww=nst+7:nn:n;
320 iuv=nst+8:nn:n;
321 ip=nst+9:nn:n;
322 ik=nst+10:nn:n;
323 ivis=nst+11:nn:n;
324 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,:);

```

```

391
392 dy=diff(yp2d,1,2);
393 dy(:,nj)=dy(:,nj-1);
394 dy=repmat(dy,[1 1 nk-1]);
395
396 dx2=dx.^2;
397 dy2=dy.^2;
398 dz2=dz.^2;
399
400 C_DES = 0.67;
401 l_s_PANS = (k_model_2d.^1.5)./diss_2d; % length scale PANS
402 l_s_DES = C_DES*del; % length scale DES
403
404 dx_2d = mean(dx,3);
405 dy_2d = mean(dy,3);
406 dz = 0.2/32;
407 del = max(max(dx_2d, dy_2d),dz);
408
409 figure(600)
410 subplot(2,2,1)
411 exp_locx=0.65;
412 find_loc_i = find(xp2d(:,1) < exp_locx);
413 c=find_loc_i(end);
414 plot(yp2d(c,:),L_vk3d_spanwise_1(c,:),'r','linewidth',1.5)
415 hold on
416 plot(yp2d(c,:),L_vk3d_spanwise_2(c,:),'b','linewidth',1.5)
417 plot(yp2d(c,:),L_vk_3d_m_1(c,:),'k','linewidth',1.5)
418 plot(yp2d(c,:),L_vk_3d_m_2(c,:),'g','linewidth',1.5)
419 plot(yp2d(c,:),l_s_PANS(c,:),'c','linewidth',1.5)
420 plot(yp2d(c,:),l_s_DES(c,:),'m','linewidth',1.5)
421 ylabel('Length scale','fontsize',15)
422 xlabel('y','fontsize',15)
423 title('Location x=0.65')
424 h1=gca;
425 set(h1,'fontsize',15)
426 grid on
427 hold off
428
429 subplot(2,2,2)
430 exp_locx=0.8;
431 find_loc_i = find(xp2d(:,1) < exp_locx);
432 c=find_loc_i(end);
433 plot(yp2d(c,:),L_vk3d_spanwise_1(c,:),'r','linewidth',1.5)
434 hold on
435 plot(yp2d(c,:),L_vk3d_spanwise_2(c,:),'b','linewidth',1.5)
436 plot(yp2d(c,:),L_vk_3d_m_1(c,:),'k','linewidth',1.5)
437 plot(yp2d(c,:),L_vk_3d_m_2(c,:),'g','linewidth',1.5)
438 plot(yp2d(c,:),l_s_PANS(c,:),'c','linewidth',1.5)
439 plot(yp2d(c,:),l_s_DES(c,:),'m','linewidth',1.5)
440 ylabel('Length scale','fontsize',15)
441 xlabel('y','fontsize',15)
442 title('Location x=0.8')
443 h1=gca;
444 set(h1,'fontsize',15)
445 grid on
446 hold off
447
448 subplot(2,2,3)
449 exp_locx=0.9;
450 find_loc_i = find(xp2d(:,1) < exp_locx);
451 c=find_loc_i(end);
452 plot(yp2d(c,:),L_vk3d_spanwise_1(c,:),'r','linewidth',1.5)
453 hold on
454 plot(yp2d(c,:),L_vk3d_spanwise_2(c,:),'b','linewidth',1.5)
455 plot(yp2d(c,:),L_vk_3d_m_1(c,:),'k','linewidth',1.5)
456 plot(yp2d(c,:),L_vk_3d_m_2(c,:),'g','linewidth',1.5)

```

```
457 plot(yp2d(c,:),l_s_PANS(c,:),'c','linewidth',1.5)
458 plot(yp2d(c,:),l_s_DES(c,:),'m','linewidth',1.5)
459 ylabel('Length scale','fontsize',15)
460 xlabel('y','fontsize',15)
461 title('Location x=0.9')
462 h1=gca;
463 set(h1,'fontsize',15)
464 grid on
465 hold off
466
467 subplot(2,2,4)
468 exp_locx=1;
469 find_loc_i = find(xp2d(:,1) < exp_locx);
470 c=find_loc_i(end);
471 plot(yp2d(c,:),L_vk3d_spanwise_1(c,:),'r','linewidth',1.5)
472 hold on
473 plot(yp2d(c,:),L_vk3d_spanwise_2(c,:),'b','linewidth',1.5)
474 plot(yp2d(c,:),L_vk_3d_m_1(c,:),'k','linewidth',1.5)
475 plot(yp2d(c,:),L_vk_3d_m_2(c,:),'g','linewidth',1.5)
476 plot(yp2d(c,:),l_s_PANS(c,:),'c','linewidth',1.5)
477 plot(yp2d(c,:),l_s_DES(c,:),'m','linewidth',1.5)
478 ylabel('Length scale','fontsize',15)
479 xlabel('y','fontsize',15)
480 title('Location x=1')
481 h1=gca;
482 set(h1,'fontsize',15)
483 grid on
484 hold off
485
486 figure(601)
487 subplot(2,2,1)
488 exp_locx=1.1;
489 find_loc_i = find(xp2d(:,1) < exp_locx);
490 c=find_loc_i(end);
491 plot(yp2d(c,:),L_vk3d_spanwise_1(c,:),'r','linewidth',1.5)
492 hold on
493 plot(yp2d(c,:),L_vk3d_spanwise_2(c,:),'b','linewidth',1.5)
494 plot(yp2d(c,:),L_vk_3d_m_1(c,:),'k','linewidth',1.5)
495 plot(yp2d(c,:),L_vk_3d_m_2(c,:),'g','linewidth',1.5)
496 plot(yp2d(c,:),l_s_PANS(c,:),'c','linewidth',1.5)
497 plot(yp2d(c,:),l_s_DES(c,:),'m','linewidth',1.5)
498 ylabel('Length scale','fontsize',15)
499 xlabel('y','fontsize',15)
500 title('Location x=1.1')
501 h1=gca;
502 set(h1,'fontsize',15)
503 grid on
504 hold off
505
506 subplot(2,2,2)
507 exp_locx=1.2;
508 find_loc_i = find(xp2d(:,1) < exp_locx);
509 c=find_loc_i(end);
510 plot(yp2d(c,:),L_vk3d_spanwise_1(c,:),'r','linewidth',1.5)
511 hold on
512 plot(yp2d(c,:),L_vk3d_spanwise_2(c,:),'b','linewidth',1.5)
513 plot(yp2d(c,:),L_vk_3d_m_1(c,:),'k','linewidth',1.5)
514 plot(yp2d(c,:),L_vk_3d_m_2(c,:),'g','linewidth',1.5)
515 plot(yp2d(c,:),l_s_PANS(c,:),'c','linewidth',1.5)
516 plot(yp2d(c,:),l_s_DES(c,:),'m','linewidth',1.5)
517 ylabel('Length scale','fontsize',15)
518 xlabel('y','fontsize',15)
519 title('Location x=1.2')
520 h1=gca;
521 set(h1,'fontsize',15)
522 grid on
```

```

523 hold off
524
525 subplot(2,2,3)
526 exp_locx=1.3;
527 find_loc_i = find(xp2d(:,1) < exp_locx);
528 c=find_loc_i(end);
529 plot(yp2d(c,:),L_vk3d_spanwise_1(c,:),'r','linewidth',1.5)
530 hold on
531 plot(yp2d(c,:),L_vk3d_spanwise_2(c,:),'b','linewidth',1.5)
532 plot(yp2d(c,:),L_vk_3d_m_1(c,:),'k','linewidth',1.5)
533 plot(yp2d(c,:),L_vk_3d_m_2(c,:),'g','linewidth',1.5)
534 plot(yp2d(c,:),l_s_PANS(c,:),'c','linewidth',1.5)
535 plot(yp2d(c,:),l_s_DES(c,:),'m','linewidth',1.5)
536 ylabel('Length scale','fontsize',15)
537 xlabel('y','fontsize',15)
538 title('Location x=1.3')
539 legend('$L_{vk, 3D,s, type1}$','$L_{vk, 3D,s, type2}$','$L_{vk, 3D,mean, type1}$',
      'fontsize',25,'Orientation','horizontal','location','southoutside','interpreter','
      latex')
540 h1=gca;
541 set(h1,'fontsize',15)
542 grid on
543 hold off
544
545 subplot(2,2,4)
546 exp_locx=1.4;
547 find_loc_i = find(xp2d(:,1) < exp_locx);
548 c=find_loc_i(end);
549 plot(yp2d(c,:),L_vk_3d_m_2(c,:),'g','linewidth',1.5)
550 hold on
551 plot(yp2d(c,:),l_s_PANS(c,:),'c','linewidth',1.5)
552 plot(yp2d(c,:),l_s_DES(c,:),'m','linewidth',1.5)
553 plot(yp2d(c,:),L_vk3d_spanwise_1(c,:),'r','linewidth',1.5)
554 plot(yp2d(c,:),L_vk3d_spanwise_2(c,:),'b','linewidth',1.5)
555 plot(yp2d(c,:),L_vk_3d_m_1(c,:),'k','linewidth',1.5)
556 ylabel('Length scale','fontsize',15)
557 xlabel('y','fontsize',15)
558 title('Location x=1.4','fontsize',12)
559 legend('$L_{vk, 3D,mean, type2 }$','$l_{PANS}$','$l_{DES}$','fontsize',25,'Orientation',
      'horizontal','location','southoutside','interpreter','latex')
560 h1=gca;
561 set(h1,'fontsize',15)
562 grid on
563 hold off
564
565 l_RANS = k_resolved_2d.^1.5 ./ diss_2d;
566
567
568 figure(602)
569 subplot(2,2,1)
570 exp_locx=0.65;
571 find_loc_i= find(xp2d(:,1) < exp_locx);
572 c=find_loc_i(end);
573 plot(yp2d(c,:),L_vk_3d_m_1(c,:),'r','linewidth',1.5)
574 hold on
575 plot(yp2d(c,:),L_vk_3d_m_2(c,:),'b','linewidth',1.5)
576 plot(yp2d(c,:),l_RANS(c,:),'k','linewidth',1.5)
577 axis([0.1 0.25 0 0.3])
578 ylabel('Length scale','fontsize',15)
579 xlabel('y','fontsize',15)
580 title('Location x=0.65','fontsize',12)
581 h1=gca;
582 set(h1,'fontsize',15)
583 grid on
584 hold off
585

```

```
586 subplot(2,2,2)
587 exp_locx=0.8;
588 find_loc_i= find(xp2d(:,1) < exp_locx);
589 c=find_loc_i(end);
590 plot(yp2d(c,:),L_vk_3d_m_1(c,:),'r','linewidth',1.5)
591 hold on
592 plot(yp2d(c,:),L_vk_3d_m_2(c,:),'b','linewidth',1.5)
593 plot(yp2d(c,:),l_RANS(c,:),'k','linewidth',1.5)
594 axis([0.1 0.25 0 0.3])
595 ylabel('Length scale','fontsize',15)
596 xlabel('y','fontsize',15)
597 title('Location x=0.8','fontsize',12)
598 h1=gca;
599 set(h1,'fontsize',15)
600 grid on
601 hold off
602
603 subplot(2,2,3)
604 exp_locx=0.9;
605 find_loc_i= find(xp2d(:,1) < exp_locx);
606 c=find_loc_i(end);
607 plot(yp2d(c,:),L_vk_3d_m_1(c,:),'r','linewidth',1.5)
608 hold on
609 plot(yp2d(c,:),L_vk_3d_m_2(c,:),'b','linewidth',1.5)
610 plot(yp2d(c,:),l_RANS(c,:),'k','linewidth',1.5)
611 axis([0.1 0.25 0 0.3])
612 ylabel('Length scale','fontsize',15)
613 xlabel('y','fontsize',15)
614 title('Location x=0.9','fontsize',12)
615 h1=gca;
616 set(h1,'fontsize',15)
617 grid on
618 hold off
619
620 subplot(2,2,4)
621 exp_locx=1.0;
622 find_loc_i= find(xp2d(:,1) < exp_locx);
623 c=find_loc_i(end);
624 plot(yp2d(c,:),L_vk_3d_m_1(c,:),'r','linewidth',1.5)
625 hold on
626 plot(yp2d(c,:),L_vk_3d_m_2(c,:),'b','linewidth',1.5)
627 plot(yp2d(c,:),l_RANS(c,:),'k','linewidth',1.5)
628 axis([0.1 0.25 0 0.3])
629 ylabel('Length scale','fontsize',15)
630 xlabel('y','fontsize',15)
631 title('Location x=1.0','fontsize',12)
632 h1=gca;
633 set(h1,'fontsize',15)
634 grid on
635 hold off
636
637 figure(603)
638 subplot(2,2,1)
639 exp_locx=1.1;
640 find_loc_i= find(xp2d(:,1) < exp_locx);
641 c=find_loc_i(end);
642 plot(yp2d(c,:),L_vk_3d_m_1(c,:),'r','linewidth',1.5)
643 hold on
644 plot(yp2d(c,:),L_vk_3d_m_2(c,:),'b','linewidth',1.5)
645 plot(yp2d(c,:),l_RANS(c,:),'k','linewidth',1.5)
646 axis([0.1 0.25 0 0.3])
647 ylabel('Length scale','fontsize',15)
648 xlabel('y','fontsize',15)
649 title('Location x=1.1','fontsize',12)
650 h1=gca;
651 set(h1,'fontsize',15)
```

```

652 grid on
653 hold off
654
655 subplot(2,2,2)
656 exp_locx=1.2;
657 find_loc_i= find(xp2d(:,1) < exp_locx);
658 c=find_loc_i(end);
659 plot(yp2d(c,:),L_vk_3d_m_1(c,:),'r','linewidth',1.5)
660 hold on
661 plot(yp2d(c,:),L_vk_3d_m_2(c,:),'b','linewidth',1.5)
662 plot(yp2d(c,:),l_RANS(c,:),'k','linewidth',1.5)
663 axis([0.1 0.25 0 0.3])
664 ylabel('Length scale','fontsize',15)
665 xlabel('y','fontsize',15)
666 title('Location x=1.2','fontsize',12)
667 h1=gca;
668 set(h1,'fontsize',15)
669 grid on
670 hold off
671
672 subplot(2,2,3)
673 exp_locx=1.3;
674 find_loc_i= find(xp2d(:,1) < exp_locx);
675 c=find_loc_i(end);
676 plot(yp2d(c,:),L_vk_3d_m_1(c,:),'r','linewidth',1.5)
677 hold on
678 plot(yp2d(c,:),L_vk_3d_m_2(c,:),'b','linewidth',1.5)
679 plot(yp2d(c,:),l_RANS(c,:),'k','linewidth',1.5)
680 axis([0.1 0.25 0 0.3])
681 ylabel('Length scale','fontsize',15)
682 xlabel('y','fontsize',15)
683 title('Location x=1.3','fontsize',12)
684 legend('$L_{vk, 3D, mean, type1}$','$L_{vk, 3D, mean, type2}$','$l_{RANS}$','fontsize',25,
    'Orientation','horizontal','interpreter','latex','location','southoutside')
685 h1=gca;
686 set(h1,'fontsize',15)
687 grid on
688 hold off
689
690 subplot(2,2,4)
691 exp_locx=2.0;
692 find_loc_i= find(xp2d(:,1) < exp_locx);
693 c=find_loc_i(end);
694 plot(yp2d(c,:),L_vk_3d_m_1(c,:),'r','linewidth',1.5)
695 hold on
696 plot(yp2d(c,:),L_vk_3d_m_2(c,:),'b','linewidth',1.5)
697 plot(yp2d(c,:),l_RANS(c,:),'k','linewidth',1.5)
698 axis([0.1 0.25 0 0.3])
699 ylabel('Length scale','fontsize',15)
700 xlabel('y','fontsize',15)
701 title('Location x=2.0','fontsize',12)
702 legend('$L_{vk, 3D, mean, type1}$','$L_{vk, 3D, mean, type2}$','$l_{RANS}$','fontsize',25,
    'Orientation','horizontal','interpreter','latex','location','southoutside')
703 h1=gca;
704 set(h1,'fontsize',15)
705 grid on
706 hold off

```