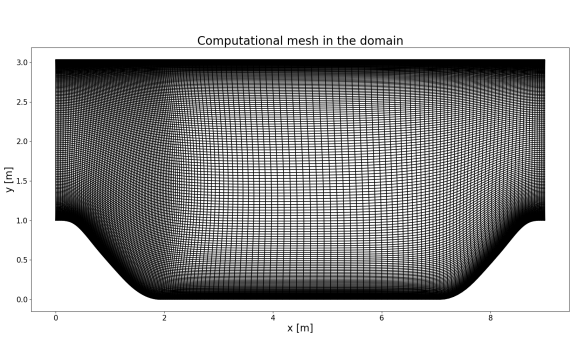
Chalmers University of Technology
MTF 270 - Turbulence Modeling

ASSIGNMENT 1
Case 3 : Flow over a Hill

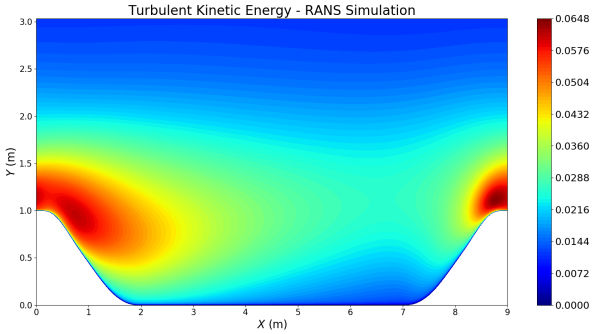
Group 14 - Project members:
Vishal Subramaniasivam

Date : April 4, 2022

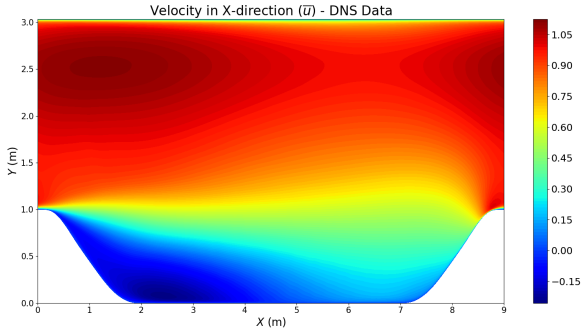
1 Introduction



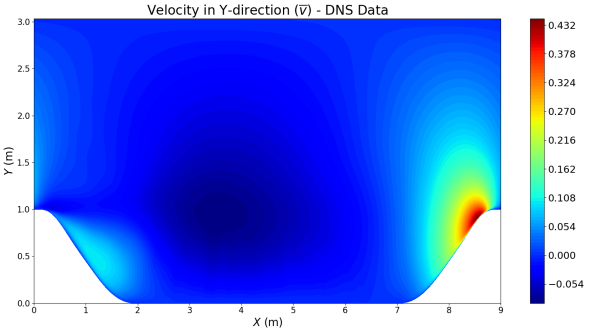
(a) Computational Mesh



(b) Turbulent Kinetic Energy



(c) Velocity in X-direction



(d) Velocity in Y-direction

Figure 1: Illustration of given data

The flow over a single hill is simulated in this project. DNS data for the fluctuating as well as the mean flow quantities are provided to us. From figure 1, we can observe that turbulence is present predominantly in two regions, the separation region near the first hill and the stagnation region near the second hill.

2 Results

2.1 Assignment 1.1

Normal Stresses(blue and yellow curve)

@ Lower wall

The normal viscous stress terms (viscos*dudx and viscos*dvdy) are negligible.From figure 4 ,the normal Reynolds stress terms $\rho\overline{u'u'}$ and $\rho\overline{v'v'}$ have a high value at the high turbulence regions specifically near the flow separation region (x:0m) and the re-circulation region (x:4.116m). Thus the resulting normal

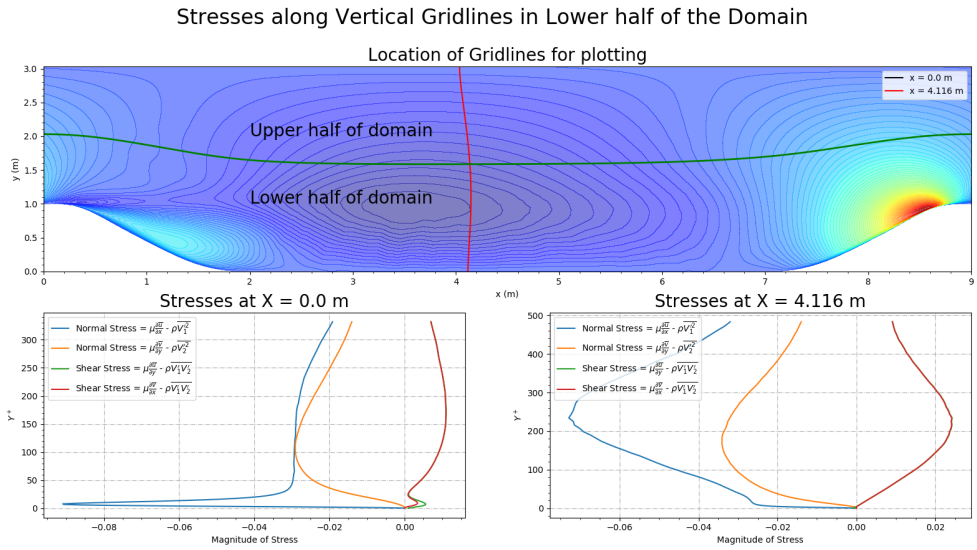


Figure 2: Stresses at different vertical lines - Lower half

stress term is highly negative.

@vertical line (x:0) : The high negative values in the blue curve near the wall is due to the high value of $\rho\overline{u'u'}$ near the beginning of the flow separation. At the same time the values near the wall at the same location for the yellow curve are not high. As the v component of the flow is high only in the log law region.

@vertical line (x:4.4116) The high negative values in the blue curve and yellow curves are due to the presence of the re-circulation region and separation between the re-circulation flow and the channel flow. The value of $\rho\overline{u'u'}$ and $\rho\overline{v'v'}$ are high in the log law region.

@ Upper wall

The value of $\rho\overline{u'u'}$ is high near the wall ,this causes the blue curve to have a high negative jump near the wall. Both $\rho\overline{u'u'}$ and $\rho\overline{v'v'}$ increase in magnitude in the log law region. This is evident in the blue and yellow curves.

Shear Stresses (green and red curves)

@ Lower wall and Upper wall

Since $\rho\overline{u'u'}$ is same as $\rho\overline{u'v'}$ and both are dominant in the log law region,both the green and the red curve are equal in the log law region.

The differences between the (dudy) term and the (dvdx) term can be seen in the non-coinciding green and red curve near the wall , in the viscous (inner region).

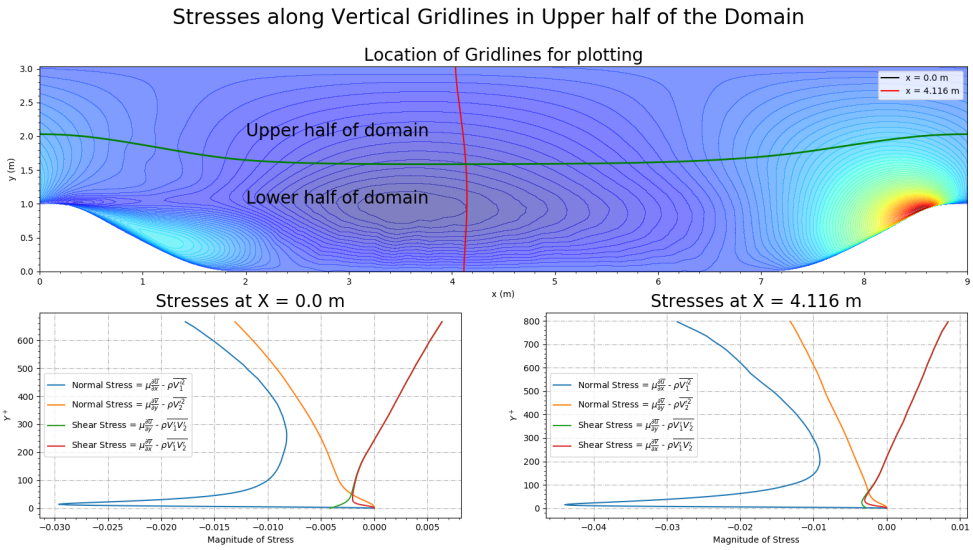
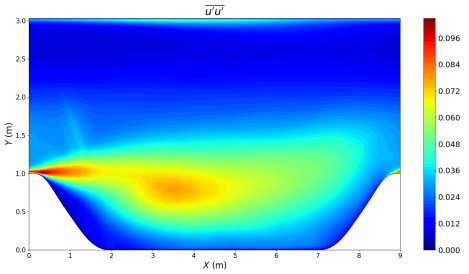
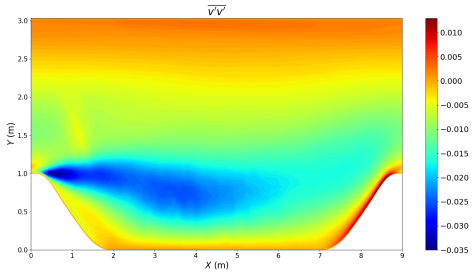


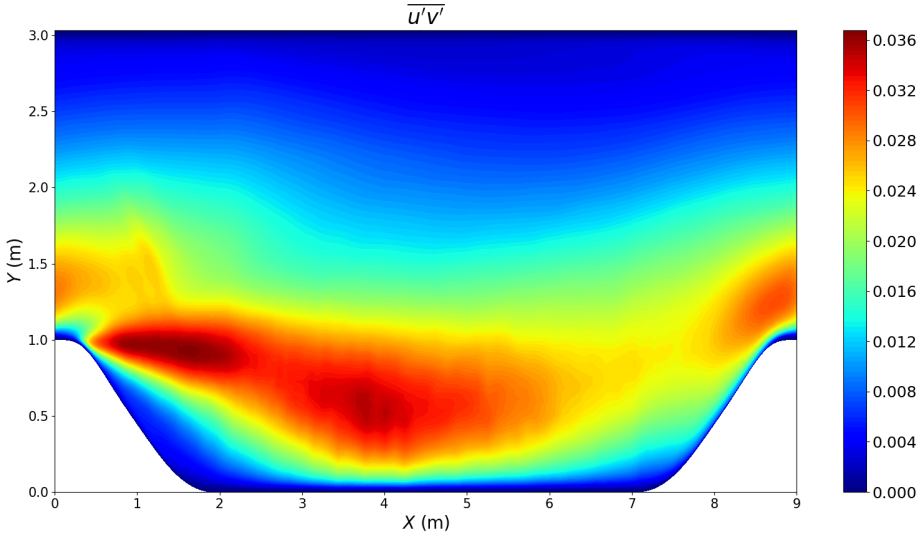
Figure 3: Stresses at different vertical lines - Upper half



(a) $\overline{u'u'}$



(b) $\overline{v'v'}$



(c) $\overline{u'v'}$

Figure 4: Effect of Limiting Function on Normal Stress and Turbulent Viscosity

2.2 Assignment 1.2

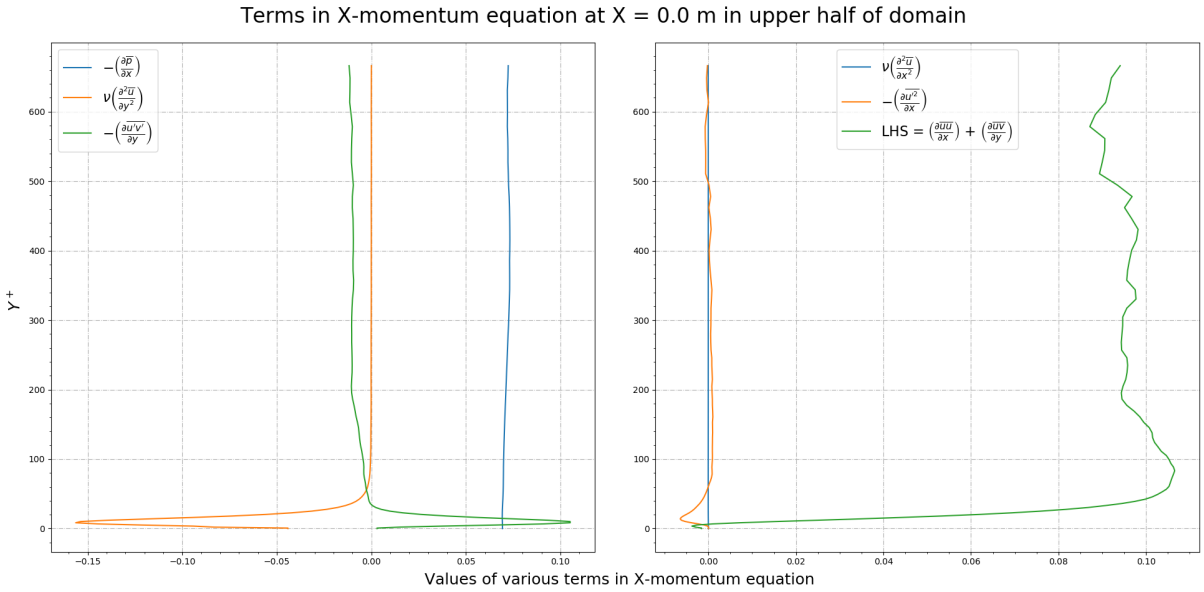


Figure 5: X momentum at X = 0 m in upper half of the domain

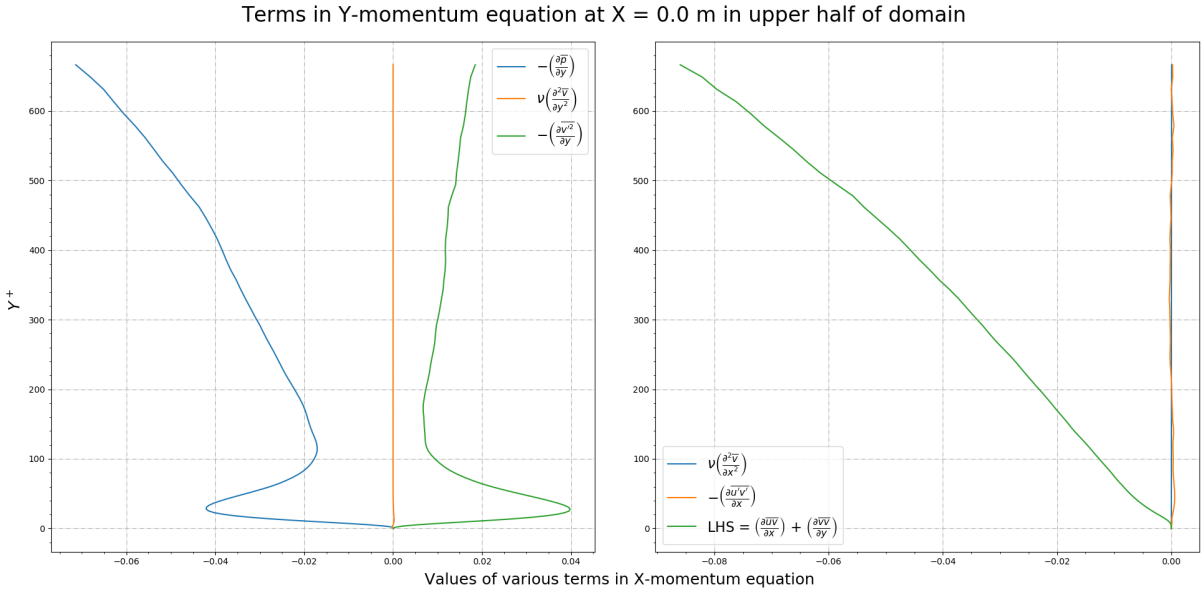


Figure 6: Y momentum at X = 0 m in upper half of the domain

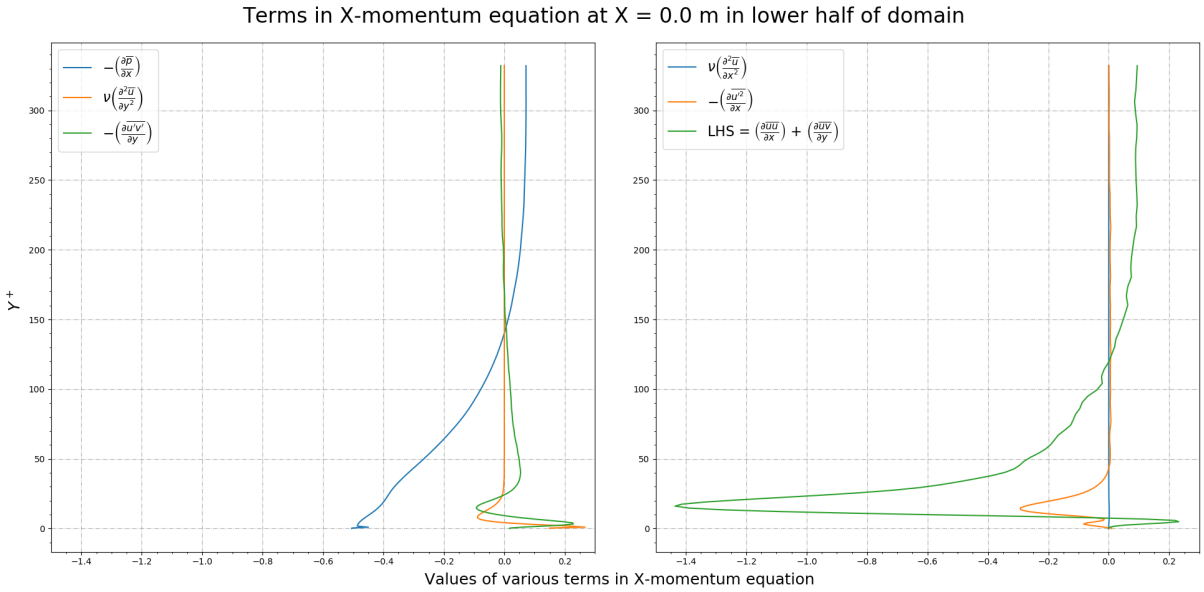


Figure 7: X momentum at X = 0 m in lower half of the domain

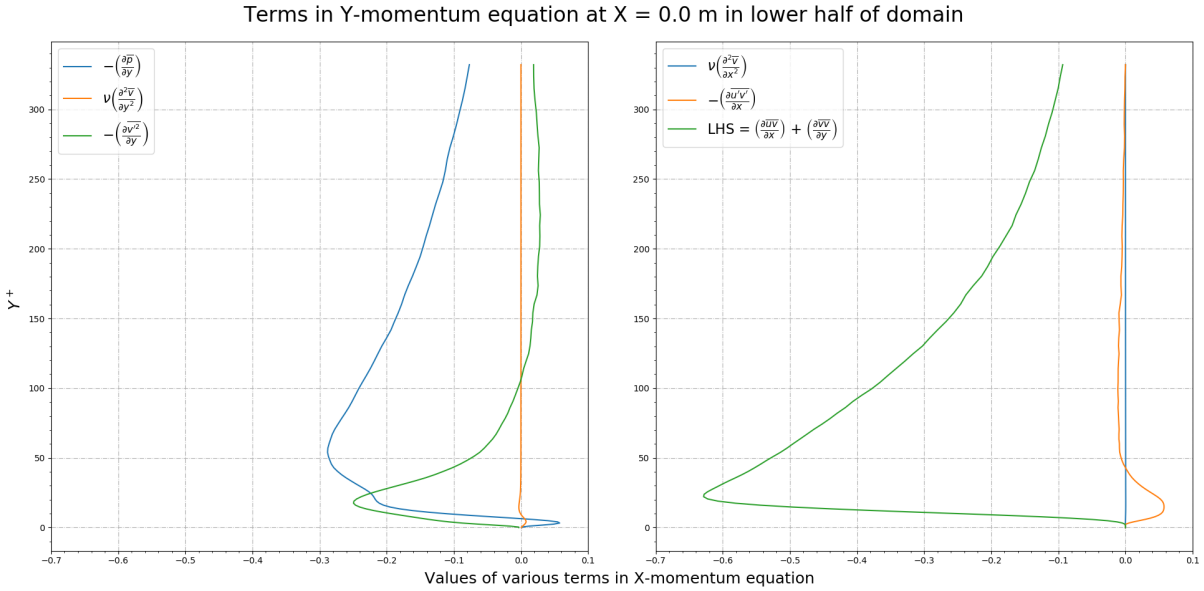


Figure 8: Y momentum at X = 0 m in lower half of the domain

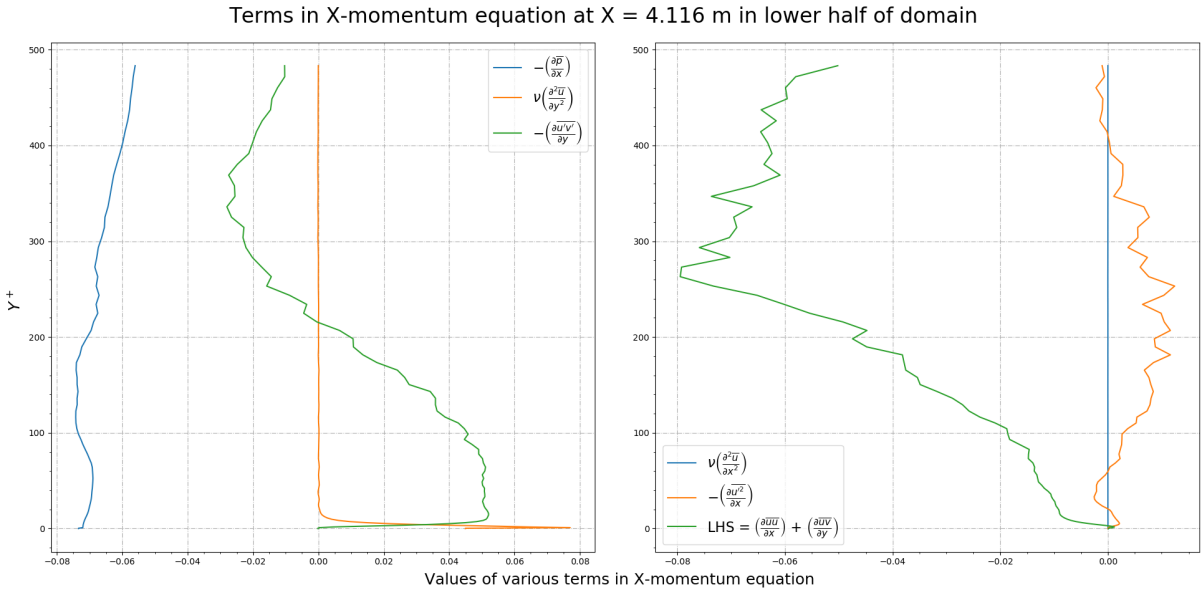


Figure 9: X momentum at X = 4.116 m in lower half of the domain

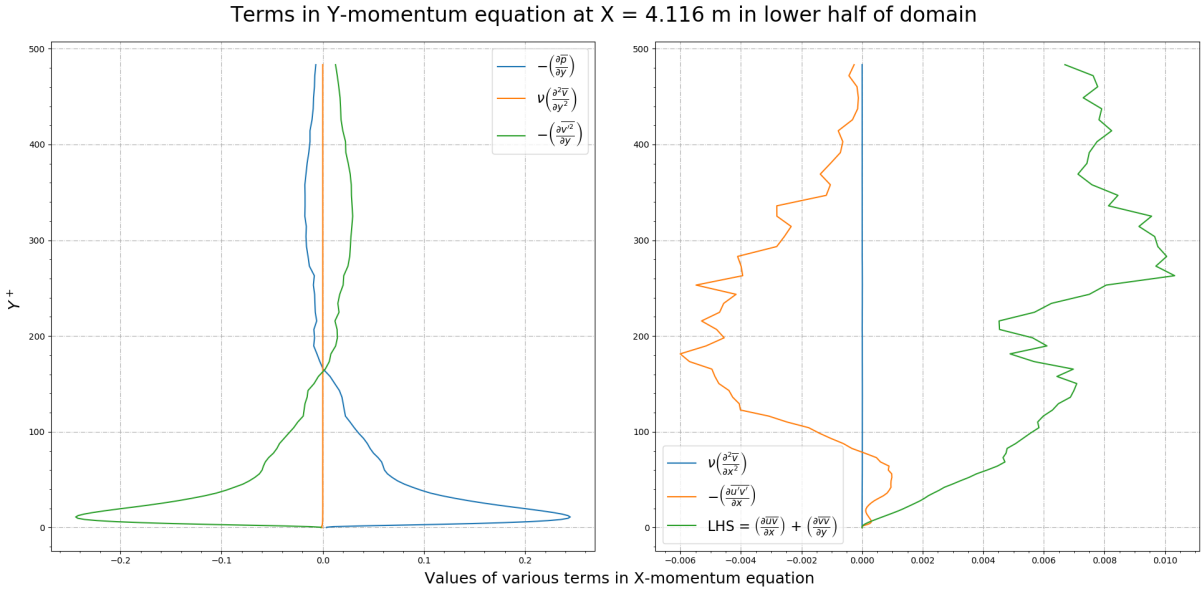


Figure 10: Y momentum at X = 4.116 m in lower half of the domain

@ Upper wall

Since the flow characteristics at the upper wall is similar to a channel flow, it is easier to start the discussion at the upper wall.

From figure 5, the pressure gradient term is seen to have a close to constant value along Y+, this is the pressure force that drives the force by creating a constant pressure drop.

Since the viscous force is dominant only in the near wall region (inner region).The only opposing force in the log law region is the gradient of the Reynolds shear force.

Near the wall (Viscous region), the Reynolds shear force is the driving force (it is the opposing force in log law region) and the opposing force is the viscous force.This is seen clearly in the plot.

The x gradients of the viscous and shear stresses are negligible low change in u with respect to x and low change in u' with respect to x

The LHS in both the X momentum and the Y momentum follow a similar behaviour to the pressure force, depicting the influence of pressure force in determining the direction of flow.

@ Lower wall

@ x=0.0 m, the flow is driven by the pressure force term, the separation and re-circulation in the flow downstream is felt upstream causing the pressure term to be negative (negative u direction flow). As the Y+ is increased the flow pressure term is back in the positive side depicting flow in the positive u direction.

The viscous force and the Reynolds force again follow the same characteristics as explained for the upper wall. The only difference here is the direction of the forces which are mainly due to the re circulation region.

Similar to the upper wall,The LHS in both the X momentum and the Y momentum follow a similar behaviour to the pressure force, depicting the influence of pressure force in determining the direction of flow.

@ x= 4.116m, similar to the discussion in the upper wall region, the Reynolds force is zero at the wall and increases till it teaches total wall force, at the same time the Viscous force reduces to zero in the log law region.

The pressure force term here is on the negative side of the plot depicting flow in the negative u direction, this is explained due to the re-circulation region present.

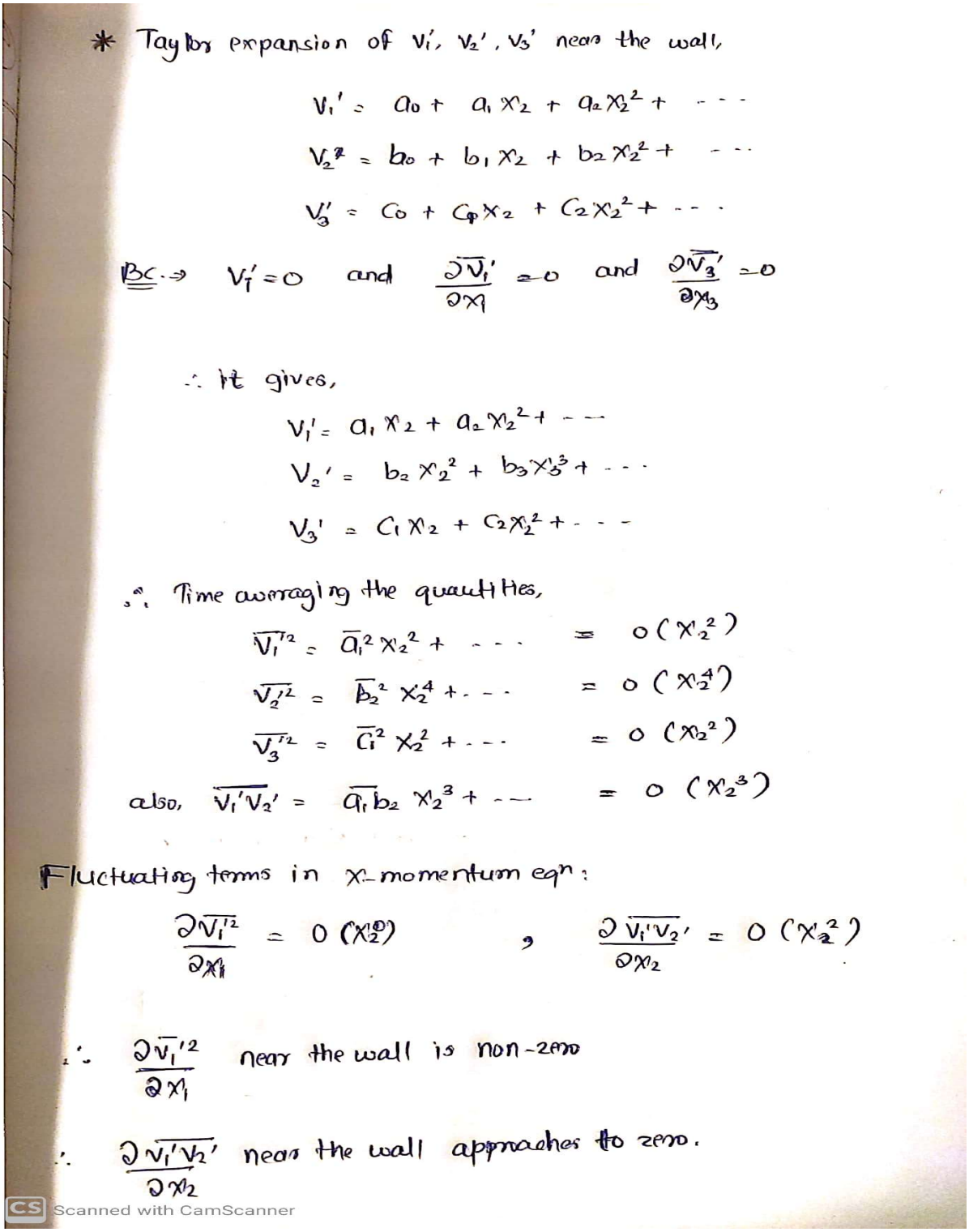


Figure 11: Taylor's expansion for fluctuating terms

2.3 Assignment 1.3

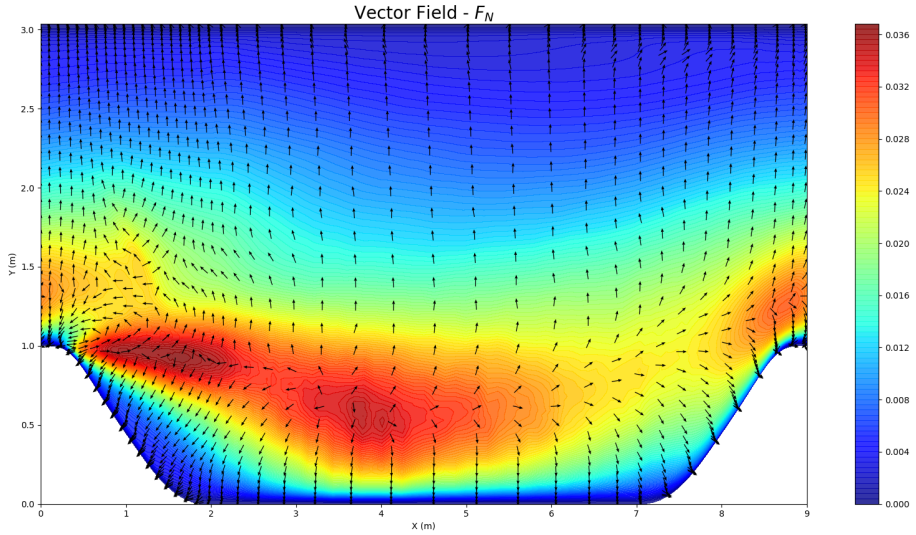


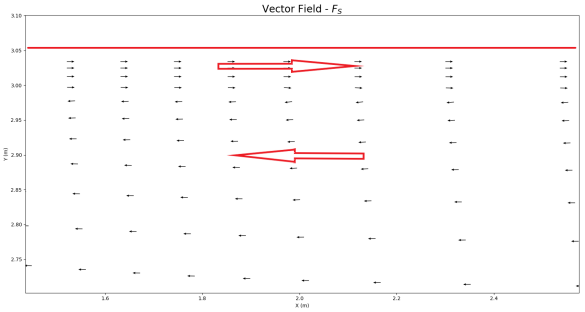
Figure 12: F.N Quiver plot over $\rho\overline{v'v'}$ contour

From Figure 12, the direction of the normal forces originate from the highest value of the viscous stress and point toward the wall.

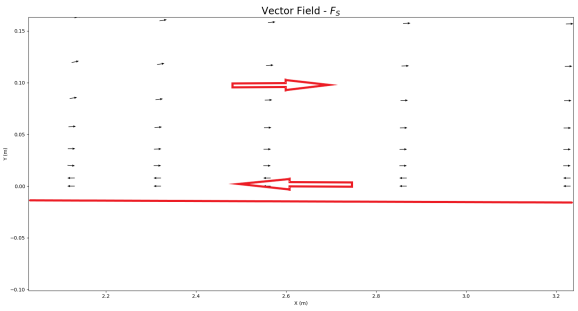
At the highest value of $\rho\overline{v'v'}$ at any vertical line, the direction of the Normal force vectors point away from the highest value towards the wall. As in nodes above the highest node, point towards the upper wall and nodes below the lowest node point towards the lower wall.

Since $\rho\overline{v'v'}$ is highest at locations with separated flow and re-circulation (areas with high v component), the direction change of the Normal Force vectors is preset in these regions.

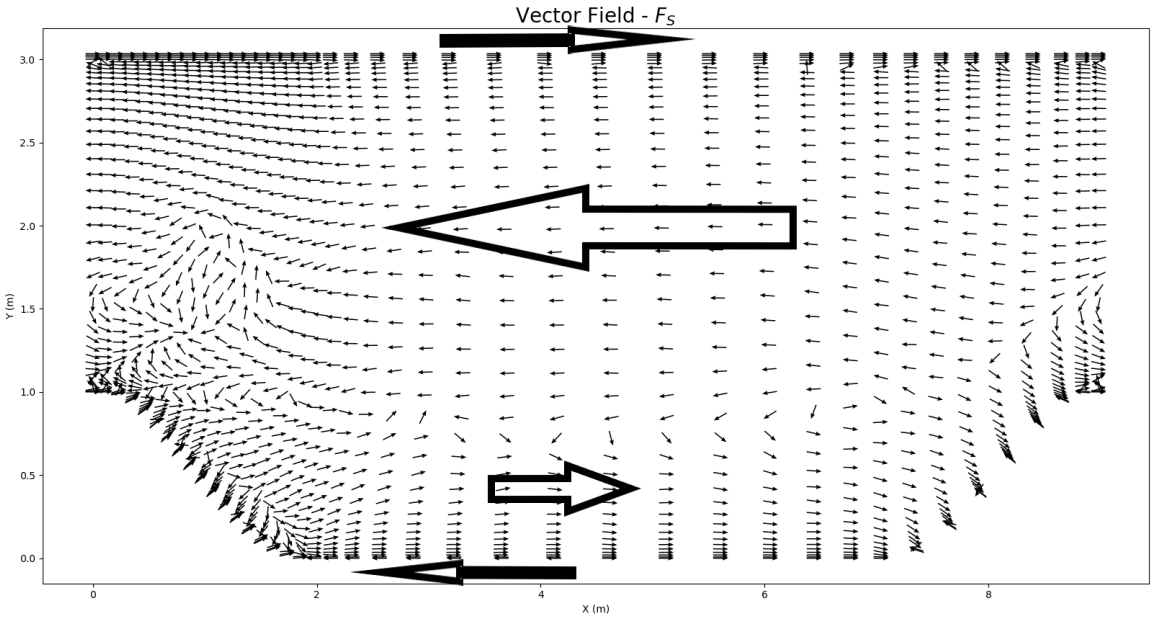
2.4 Assignment 1.4



(a) Upper Wall



(b) Lower Wall



(c) Whole domain

Figure 13: Quiver plot of Fs

From figure 13, the shear force field depict different behaviours across different parts of the domain.

Viscous regions at both the walls:

In the Viscous region (inner region) the Reynolds shear forces act as the driving force for the flow and are opposed by Viscous shear forces. Therefore here the direction of the force will be in the same direction of the flow.

Log law region at both the walls:

In the log law region the Reynolds shear forces act as an opposing force to the flow, here the driving force is the pressure force. Therefore here the direction of the force will be against the direction of the flow.

As seen in the quiver plot for shear force, Near the walls,the direction of the shear force is the same as the flow direction.

Again, the re-circulation region away from the lower wall (log law region) where the flow direction is toward the left , here the Force direction is towards the right. Similarly in the log law region away from the upper wall where the flow is towards the right, here the shear force direction is towards the left.

2.5 Assignment 1.5

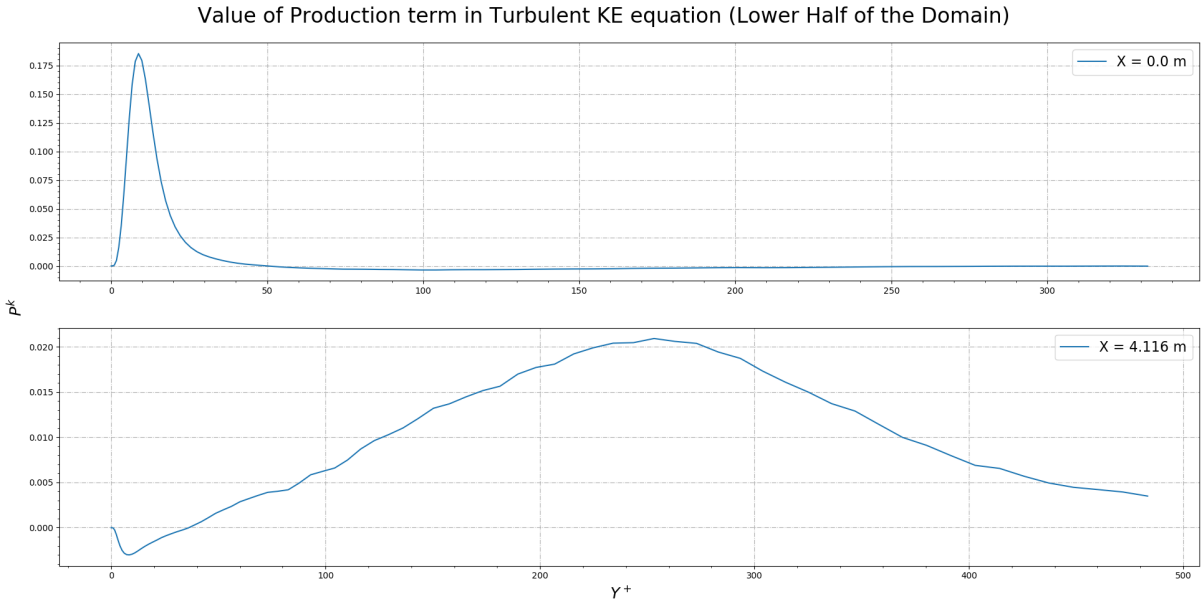


Figure 14: Production term Upper Half of Domain

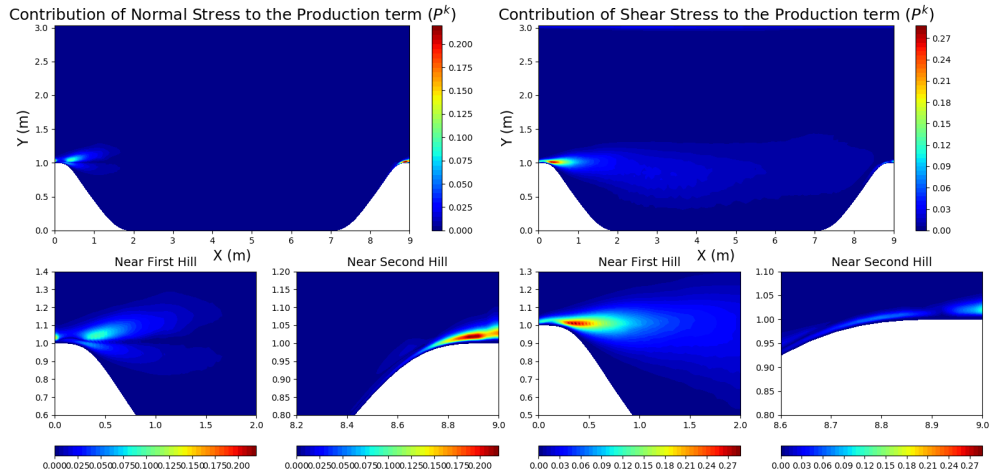
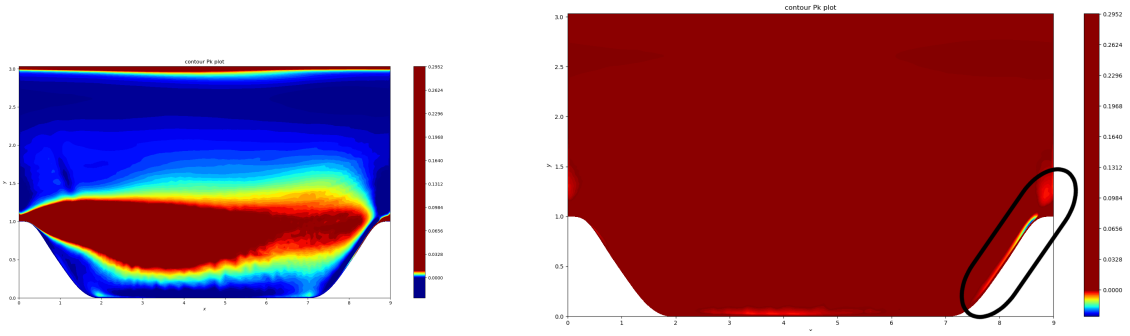


Figure 15: Contribution to Pk term

The value of Pk is highest in regions at high energy containing eddys (small length scales) ,this is present away from the wall and in the log law region. Basically, Pk has the highest values in highly turbulent regions.

From figure 15, regions with high Reynolds stresses are regions with the highest Pk values. Pk is negligible near the walls. In figure 14,The difference in Pk curves at different Vertical lines are due to different Reynolds stress values at these lines.



(a) Small Positive values of Pk

(b) Negative values of Pk

Figure 16: Small Positive and Negative values of Pk

From figure 16, Pk can be visualized as a work term, Pk > 0 in regions where the mean flow performs work on the fluctuating velocity field and Pk < 0 is regions where the fluctuating velocity field performs work on the mean flow.The latter case happens when the gradient of mean flow velocity is in the opposite direction of fluctuation velocity.

2.6 Assignment 1.6

Figure 17 depicts the dissipation in the domain, the dissipation near the slope of the first hill and slope of the second hill. This is close to the beginning of separation region.

Figure 18 shows the difference in Pk term and the Dissipation term. The completely white regions are regions where the Pk term is equal to the dissipation term.

Note: We are not sure of the explanation but a visual inspection depicts the terms cancel out at the transition from inner region to log law region.

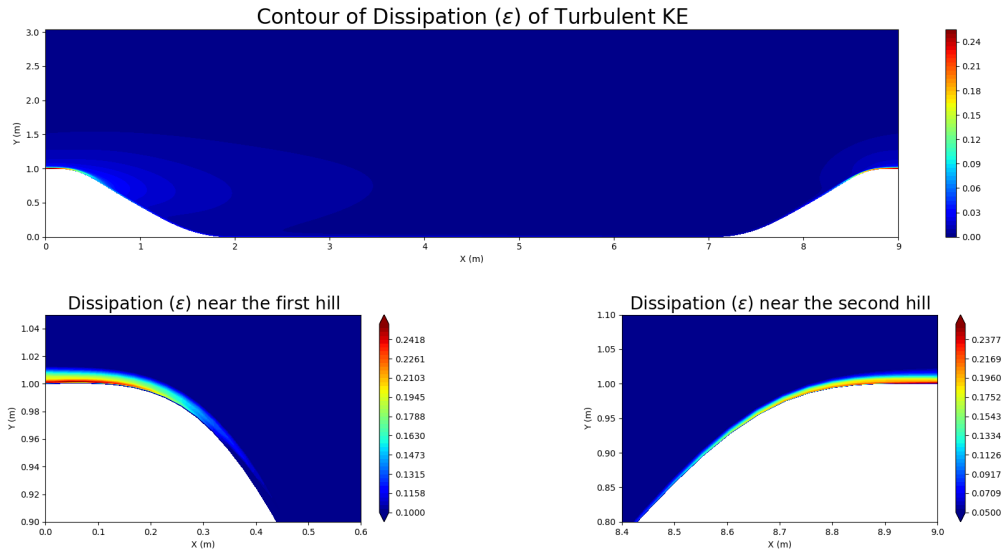


Figure 17: Dissipation

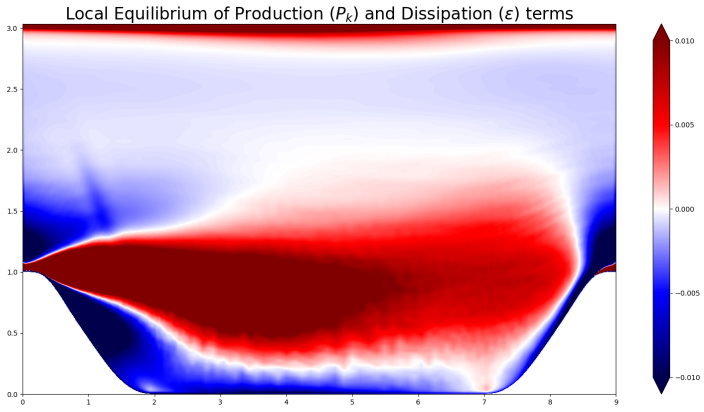


Figure 18: Equilibrium of Pk and Dissipation

2.7 Assignment 1.7

Figure 19 and figure 21 represents the various terms in $\overline{V_1{'}^2}$ equation for the lower and upper domain respectively. Figure 20 and figure 22 represents the various terms in $\overline{V_1{'}V_2{'}}$ equation for the lower and upper domain respectively.

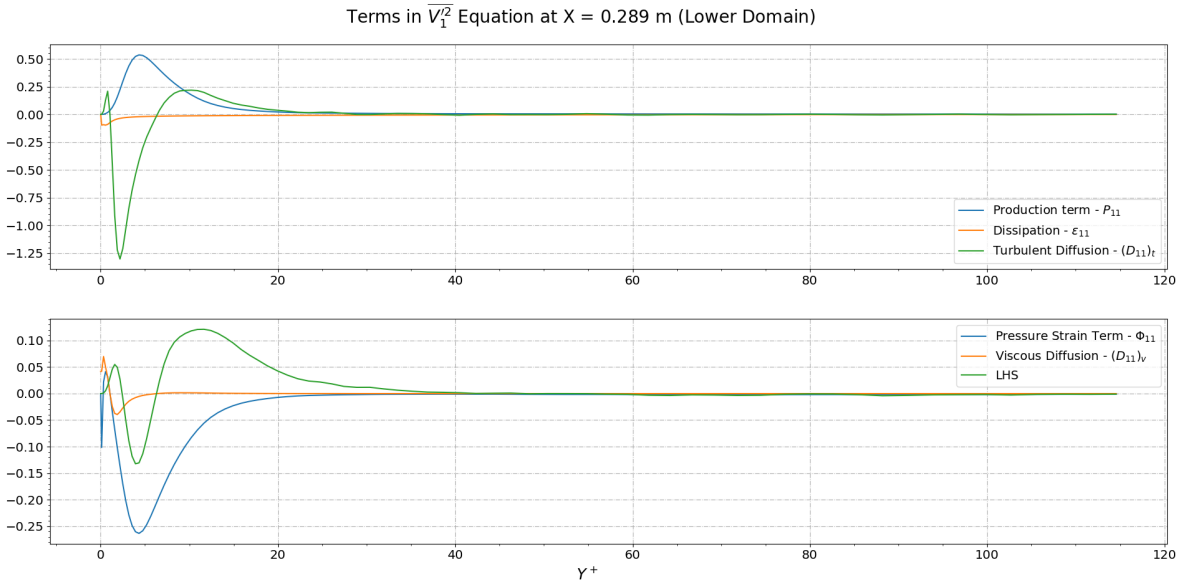


Figure 19: Terms in $\overline{V_1{'}^2}$ equation in lower domain

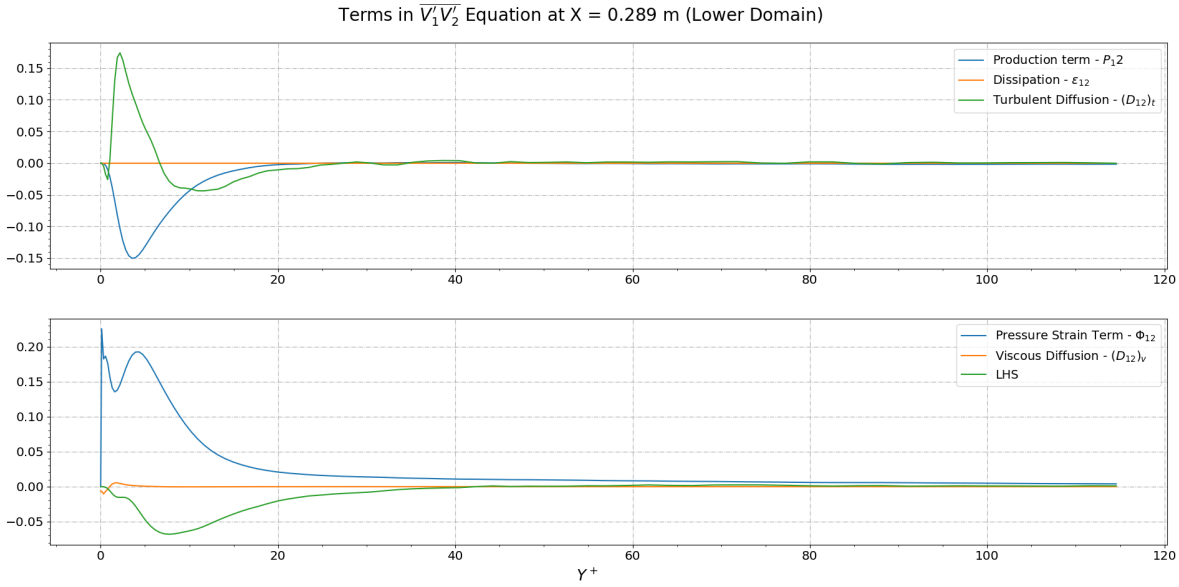


Figure 20: Terms in $\overline{V_1{'}V_2{'}}$ equation in lower domain

- For lower domain in $\overline{V_1{'}^2}$ equation, the production term acts as the main source term, whereas pressure-strain and turbulent diffusion term acts as the sink near the wall. The viscous diffusion

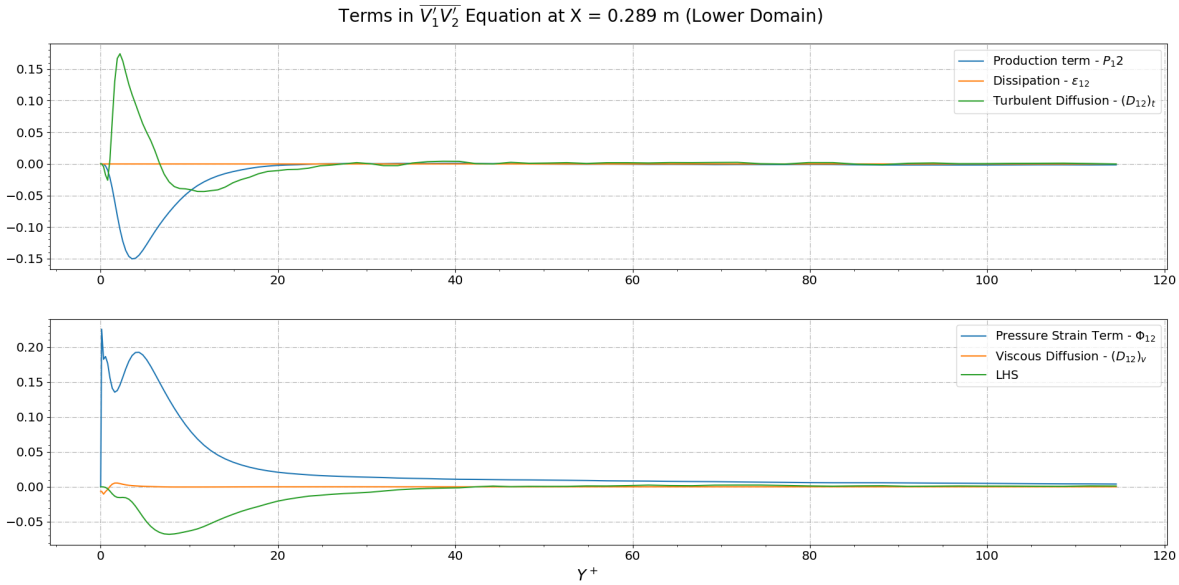


Figure 21: Terms in $\overline{V_1{}'^2}$ equation in upper domain

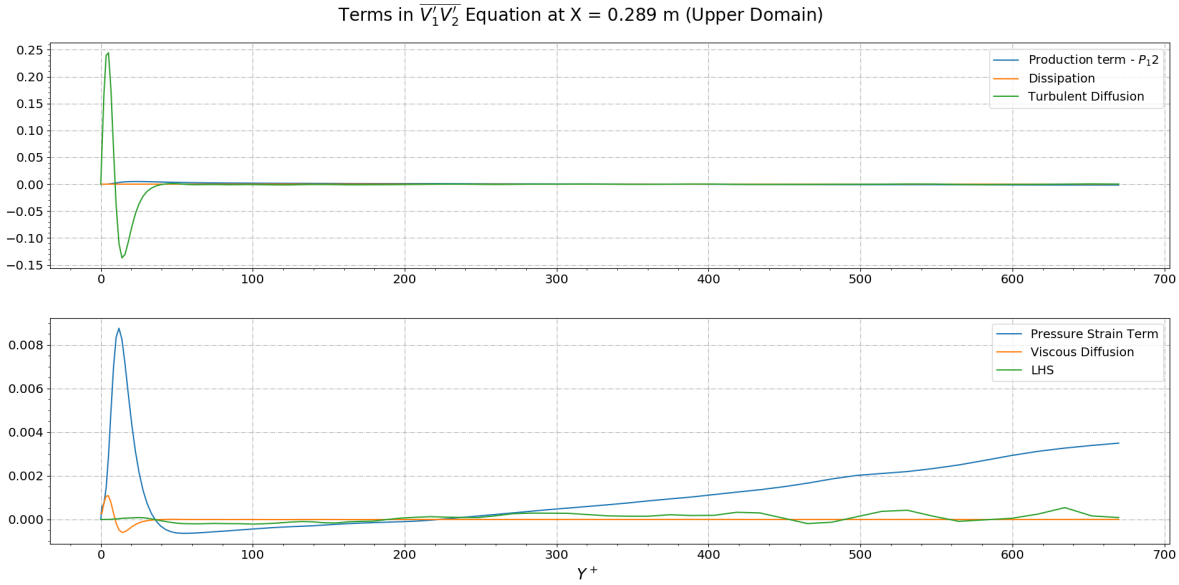


Figure 22: Terms in $\overline{V_1{}'V_2{}'}$ equation in upper domain

term is higher near the wall but it goes to zero as we enter the log-law region. If we zoom in the outer region, production term balances the pressure-strain term and dissipation term.

- For upper domain in in $\overline{V_1{}'^2}$ equation, the turbulent diffusion is very high as compared to other quantities. This is because the term is calculated based on the eddy viscosity, which is very high near the upper wall as Boussinesq assumption over-predicts the turbulence due to its isotropic turbulence assumption. Near the wall, viscous diffusion is high and pressure-strain acts as the source term.
- For lower domain in $\overline{V_1{}'V_2{}'}$ equation, the production term is negative, this implies that the shear stress is negative in this region. The dissipation term is zero. Pressure-strain term and turbulent

diffusion term acts as the sink near the wall. The pressure-strain term is balanced by the production term in outer region.

- For upper domain in $\overline{V_1{}'V_2{}'}$ equation, the pressure strain term increases as we go away from the wall, this means the shear stress is being redistributed to the normal stresses (as pressure-strain term is called Robinhood term) in that region.

2.8 Assignment 1.8

Figure 23 and figure 24 compares the Eddy viscosity stresses with the Reynolds stress at two locations in the domain fro upper as well as lower domain.

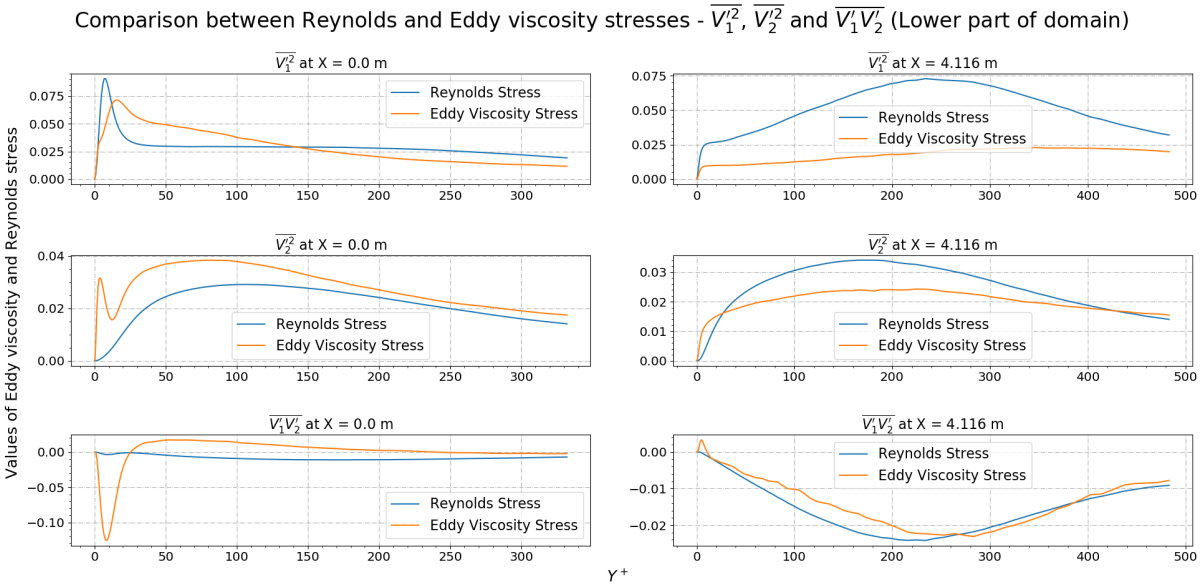


Figure 23: Reynolds and Eddy Viscosity stresses in Lower part of Domain

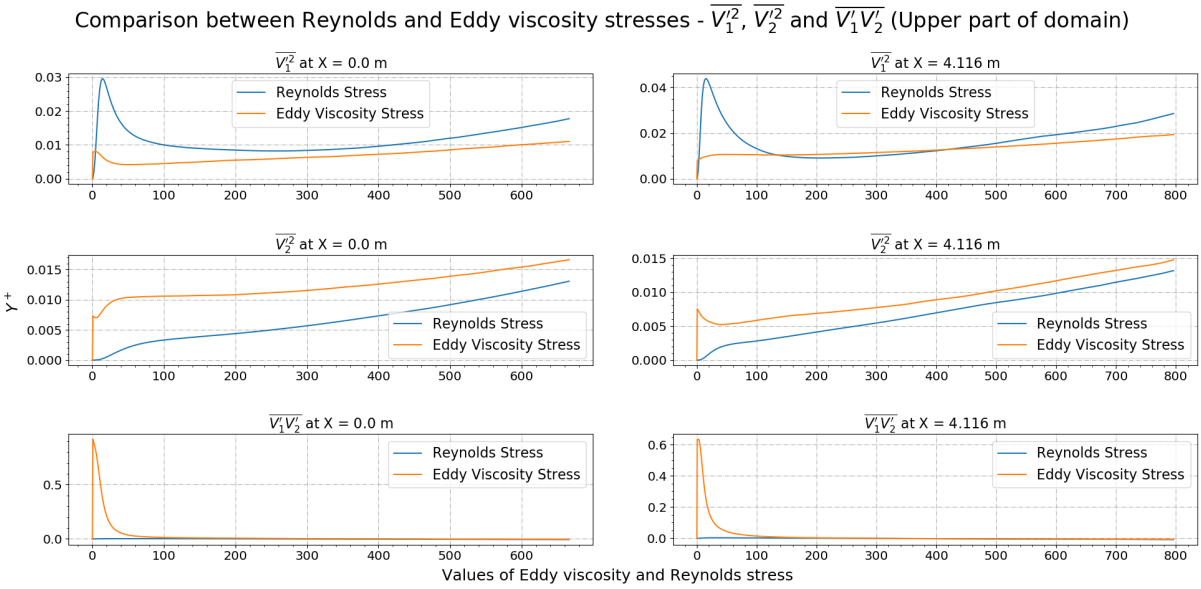


Figure 24: Reynolds and Eddy Viscosity stresses in Upper part of Domain

- In the lower part of the domain (figure 23), the eddy viscosity stresses are larger than the reynolds stress near the wall (in viscous and log-law region). But the reynolds stress in streamwise direction is higher than eddy viscosity stress.

- In the upper part of the domain (figure 24), the wall normal and shear stress calculated by Boussinesq assumption are higher than the Reynolds stress, whereas the Reynolds stress in flow direction is higher than its eddy viscosity counterpart. Thus the assumption of isotropic turbulence tends to over-predict the stresses in eddy viscosity model especially near the wall.
- On the left side of figure 25, the regions where the eddy viscosity stresses are greater than the reynolds stresses can be visualized. Regions with a positive value means that eddy viscosity stress is greater than the reynolds stress.
- The normal stresses are over-predicted along the lower wall in Boussinesq assumption, whereas along the top wall the shear stresses calculated using Boussinesq assumption are larger than the actual ones.
- Due to the higher stresses calculated near the walls than the actual ones in Boussinesq assumption, the turbulent viscosity is also high and that in turn over-predicts the turbulence near walls. Realizibility concept can be used to dampen the near wall turbulence which is explained in subsequent sections.
- In the recirculation and other regions, reynolds stresses are higher than the eddy viscosity stresses. This figure better explains the line graphs given in figure 23 and figure 24.

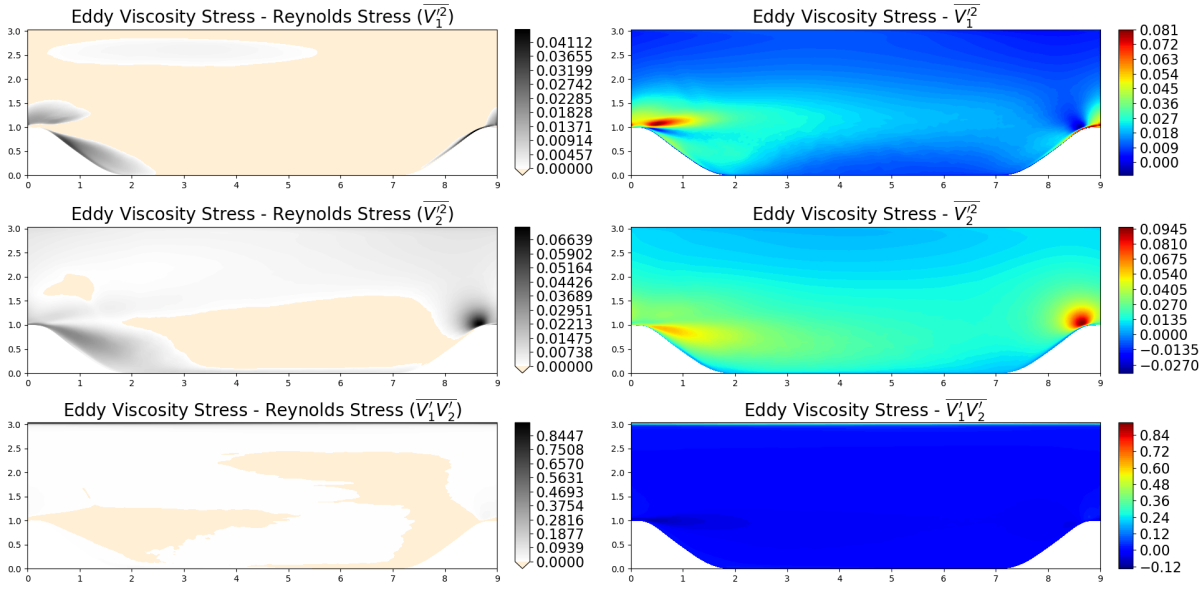


Figure 25: Contour for Reynolds and Eddy Viscosity Stresses

2.9 Assignment 1.9

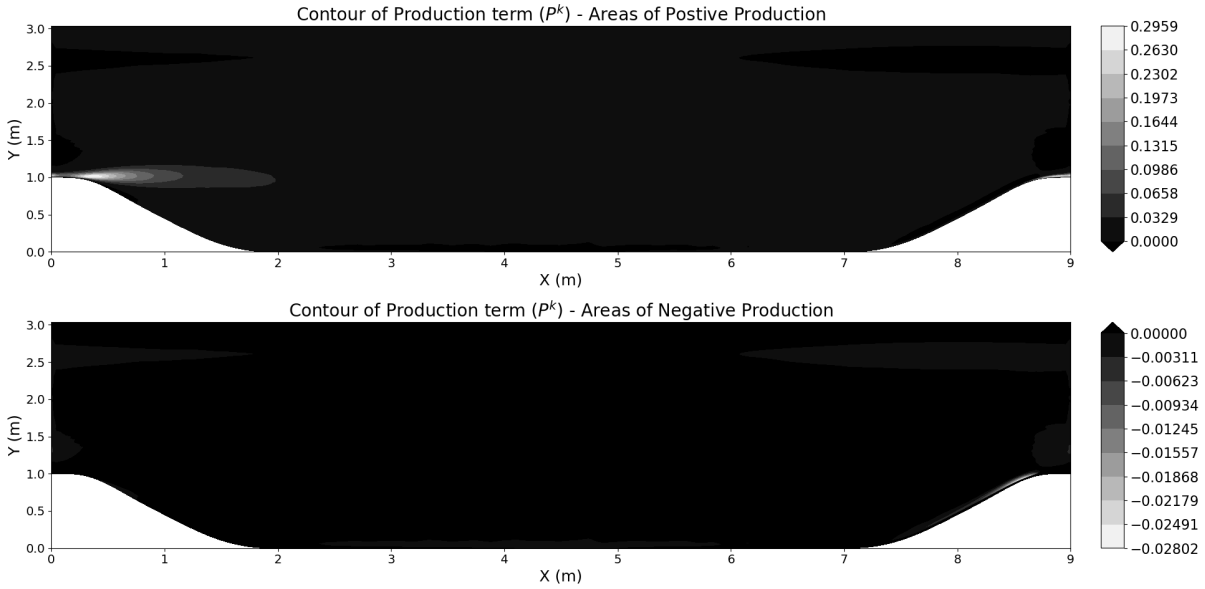


Figure 26: Production term of Turbulent Kinetic Energy - Exact

Figure 26 shows the contour of the exact production term of turbulent kinetic energy. In the top figure, we can see the areas where production term is positive - black means it is negative production. In the bottom figure, we get an idea of how big the negative production terms are.

- It can be seen that the negative production is higher near the second hill, i.e. stagnation region. The production term is the product of an inertial force per unit mass (acceleration) and fluctuating velocity. Hence, the fluctuations are doing work on mean flow field in this region.
- The production term is biggest near separation region, which is likely as turbulence is generated at the hill where flow is separating from the bottom wall.

Figure 27 shows the contour of the production term of turbulent kinetic energy calculated using Boussinesq assumption.

- The turbulence is over-predicted as can be seen from the figure near the top wall. The production value is very much higher than what it is in rest of the domain. This can be because of the isotropic turbulence assumption in Boussinesq hypothesis.
- In rest of the domain, the production is present in separation and stagnation region. One thing is that the production term is positive everywhere, unlike the exact production term shown in figure 26.
- An intresting observation - when we apply the realizability concept, it dampens the turbulence near the top wall, as discussed in section 2.10 below. Due to this, the production term near the top wall also decreases substantially, see figure 28. If we compare figure 27 and 28, near the top wall, using realizability concept gives results closer to the DNS data.

Figure 29 shows that the eigenvectors of modeled Reynolds stress and Strain rate tensor are parallel. This is the reason why the modeled production term in turbulent kinetic energy is always positive.

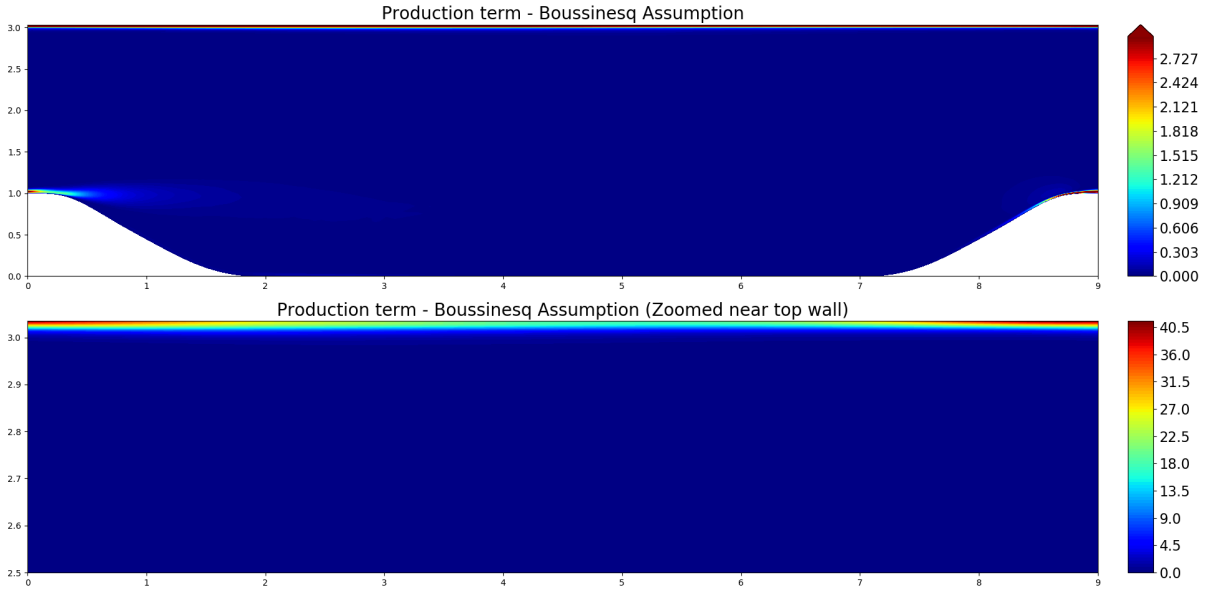


Figure 27: Production term of Turbulent Kinetic Energy - Boussinesq Assumption

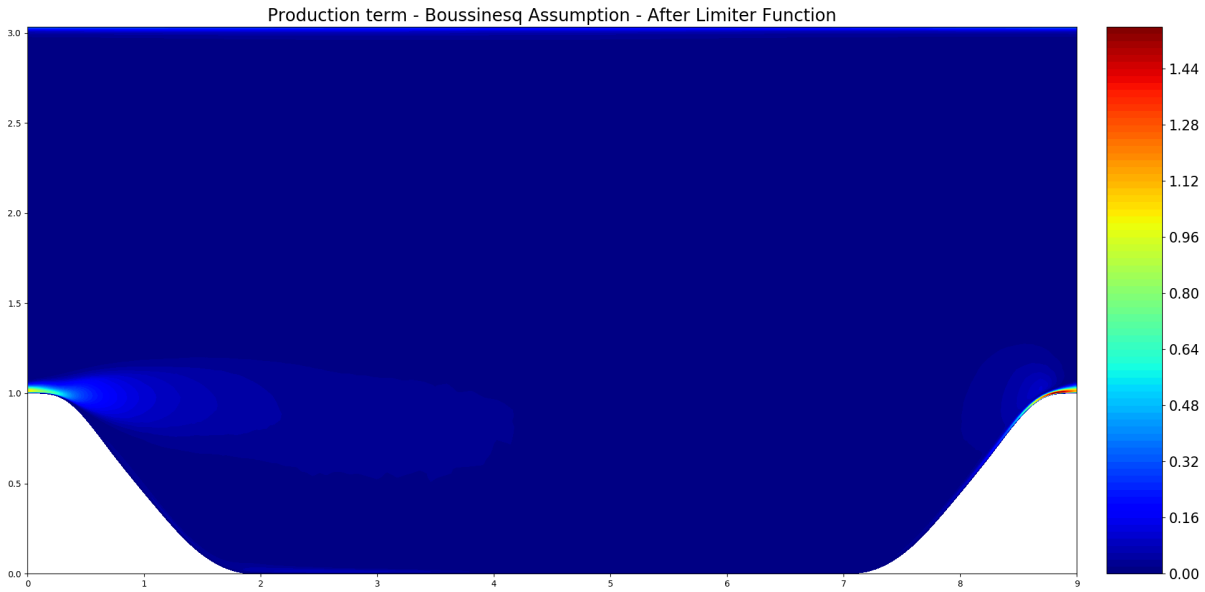


Figure 28: Production term of Turbulent Kinetic Energy - Boussinesq Assumption (After applying the limiter function)

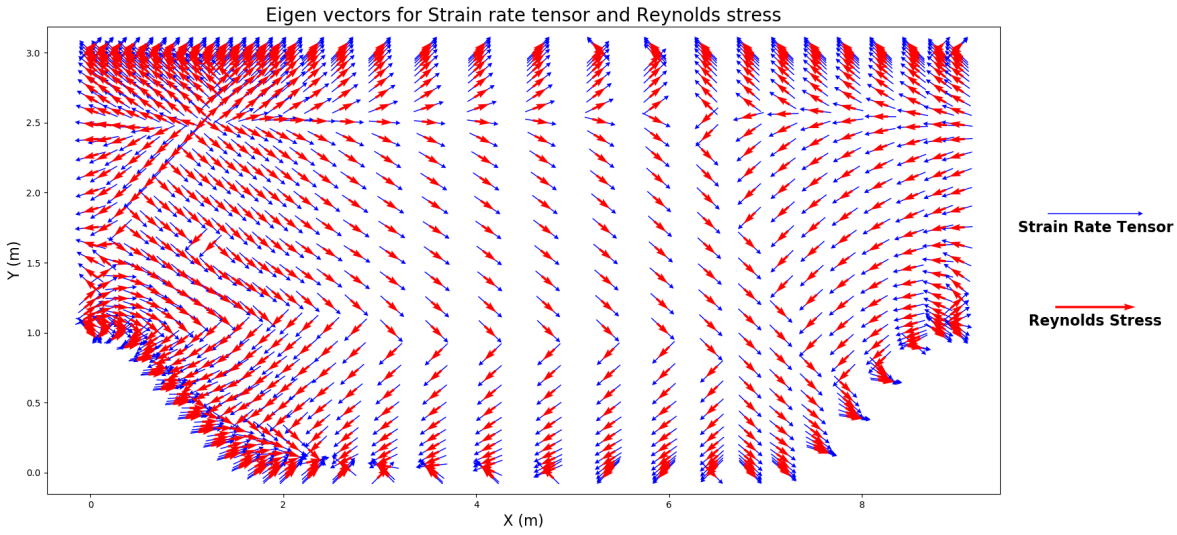
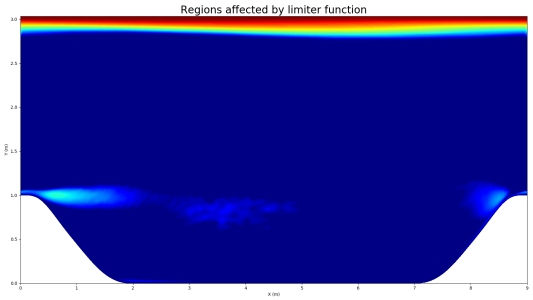
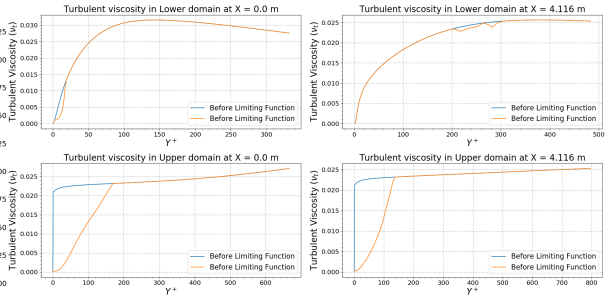


Figure 29: Eigenvectors of Reynolds stress and Strain rate tensor

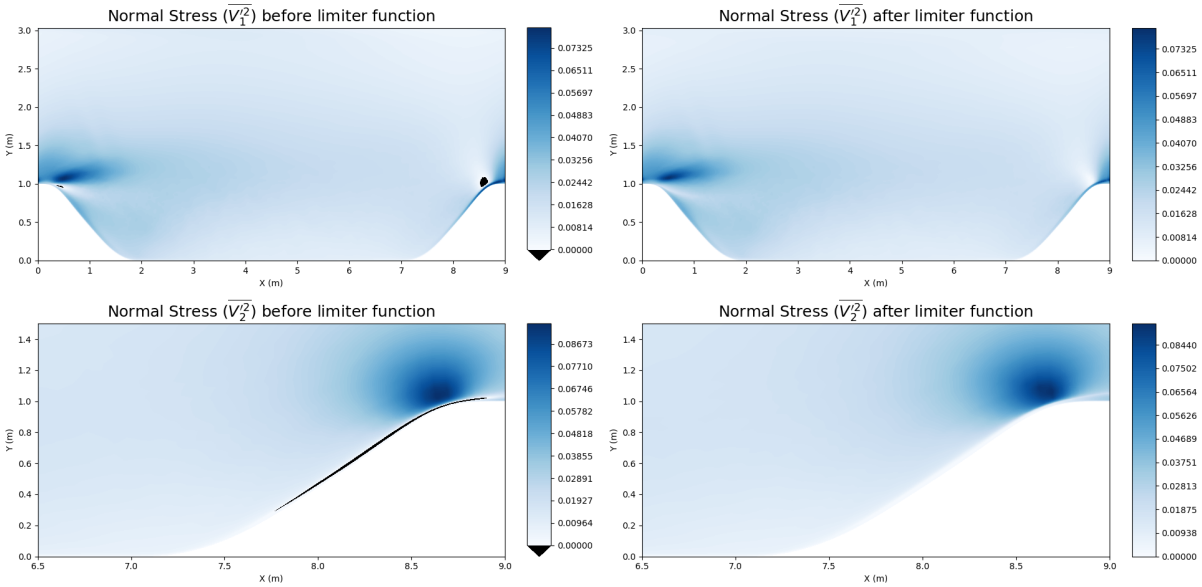
2.10 Assignment 1.10



(a) Contour of Turbulent Viscosity



(b) Turbulent Viscosity at various 'X' locations



(c) Contour of Normal Stresses

Figure 30: Effect of Limiting Function on Normal Stress and Turbulent Viscosity

The realizibilty concept is used to limit the normal stresses in stagnation regions such that they are positive at all times. We limit the turbulent viscosity using the eigenvalues of the strain rate tensor.

- In figure 30a we can observe the regions affected by the limiter function. The contour basically is the difference of original turbulent viscosity $(\nu_t)_O$ and modified turbulent viscosity $(\nu_t)_M$. Hence the blue region (value = 0) represents that the limiter function has no effect on that particular region (i.e. $(\nu_t)_O < (\nu_t)_M$).
- It can be seen that the limiter has the most effect near the top wall. Near the top wall, Boussinesq hypothesis over-predicts the turbulence viscosity due to its assumption of isotropic turbulence. Using the realizibilty concept, we can dampen the turbulent viscosity at the wall.
- Near the hill, where separation (left hill) and stagnation region (right hill) are present, turbulent viscosity is limited by the limiter function. Realizibilty concept dampens the turbulence in these regions as well.
- In figure 30c, the contour of normal stresses ($\overline{V_1'^2}$ and $\overline{V_2'^2}$) are shown. The sub-figures on left side are for normal stresses calculated using boussinesq assumption using the original turbulent viscosity

$(\nu_t)_O$. Whereas the sub-figures on right side denotes the value of normal stresses calculated using the modified turbulent viscosity $(\nu_t)_M$.

- Originally, the normal stresses are negative at some places, predominantly at the stagnation region near the hill as indicated by black color in figure 30c. After using the limiter function, we can see that the normal stresses are not negative anymore, thus we can conclude that the limiter concept in realizibility makes the normal stresses $(\overline{V_i'^2})$ positive.

2.11 Assignment - Backward Facing Step

Various Eddy viscosity as well as Reynolds stress turbulence models were implemented on the given geometry in Star-CCM+. The base mesh size in all the models was constant with addition of prism layers near walls based on the requirement of the turbulence model to resolve or model the boundary layer.

- The Standard $k-\varepsilon$ and Realizable $k-\varepsilon$ model implements wall functions (High y^+ wall treatment) to model the near wall behaviour and needs a y^+ value greater than 30. Hence a base size of 0.1 m was used without any prism layers.
- The Standard and SST $k-\omega$ as well as the RSM Elliptic Blending model uses All y^+ wall treatment and y^+ value is kept below 1 using prism layers to ensure that the viscous sub-layer is properly resolved by the mesh.
- The two layer models for Standard $k-\varepsilon$, Realizable $k-\varepsilon$ and RSM Linear Pressure Term implements a blending function similar to All y^+ wall treatment to model the near wall turbulence. For these models as well, y^+ value is kept below 1.

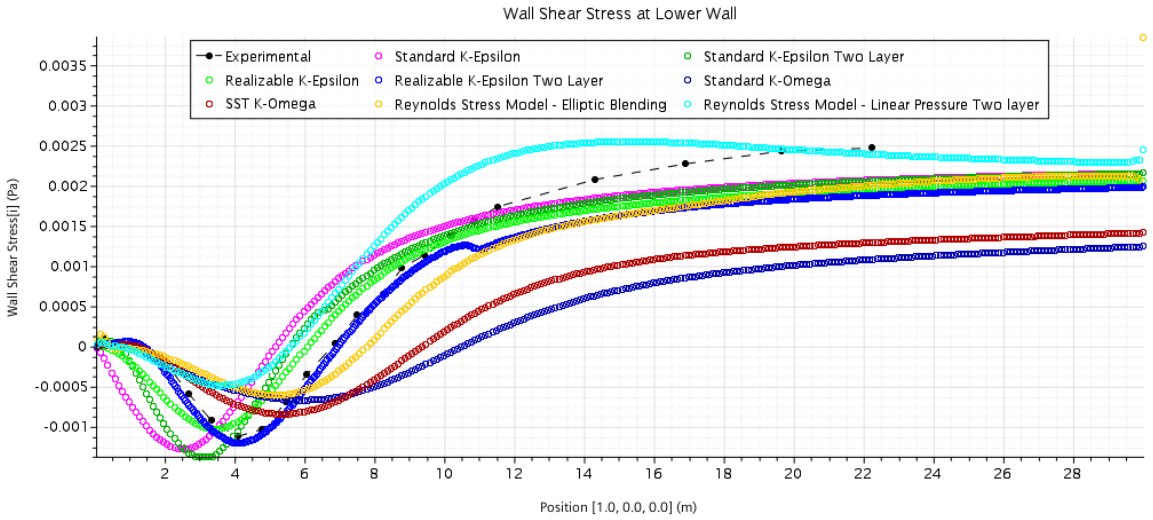


Figure 31: Wall Shear Stress at Lower wall using various Turbulence models

The simulation results for the Wall shear stress and Nusselt number on the lower wall are compared with experimental data as seen in figure 31 and figure 32. From figure 31, we can observe that the realizable $k-\varepsilon$ and realizable $k-\varepsilon$ two layer model is better than other models at predicting the wall shear stress. The $k-\varepsilon$ models perform better than $k-\omega$ models when there is separation due to geometry changes, which can be seen from the results as well. Both Standard and SST $k-\omega$ models are giving an erraneous result. To conclude this study, realizable $k-\varepsilon$ model is better in predicting the wall shear stress and nusselt number for a backward facing step. Based on the computational power available, one can choose between the two layer or high y^+ wall treatment.

Figure 33 and figure 34 are the contours for velocity and turbulent kinetic energy obtained for Realizable $k-\varepsilon$ model. The re-circulation region can be seen near the step. In the shear layer, which starts from the step, the turbulent kinetic energy increases as there is formation of vortices. It is maximum in the region of primary vortex and where the flow is about to reattach the lower wall.

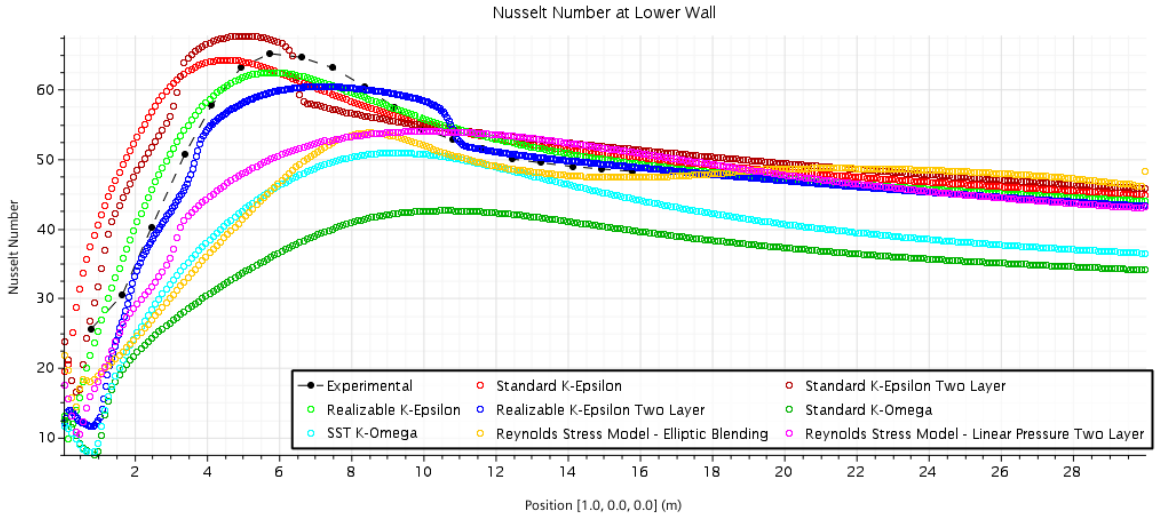


Figure 32: Nusselt Number at Lower wall using various Turbulence models

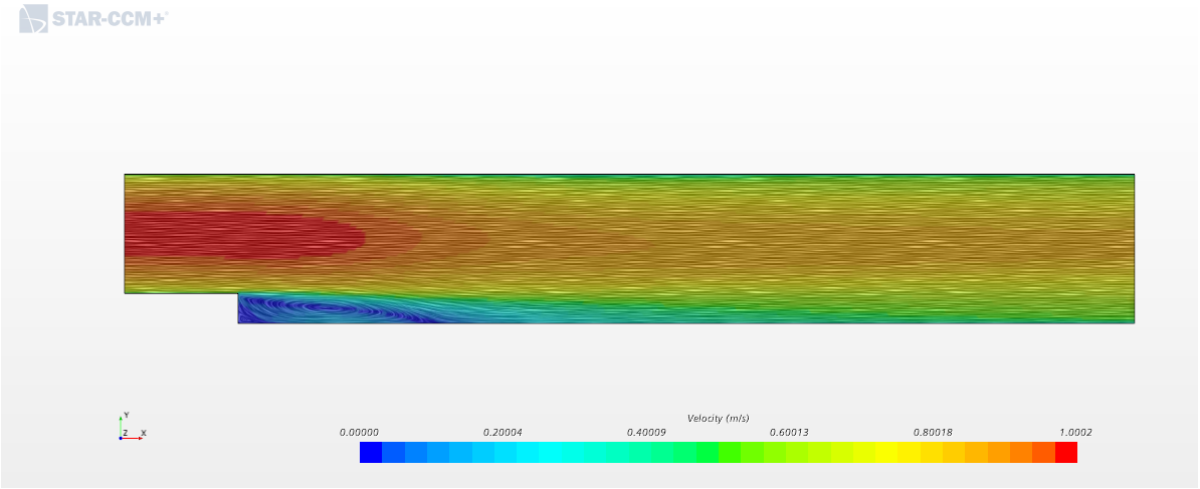


Figure 33: Velocity LIC for Realizable $k - \varepsilon$ model

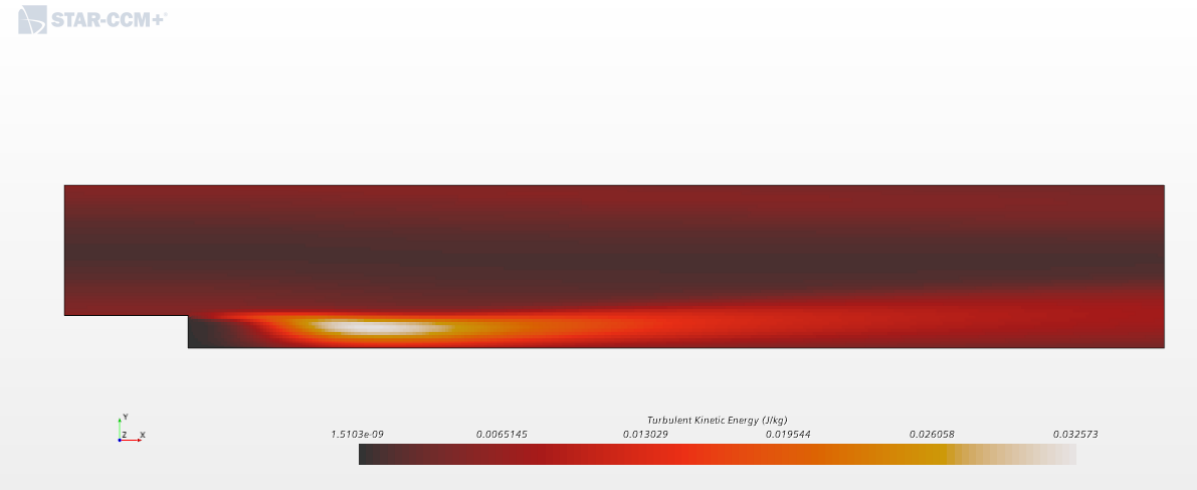


Figure 34: Turbulent Kinetic Energy for Realizable $k - \varepsilon$ model

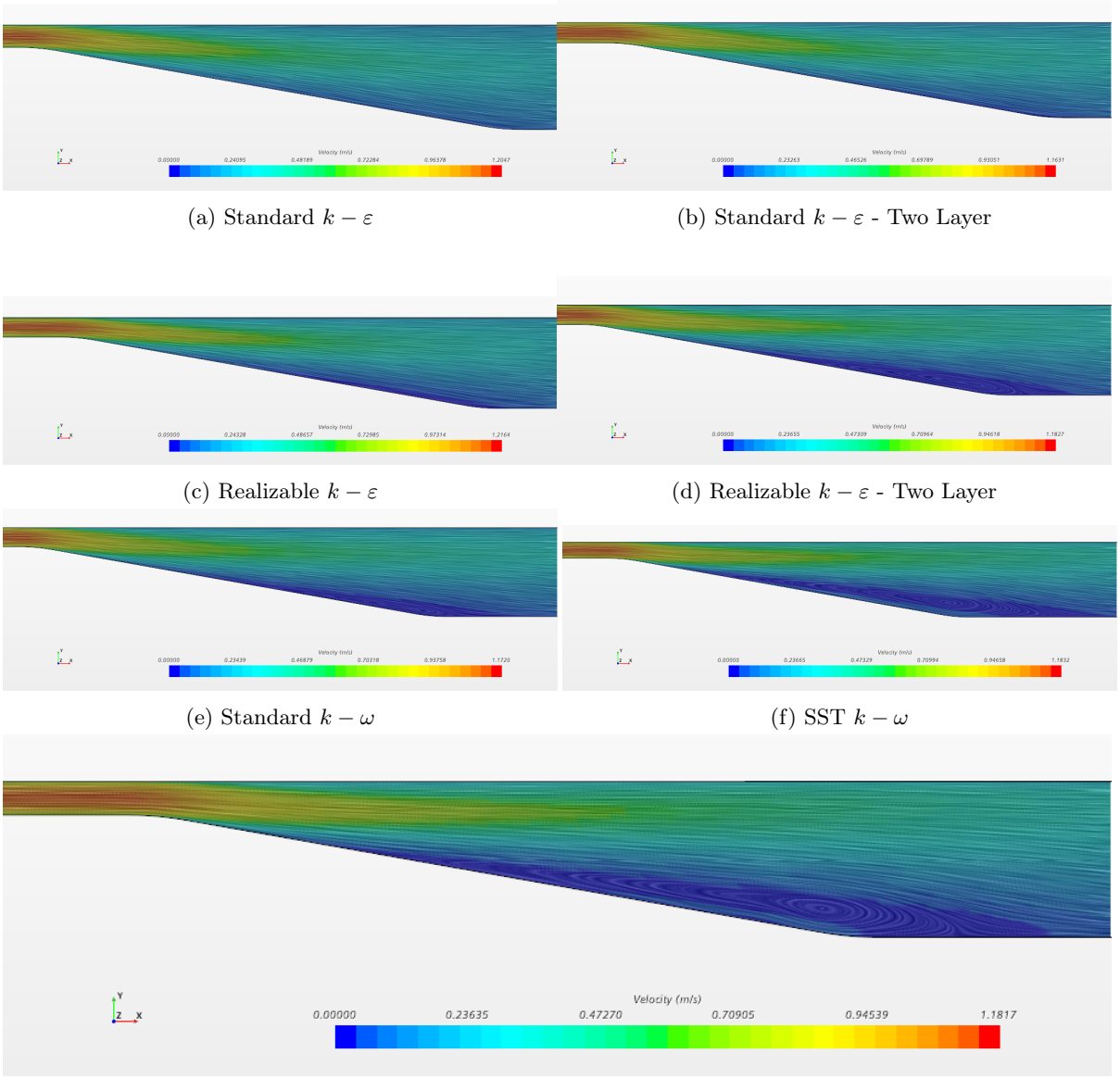
2.12 Assignment - Asymmetric Diffuser (Turbulence Models)

The turbulence models that were used for Backward facing step are used for this case as well. The base mesh size is kept same in all the cases with a finer mesh near walls to achieve the desired y^+ value as per selected turbulence model.

Figure 35 shows the velocity LIC's for all the turbulence models. A diffuser is characterised by flow separation and re-circulation in the diverging section due to adverse pressure gradient. The Standard $k-\varepsilon$ and Standard $k-\varepsilon$ (two layer) model does not predict the re-circulation region and the flow remains attached throughout, as seen in figure 35a and 35b. Realizable $k-\varepsilon$ is also bad at predicting the seperation, but slightly better than Standard $k-\varepsilon$ models, see figure 35c and 35d. The reason behind poor performance of $k-\varepsilon$ models can be due to the fact that they assume isotropic turbulence near the wall. These models basically use wall functions to model the boundary layer in viscous and buffer region. As the boundary layer is not properly resolved, it cannot predict separation arising due to adverse pressure gradient.

Realizable $k-\varepsilon$ (two layer), Standard $k-\omega$, SST $k-\omega$ model and Reynolds stress models are able to model the flow separation and the re-circulation zone. Realizable $k-\varepsilon$ (two layer) and SST $k-\omega$ model are nearly similar when it comes to predicting the flow separation, but the near-wall boundary layer is better resolved in $k-\omega$ model as seen from the comparison with experimental data in figure 36.

Figure 36a - Figure 36j compares the velocity profile at various locations in the diffuser for different turbulence models. Upstream of the diffuser, the velocity profile is nearly the same for all turbulence models as seen from figure 36a. Downstream of the diffuser, the velocity profiles vary drastically as the re-circulation region changes depending on the chosen turbulence model for simulation. The $k-\omega$ (especially, SST variant) and Reynolds stress models agree the most with experiments at nearly all the locations as shown in figure 36. To conclude, we could say that SST $k-\omega$ and Reynolds stress - Elliptic blending model performs well for a asymmetric diffuser geometry. But the Reynolds stress model are computationally heavy and require more iterations to reach a steady state, hence SST $k-\omega$ is preferred on=ver it. Whereas, the $k-\varepsilon$ models are bad at prediction flow separation due to adverse pressure gradient.



(a) Standard $k-\varepsilon$

(b) Standard $k-\varepsilon$ - Two Layer

(c) Realizable $k-\varepsilon$

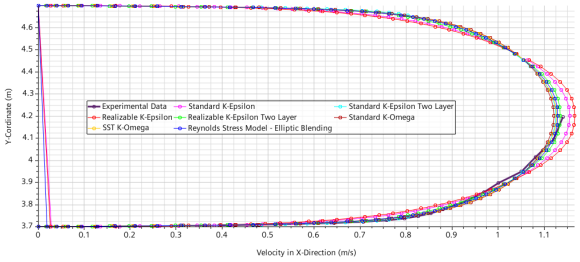
(d) Realizable $k-\varepsilon$ - Two Layer

(e) Standard $k-\omega$

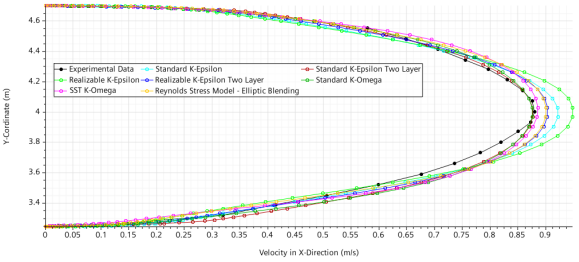
(f) SST $k-\omega$

(g) Reynolds Stress Model - Elliptic Blending

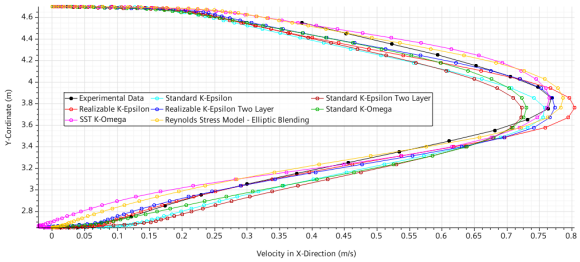
Figure 35: LIC of velocity for various turbulence models



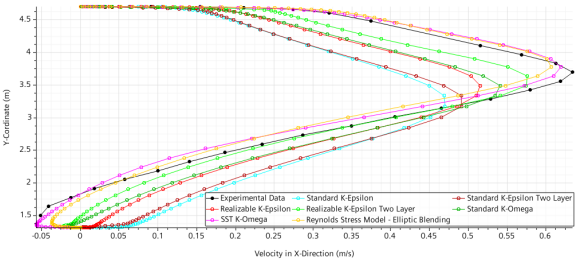
(a) Velocity at X = -5.87 m



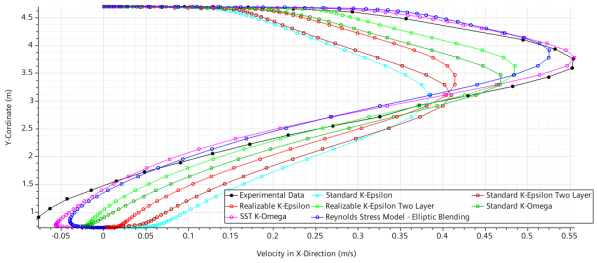
(b) Velocity at X = 2.59 m



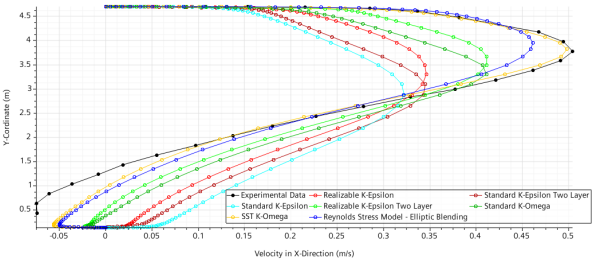
(c) Velocity at X = 5.98 m



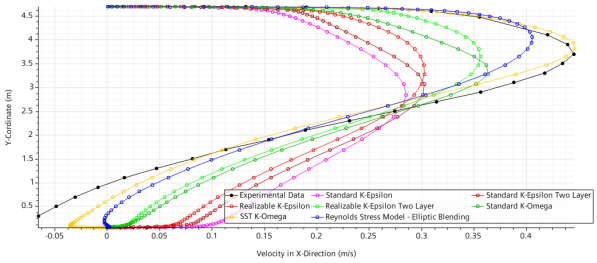
(d) Velocity at X = 13.56 m



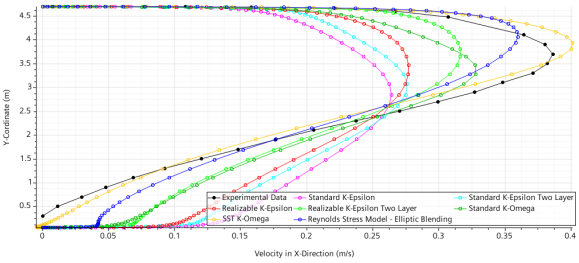
(e) Velocity at X = 16.93 m



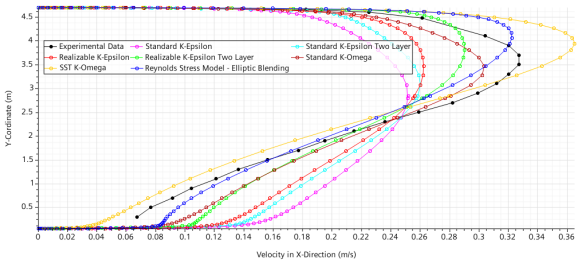
(f) Velocity at X = 20.32 m



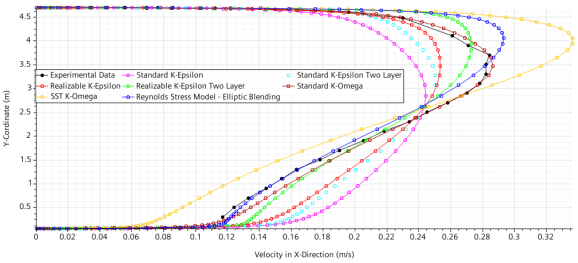
(g) Velocity at X = 23.71 m



(h) Velocity at X = 27.09 m



(i) Velocity at X = 30.48 m



(j) Velocity at X = 33.87 m

Figure 36: Comparison of Turbulence Models

2.13 Assignment - Asymmetric Diffuser (Pressure Recovery)

The pressure recovery and pressure drop is compared for various turbulence models for the asymmetric diffuser case. Pressure drop is used as a convergence criteria for the simulations. Pressure recovery is an important parameter to evaluate the diffuser performance. The diffuser should convert maximum possible kinetic energy to pressure by decelerating the fluid as a consequence of increase in area. Thus pressure recovery gives a measure of how much static pressure is regained.

$$\text{Pressure Recovery} = \text{Static Pressure @ Outlet - Static Pressure @ Inlet} \tag{1}$$

$$\text{Pressure Drop} = \text{Total Pressure @ Inlet - Total Pressure @ Outlet} \tag{2}$$

Table 1: Pressure Recovery and Pressure Drop for various Turbulence models

Turbulence Models	**Pressure Recovery (Pa)**	**Pressure Drop (Pa)**
Standard K-ε	-0.2150548	0.7785385
Standard K-ε Two Layer	0.0638940	0.4994603
Realizable K-ε	-0.2030131	0.7660116
Realizable K-ε Two Layer	0.0588291	0.5040755
Standard K-ω	0.0608963	0.5009818
SST K-ω	0.0487484	0.5104201
Reynolds Stress Model - Elliptic Blending	0.0263190	0.5369382

The Standard and Realizable $k-\varepsilon$ model is a poor choice of turbulence model for this type of flow as we saw in earlier section. Additionally, it gives an erroneous value of pressure recovery as well. The reason could be since the y^+ values are less than 1, but these models use a high y^+ near wall treatment which requires for $y^+ > 30$. Due to this bad meshing approach, they are far away from predicting the correct flow behaviour even when compared to its two layer model counterparts.

A Appendix

```
import scipy.io as sio
import numpy as np
import matplotlib.pyplot as plt
from dphidx_dy import dphidx_dy
from WallDistance import WallDistance
from WallNormal import WallNormal
from IPython import display
from matplotlib.ticker import (MultipleLocator, FormatStrFormatter,
                               AutoMinorLocator)
#plt.rcParams.update({'font.size': 20})

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

# read data file
tec=np.genfromtxt("tec.dat", dtype=None,comments="%")

#text='VARIABLES = X Y P U V u2 v2 w2 uv mu_sgs prod'

x=tec[:,0]
y=tec[:,1]
p=tec[:,2]
u=tec[:,3]
v=tec[:,4]
uu=tec[:,5]
vv=tec[:,6]
ww=tec[:,7]
uv=tec[:,8]
k=0.5*(uu+vv+ww)

if max(y) == 1.:
   ni=170
   nj=194
   nu=1./10000.
else:
   nu=1./10595.
   if max(x) > 8.:
     nj=162
     ni=162
   else:
     ni=402
     nj=162

viscos=nu

u2d=np.reshape(u,(nj,ni))
v2d=np.reshape(v,(nj,ni))
p2d=np.reshape(p,(nj,ni))
x2d=np.reshape(x,(nj,ni))
y2d=np.reshape(y,(nj,ni))
uu2d=np.reshape(uu,(nj,ni)) #=mean{v'_1v'_1}
uv2d=np.reshape(uv,(nj,ni)) #=mean{v'_1v'_2}
vv2d=np.reshape(vv,(nj,ni)) #=mean{v'_2v'_2}
k2d=np.reshape(k,(nj,ni))

u2d=np.transpose(u2d)
v2d=np.transpose(v2d)
p2d=np.transpose(p2d)
x2d=np.transpose(x2d)
y2d=np.transpose(y2d)
uu2d=np.transpose(uu2d)
vv2d=np.transpose(vv2d)
uv2d=np.transpose(uv2d)
```

```
k2d=np.transpose(k2d)

# set periodic b.c on west boundary
#u2d[0,:]=u2d[-1,:]
#v2d[0,:]=v2d[-1,:]
#p2d[0,:]=p2d[-1,:]
#uu2d[0,:]=uu2d[-1,:]

# read k and eps from a 2D RANS simulations. They should be used for computing the damping function f
k_eps_RANS = np.loadtxt("k_eps_RANS.dat")
k_RANS=k_eps_RANS[:,0]
diss_RANS=k_eps_RANS[:,1]
vist_RANS=k_eps_RANS[:,2]

ntstep=k_RANS[0]

k_RANS_2d=np.reshape(k_RANS,(ni,nj))/ntstep
diss_RANS_2d=np.reshape(diss_RANS,(ni,nj))/ntstep
vist_RANS_2d=np.reshape(vist_RANS,(ni,nj))/ntstep

# set small values on k & eps at upper and lower boundaries to prevent NaN on division
diss_RANS_2d[:,0]= 1e-10
k_RANS_2d[:,0]= 1e-10
vist_RANS_2d[:,0]= nu
diss_RANS_2d[:,-1]= 1e-10
k_RANS_2d[:,-1]= 1e-10
vist_RANS_2d[:,-1]= nu

# set Neumann of p at upper and lower boundaries
p2d[:,1]=p2d[:,2]
p2d[:,-1]=p2d[:,-1-1]

# x and y are fo the cell centers. The dphidx_dy routine needs the face coordinate, xf2d, yf2d
# load them
xc_yc = np.loadtxt("xc_yc.dat")
xf=xc_yc[:,0]
yf=xc_yc[:,1]
xf2d=np.reshape(xf,(nj,ni))
yf2d=np.reshape(yf,(nj,ni))
xf2d=np.transpose(xf2d)
yf2d=np.transpose(yf2d)

# delete last row
xf2d = np.delete(xf2d, -1, 0)
yf2d = np.delete(yf2d, -1, 0)
# delete last columns
xf2d = np.delete(xf2d, -1, 1)
yf2d = np.delete(yf2d, -1, 1)

# compute the gradient dudx, dudy at point P
dudx= np.zeros((ni,nj))
dudy= np.zeros((ni,nj))
dvdx= np.zeros((ni,nj))
dvdy= np.zeros((ni,nj))

dudx,dudy=dphidx_dy(xf2d,yf2d,u2d)
dvdx,dvdy=dphidx_dy(xf2d,yf2d,v2d)
'''
################################ vector plot
fig1 = plt.figure("Figure 1")
k=2# plot every forth vector
ss=3.2 #vector length
plt.quiver(x2d[::k,::k],y2d[::k,::k],u2d[::k,::k],v2d[::k,::k])
plt.xlabel("$x$")
```

```
plt.ylabel("$y$")
plt.title("vector plot")

################################ contour plot
fig2 = plt.figure("Figure 2")
plt.contourf(x2d,y2d,u2d,100,cmap='jet')
plt.xlabel("$X$ (m)",fontsize=20)
plt.ylabel("$Y$ (m)",fontsize=20)
plt.title("Velocity in X-direction ($\overline{u}$) - DNS Data",fontsize=24)
cb = plt.colorbar()
cb.ax.tick_params(labelsize=18)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)

fig3 = plt.figure("Figure 3")
plt.contourf(x2d,y2d,v2d,100,cmap='jet')
plt.xlabel("$X$ (m)",fontsize=20)
plt.ylabel("$Y$ (m)",fontsize=20)
plt.title("Velocity in Y-direction ($\overline{v}$) - DNS Data",fontsize=24)
cb = plt.colorbar()
cb.ax.tick_params(labelsize=18)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)

fig4 = plt.figure("Figure 4")
plt.contourf(x2d,y2d,p2d,100,cmap='jet')
plt.xlabel("$X$ (m)",fontsize=20)
plt.ylabel("$Y$ (m)",fontsize=20)
plt.title("Pressure ($\overline{P}$) - DNS Data",fontsize=24)
cb = plt.colorbar()
cb.ax.tick_params(labelsize=18)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)

################################ contour plot
fig5 = plt.figure("Figure 5")
plt.contourf(x2d,y2d,k_RANS_2d,100,cmap='jet')
plt.xlabel("$X$ (m)",fontsize=20)
plt.ylabel("$Y$ (m)",fontsize=20)
plt.title("Turbulent Kinetic Energy - RANS Simulation",fontsize=24)
cb = plt.colorbar()
cb.ax.tick_params(labelsize=18)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)

#************
# plot uv
fig6 = plt.figure("Figure 6")
i=10
plt.plot(uv2d[i,:],y2d[i,:],'b-')
plt.xlabel('$\overline{u^\prime v^\prime}$')
plt.ylabel('y/H')
'''
#%%
# Plot Mesh for Visualization
a = np.transpose(xf2d)
b = np.transpose(yf2d)
fig1 = plt.figure()
axes = fig1.add_axes([0.1,0.1,0.8,0.8])
axes.plot(xf2d,yf2d,color='black')
axes.plot(a,b,color='black')
#axes.plot(x2d[:,81],y2d[:,81],color='red',lw=2)
axes.tick_params(axis="x", labelsize=15)
axes.tick_params(axis="y", labelsize=15)
axes.set_xlabel('x [m]',fontsize=20)
axes.set_ylabel('y [m]',fontsize=20)
```

```
axes.set_title('Computational mesh in the domain',fontsize=24)

# Finding Y+ value in the domain
#yplus[:,0-81] - lowerhalf of the domain
#yplus[:,82-162] - upper half of the domain.

# Diffusion term in momentum equation
dub_dx,dub_dy = dphidx_dy(xf2d,yf2d,u2d)
dvb_dx,dvb_dy = dphidx_dy(xf2d,yf2d,v2d)

tau_wall = np.zeros((ni,nj))
yplus = np.zeros((ni,nj))
for i in range(ni):
    for j in range(1,nj-1):
        tau_wall[i,j] = nu * np.abs(dub_dy[i,0])
        if j<=81:
            yplus[i,j] = (y2d[i,j]-y2d[i,0]) * np.sqrt(np.abs(dub_dy[i,0]/nu))
        else:
            yplus[i,j] = (y2d[i,-1]-y2d[i,j]) * np.sqrt(np.abs(dub_dy[i,-1]/nu))

#%% Assignment 1.1 - Plot the stresses along vertical grid lines at high and low
    turbulence regions
# Plot uu2d,vv2d, uv2d at two locations in the domain
gl1 = 0
gl2 = 100

tau11 = (nu * dub_dx) - uu2d
tau12 = (nu * dub_dy) - uv2d
tau22 = (nu * dvb_dy) - vv2d
tau21 = (nu * dvb_dx) - uv2d

# Near bottom wall
fig2 = plt.figure()
ax1 = plt.subplot2grid((2,2), (0,0), colspan=2)
ax2 = plt.subplot2grid((2,2), (1,0))
ax3 = plt.subplot2grid((2,2), (1,1))
ax1.contourf(x2d,y2d,v2d,cmap='jet',levels=100,alpha=0.5)
ax1.plot(x2d[gl1,:],y2d[gl1,:],color='black')
ax1.plot(x2d[gl2,:],y2d[gl2,:],color='red')
ax1.plot(x2d[:,81],y2d[:,81],color='green',lw=2)
ax1.set_title('Location of Gridlines for plotting',fontsize=20)
ax1.text(2, 2, 'Upper half of domain', fontsize=20, color="black")
ax1.text(2, 1, 'Lower half of domain', fontsize=20, color="black")
ax2.plot(tau11[gl1,0:81],yplus[gl1,0:81])
ax2.plot(tau22[gl1,0:81],yplus[gl1,0:81])
ax2.plot(tau12[gl1,0:81],yplus[gl1,0:81])
ax2.plot(tau21[gl1,0:81],yplus[gl1,0:81])
ax2.set_title('Stresses at X = '+str(np.around(x2d[gl1,0],3))+' m',fontsize=20)
ax3.plot(tau11[gl2,0:81],yplus[gl2,0:81])
ax3.plot(tau22[gl2,0:81],yplus[gl2,0:81])
ax3.plot(tau12[gl2,0:81],yplus[gl2,0:81])
ax3.plot(tau21[gl2,0:81],yplus[gl2,0:81])
ax3.set_title('Stresses at X = '+str(np.around(x2d[gl2,0],3))+' m',fontsize=20)
#ax3.set_ylim(0,100)
for ax in [ax1,ax2,ax3]:
    ax.tick_params(labelsize='medium')
    ax.xaxis.set_minor_locator(AutoMinorLocator())
    ax.yaxis.set_minor_locator(AutoMinorLocator())
    if ax == ax1:
        ax.legend(['x = '+str(np.around(x2d[gl1,0],3))+' m','x = '+str(np.around(x2d[gl2
    ,0],3))+' m'],loc='best')
        ax.set_xlabel('x (m)')
        ax.set_ylabel('y (m)')
    else:
        ax.legend([r'Normal Stress = $\mu\frac{\partial{\overline{u}}}{\partial{x}}$ - $
    \rho\overline{V_1^{\prime2}}$',
```

```
                    r'Normal Stress = $\mu\frac{\partial{\overline{v}}}{\partial{y}}$ - $\rho\overline{V_2^{\prime2}}$',
                    r'Shear Stress = $\mu\frac{\partial{\overline{u}}}{\partial{y}}$ - $\rho\overline{V_1^{\prime}V_2^{\prime}}$',
                    r'Shear Stress = $\mu\frac{\partial{\overline{v}}}{\partial{x}}$ - $\rho\overline{V_1^{\prime}V_2^{\prime}}$'],loc='best')
        ax.grid(True, linestyle='-.')
        ax.set_xlabel('Magnitude of Stress')
        ax.set_ylabel(r'$Y^+$')
fig2.suptitle('Stresses along Vertical Gridlines in Lower half of the Domain', fontsize=24)
'''
fig2a = plt.figure()
ax1 = plt.subplot2grid((2,2), (0,0), colspan=2)
ax2 = plt.subplot2grid((2,2), (1,0))
ax3 = plt.subplot2grid((2,2), (1,1))
ax1.contourf(x2d,y2d,v2d,cmap='jet',levels=50,alpha=0.5)
ax1.plot(x2d[gl1,:],y2d[gl1,:],color='black')
ax1.plot(x2d[gl2,:],y2d[gl2,:],color='red')
ax1.plot(x2d[:,81],y2d[:,81],color='green',lw=2)
ax1.set_title('Location of Gridlines for plotting',fontsize=20)
ax1.text(2, 2, 'Upper half of domain', fontsize=20, color="black")
ax1.text(2, 1, 'Lower half of domain', fontsize=20, color="black")
ax2.plot(uu2d[gl1,0:81],yplus[gl1,0:81])
ax2.plot(vv2d[gl1,0:81],yplus[gl1,0:81])
ax2.plot(uv2d[gl1,0:81],yplus[gl1,0:81])
ax2.set_title('Stresses at X = '+str(np.around(x2d[gl1,0],3))+' m',fontsize=20)
ax3.plot(uu2d[gl2,0:81],yplus[gl2,0:81])
ax3.plot(vv2d[gl2,0:81],yplus[gl2,0:81])
ax3.plot(uv2d[gl2,0:81],yplus[gl2,0:81])
ax3.set_title('Stresses at X = '+str(np.around(x2d[gl2,0],3))+' m',fontsize=20)
#ax3.set_ylim(0,100)
for ax in [ax1,ax2,ax3]:
    ax.tick_params(labelsize='medium')
    ax.xaxis.set_minor_locator(AutoMinorLocator())
    ax.yaxis.set_minor_locator(AutoMinorLocator())
    if ax == ax1:
        ax.legend(['x = '+str(np.around(x2d[gl1,0],3))+' m','x = '+str(np.around(x2d[gl2,0],3))+' m'],loc='best')
        ax.set_xlabel('x (m)')
        ax.set_ylabel('y (m)')
    else:
        ax.legend([r'Normal Stress = $\overline{V_1^{\prime2}}$',r'Normal Stress = $\overline{V_2^{\prime2}}$',
                   r'Shear Stress = $\overline{V_1^{\prime}V_2^{\prime}}$'],
          loc='best')
        ax.grid(True, linestyle='-.')
        ax.set_xlabel('Magnitude of Stress')
        ax.set_ylabel(r'$Y^+$')
fig2a.suptitle('Stresses along Vertical Gridlines in Lower half of the Domain', fontsize=24)
'''
# Near top wall
fig2b = plt.figure()
ax1 = plt.subplot2grid((2,2), (0,0), colspan=2)
ax2 = plt.subplot2grid((2,2), (1,0))
ax3 = plt.subplot2grid((2,2), (1,1))
ax1.contourf(x2d,y2d,v2d,cmap='jet',levels=100,alpha=0.5)
ax1.plot(x2d[gl1,:],y2d[gl1,:],color='black')
ax1.plot(x2d[gl2,:],y2d[gl2,:],color='red')
ax1.plot(x2d[:,81],y2d[:,81],color='green',lw=2)
ax1.set_title('Location of Gridlines for plotting',fontsize=20)
ax1.text(2, 2, 'Upper half of domain', fontsize=20, color="black")
ax1.text(2, 1, 'Lower half of domain', fontsize=20, color="black")
ax2.plot(tau11[gl1,82:],yplus[gl1,82:])
ax2.plot(tau22[gl1,82:],yplus[gl1,82:])
```

```
ax2.plot(tau12[gl1,82:],yplus[gl1,82:])
ax2.plot(tau21[gl1,82:],yplus[gl1,82:])
ax2.set_title('Stresses at X = '+str(np.around(x2d[gl1,0],3))+' m',fontsize=20)
ax3.plot(tau11[gl2,82:],yplus[gl2,82:])
ax3.plot(tau22[gl2,82:],yplus[gl2,82:])
ax3.plot(tau12[gl2,82:],yplus[gl2,82:])
ax3.plot(tau21[gl2,82:],yplus[gl2,82:])
ax3.set_title('Stresses at X = '+str(np.around(x2d[gl2,0],3))+' m',fontsize=20)
#ax3.set_ylim(0,100)
for ax in [ax1,ax2,ax3]:
    ax.tick_params(labelsize='medium')
    ax.xaxis.set_minor_locator(AutoMinorLocator())
    ax.yaxis.set_minor_locator(AutoMinorLocator())
    if ax == ax1:
        ax.legend(['x = '+str(np.around(x2d[gl1,0],3))+' m','x = '+str(np.around(x2d[gl2
    ,0],3))+' m'],loc='best')
        ax.set_xlabel('x (m)')
        ax.set_ylabel('y (m)')
    else:
        ax.legend([r'Normal Stress = $\mu\frac{\partial{\overline{u}}}{\partial{x}}$ - $
    \rho\overline{V_1^{\prime2}}$',
                   r'Normal Stress = $\mu\frac{\partial{\overline{v}}}{\partial{y}}$ - $
    \rho\overline{V_2^{\prime2}}$',
                   r'Shear Stress = $\mu\frac{\partial{\overline{u}}}{\partial{y}}$ - $\
    rho\overline{V_1^{\prime}V_2^{\prime}}$',
                   r'Shear Stress = $\mu\frac{\partial{\overline{v}}}{\partial{x}}$ - $\
    rho\overline{V_1^{\prime}V_2^{\prime}}$'],loc='best')
        ax.grid(True, linestyle='-.')
        ax.set_xlabel('Magnitude of Stress')
        ax.set_ylabel(r'$Y^+$')
fig2b.suptitle('Stresses along Vertical Gridlines in Upper half of the Domain', fontsize
    =24)


# %%Assignment 1.2 - Compute and plot all terms in Momentum Equation
# Convection term
dubub_dx,dubub_dy = dphidx_dy(xf2d,yf2d,np.multiply(u2d,u2d))
dubvb_dx,dubvb_dy = dphidx_dy(xf2d,yf2d,np.multiply(u2d,v2d))
dvbvb_dx,dvbvb_dy = dphidx_dy(xf2d,yf2d,np.multiply(v2d,v2d))

# Pressure source term
dpb_dx,dpb_dy = dphidx_dy(xf2d,yf2d,p2d)

# Diffusion term
d2ub_dxdx,d2ub_dxdy = dphidx_dy(xf2d,yf2d,dub_dx)
d2ub_dydx,d2ub_dydy = dphidx_dy(xf2d,yf2d,dub_dy)
d2vb_dxdx,d2vb_dxdy = dphidx_dy(xf2d,yf2d,dvb_dx)
d2vb_dydx,d2vb_dydy = dphidx_dy(xf2d,yf2d,dvb_dy)

# Reynolds stress term
dufuf_dx,dufuf_dy = dphidx_dy(xf2d,yf2d,uu2d)
dvfvf_dx,dvfvf_dy = dphidx_dy(xf2d,yf2d,vv2d)
dufvf_dx,dufvf_dy = dphidx_dy(xf2d,yf2d,uv2d)

#mom_x = dubub_dx + dubvb_dy + dpb_dx - nu*(d2ub_dxdx - d2ub_dydy) + dufuf_dx + dufvf_dy
#mom_y = dubvb_dx + dvbvb_dy + dpb_dy - nu*(d2vb_dxdx - d2vb_dydy) + dufvf_dx + dvfvf_dy

# In lower Domain
fig3,axes = plt.subplots(nrows=1,ncols=2,constrained_layout=True)
axes[0].plot(-dpb_dx[gl1,0:81],yplus[gl1,0:81],label=r'$-\left(\frac{\partial \overline{
    p}} {\partial{x}}\right)$')
axes[0].plot(nu*d2ub_dydy[gl1,0:81],yplus[gl1,0:81],label=r'$\nu\left(\frac{\partial
    ^{2}\overline{u}} {\partial{y^{2}}}\right)$')
axes[0].plot(-dufvf_dy[gl1,0:81],yplus[gl1,0:81],label=r'$-\left(\frac{\partial{\
    overline{u^{\prime}v^{\prime}}}} {\partial{y}}\right)$')
axes[1].plot(nu*d2ub_dxdx[gl1,0:81],yplus[gl1,0:81],label=r'$\nu\left(\frac{\partial
```

```
    ^{2}\overline{u}} {\partial{x^{2}}}\right)$')
axes[1].plot(-dufuf_dx[gl1,0:81],yplus[gl1,0:81],label=r'$-\left(\frac{\partial{\
    overline{u^{\prime2}}}} {\partial{x}}\right)$')
axes[1].plot((dubub_dx[gl1,0:81]+dubvb_dy[gl1,0:81]),yplus[gl1,0:81],
    label=r'LHS = $\left(\frac{\partial{\overline{u}}{\overline{u}}} {\partial{x}}\right
    )$ + $\left(\frac{\partial{\overline{u}}{\overline{v}}} {\partial{y}}\right)$')
for ax in axes:
    ax.tick_params(labelsize='medium')
    ax.xaxis.set_minor_locator(AutoMinorLocator())
    ax.yaxis.set_minor_locator(AutoMinorLocator())
    ax.grid(True, linestyle='-.')
    ax.legend(fontsize=16,loc='best')
    ax.set_xlabel('.', color=(0, 0, 0, 0),fontsize=20)
    ax.set_ylabel('.', color=(0, 0, 0, 0),fontsize=20)
    ax.set_xlim([-1.5,0.3])
fig3.add_subplot(111, frameon=False)
plt.tick_params(labelcolor="none", top=False, bottom=False, left=False, right=False)
plt.xlabel("Values of various terms in X-momentum equation",fontsize=18)
plt.ylabel(r"$Y^+$",fontsize=18)
fig3.suptitle('Terms in X-momentum equation at X = '+str(np.around(x2d[gl1,0],3))+' m in
     lower half of domain', fontsize=24)

fig3a,axes = plt.subplots(nrows=1,ncols=2,constrained_layout=True)
axes[0].plot(-dpb_dy[gl1,0:81],yplus[gl1,0:81],label=r'$-\left(\frac{\partial \overline{
    p}} {\partial{y}}\right)$')
axes[0].plot(nu*d2vb_dydy[gl1,0:81],yplus[gl1,0:81],label=r'$\nu\left(\frac{\partial
    ^{2}\overline{v}} {\partial{y^{2}}}\right)$')
axes[0].plot(-dvfvf_dy[gl1,0:81],yplus[gl1,0:81],label=r'$-\left(\frac{\partial{\
    overline{v^{\prime2}}}} {\partial{y}}\right)$')
axes[1].plot(nu*d2vb_dxdx[gl1,0:81],yplus[gl1,0:81],label=r'$\nu\left(\frac{\partial
    ^{2}\overline{v}} {\partial{x^{2}}}\right)$')
axes[1].plot(-dufvf_dx[gl1,0:81],yplus[gl1,0:81],label=r'$-\left(\frac{\partial{\
    overline{u^{\prime}v^{\prime}}}} {\partial{x}}\right)$')
axes[1].plot((dubvb_dx[gl1,0:81]+dvbvb_dy[gl1,0:81]),yplus[gl1,0:81],
    label=r'LHS = $\left(\frac{\partial{\overline{u}}{\overline{v}}} {\partial{x}}\right
    )$ + $\left(\frac{\partial{\overline{v}}{\overline{v}}} {\partial{y}}\right)$')
for ax in axes:
    ax.tick_params(labelsize='medium')
    ax.xaxis.set_minor_locator(AutoMinorLocator())
    ax.yaxis.set_minor_locator(AutoMinorLocator())
    ax.grid(True, linestyle='-.')
    ax.legend(fontsize=16,loc='best')
    ax.set_xlabel('.', color=(0, 0, 0, 0),fontsize=20)
    ax.set_ylabel('.', color=(0, 0, 0, 0),fontsize=20)
    ax.set_xlim([-0.7,0.1])
fig3a.add_subplot(111, frameon=False)
plt.tick_params(labelcolor="none", top=False, bottom=False, left=False, right=False)
plt.xlabel("Values of various terms in X-momentum equation",fontsize=18)
plt.ylabel(r"$Y^+$",fontsize=18)
fig3a.suptitle('Terms in Y-momentum equation at X = '+str(np.around(x2d[gl1,0],3))+' m
    in lower half of domain', fontsize=24)

fig4,axes = plt.subplots(nrows=1,ncols=2,constrained_layout=True)
axes[0].plot(-dpb_dx[gl2,0:81],yplus[gl2,0:81],label=r'$-\left(\frac{\partial \overline{
    p}} {\partial{x}}\right)$')
axes[0].plot(nu*d2ub_dydy[gl2,0:81],yplus[gl2,0:81],label=r'$\nu\left(\frac{\partial
    ^{2}\overline{u}} {\partial{y^{2}}}\right)$')
axes[0].plot(-dufvf_dy[gl2,0:81],yplus[gl2,0:81],label=r'$-\left(\frac{\partial{\
    overline{u^{\prime}v^{\prime}}}} {\partial{y}}\right)$')
axes[1].plot(nu*d2ub_dxdx[gl2,0:81],yplus[gl2,0:81],label=r'$\nu\left(\frac{\partial
    ^{2}\overline{u}} {\partial{x^{2}}}\right)$')
axes[1].plot(-dufuf_dx[gl2,0:81],yplus[gl2,0:81],label=r'$-\left(\frac{\partial{\
    overline{u^{\prime2}}}} {\partial{x}}\right)$')
axes[1].plot((dubub_dx[gl2,0:81]+dubvb_dy[gl2,0:81]),yplus[gl2,0:81],label=r'LHS = $\
    left(\frac{\partial{\overline{u}}{\overline{u}}} {\partial{x}}\right)$ + $\left(\
    frac{\partial{\overline{u}}{\overline{v}}} {\partial{y}}\right)$')
```

```
for ax in axes:
    ax.tick_params(labelsize='medium')
    ax.xaxis.set_minor_locator(AutoMinorLocator())
    ax.yaxis.set_minor_locator(AutoMinorLocator())
    ax.grid(True, linestyle='-.')
    ax.legend(fontsize=16,loc='best')
    ax.set_xlabel('.', color=(0, 0, 0, 0),fontsize=20)
    ax.set_ylabel('.', color=(0, 0, 0, 0),fontsize=20)
fig4.add_subplot(111, frameon=False)
plt.tick_params(labelcolor="none", top=False, bottom=False, left=False, right=False)
plt.xlabel("Values of various terms in X-momentum equation",fontsize=18)
plt.ylabel(r"$Y^+$",fontsize=18)
fig4.suptitle('Terms in X-momentum equation at X = '+str(np.around(x2d[gl2,0],3))+' m in
     lower half of domain', fontsize=24)

fig4a,axes = plt.subplots(nrows=1,ncols=2,constrained_layout=True)
axes[0].plot(-dpb_dy[gl2,0:81],yplus[gl2,0:81],label=r'$-\left(\frac{\partial \overline{
    p}} {\partial{y}}\right)$')
axes[0].plot(nu*d2vb_dydy[gl2,0:81],yplus[gl2,0:81],label=r'$\nu\left(\frac{\partial
    ^{2}\overline{v}} {\partial{y^{2}}}\right)$')
axes[0].plot(-dvfvf_dy[gl2,0:81],yplus[gl2,0:81],label=r'$-\left(\frac{\partial{\
    overline{v^{\prime2}}}} {\partial{y}}\right)$')
axes[1].plot(nu*d2vb_dxdx[gl2,0:81],yplus[gl2,0:81],label=r'$\nu\left(\frac{\partial
    ^{2}\overline{v}} {\partial{x^{2}}}\right)$')
axes[1].plot(-dufvf_dx[gl2,0:81],yplus[gl2,0:81],label=r'$-\left(\frac{\partial{\
    overline{u^{\prime}v^{\prime}}}} {\partial{x}}\right)$')
axes[1].plot((dubvb_dx[gl2,0:81]+dvbvb_dy[gl2,0:81]),yplus[gl2,0:81],label=r'LHS = $\
    left(\frac{\partial{\overline{u}}{\overline{v}}} {\partial{x}}\right)$ + $\left(\
    frac{\partial{\overline{v}}{\overline{v}}} {\partial{y}}\right)$')
for ax in axes:
    ax.tick_params(labelsize='medium')
    ax.xaxis.set_minor_locator(AutoMinorLocator())
    ax.yaxis.set_minor_locator(AutoMinorLocator())
    ax.grid(True, linestyle='-.')
    ax.legend(fontsize=16,loc='best')
    ax.set_xlabel('.', color=(0, 0, 0, 0),fontsize=20)
    ax.set_ylabel('.', color=(0, 0, 0, 0),fontsize=20)
fig4a.add_subplot(111, frameon=False)
plt.tick_params(labelcolor="none", top=False, bottom=False, left=False, right=False)
plt.xlabel("Values of various terms in X-momentum equation",fontsize=18)
plt.ylabel(r"$Y^+$",fontsize=18)
fig4a.suptitle('Terms in Y-momentum equation at X = '+str(np.around(x2d[gl2,0],3))+' m
    in lower half of domain', fontsize=24)

# In Upper Domain
grid = 0
fig3b,axes = plt.subplots(nrows=1,ncols=2,constrained_layout=True)
axes[0].plot(-dpb_dx[grid,82:],yplus[grid,82:],label=r'$-\left(\frac{\partial \overline{
    p}} {\partial{x}}\right)$')
axes[0].plot(nu*d2ub_dydy[grid,82:],yplus[grid,82:],label=r'$\nu\left(\frac{\partial
    ^{2}\overline{u}} {\partial{y^{2}}}\right)$')
axes[0].plot(-dufvf_dy[grid,82:],yplus[grid,82:],label=r'$-\left(\frac{\partial{\
    overline{u^{\prime}v^{\prime}}}} {\partial{y}}\right)$')
axes[1].plot(nu*d2ub_dxdx[grid,82:],yplus[grid,82:],label=r'$\nu\left(\frac{\partial
    ^{2}\overline{u}} {\partial{x^{2}}}\right)$')
axes[1].plot(-dufuf_dx[grid,82:],yplus[grid,82:],label=r'$-\left(\frac{\partial{\
    overline{u^{\prime2}}}} {\partial{x}}\right)$')
axes[1].plot((dubub_dx[grid,82:]+dubvb_dy[grid,82:]),yplus[grid,82:],label=r'LHS = $\
    left(\frac{\partial{\overline{u}}{\overline{u}}} {\partial{x}}\right)$ + $\left(\
    frac{\partial{\overline{u}}{\overline{v}}} {\partial{y}}\right)$')
for ax in axes:
    ax.tick_params(labelsize='medium')
    ax.xaxis.set_minor_locator(AutoMinorLocator())
    ax.yaxis.set_minor_locator(AutoMinorLocator())
    ax.grid(True, linestyle='-.')
    ax.legend(fontsize=16,loc='best')
```

```
    ax.set_xlabel('.', color=(0, 0, 0, 0),fontsize=20)
    ax.set_ylabel('.', color=(0, 0, 0, 0),fontsize=20)
fig3b.add_subplot(111, frameon=False)
plt.tick_params(labelcolor="none", top=False, bottom=False, left=False, right=False)
plt.xlabel("Values of various terms in X-momentum equation",fontsize=18)
plt.ylabel(r"$Y^+$",fontsize=18)
fig3b.suptitle('Terms in X-momentum equation at X = '+str(np.around(x2d[grid,0],3))+' m
    in upper half of domain', fontsize=24)

fig4b,axes = plt.subplots(nrows=1,ncols=2,constrained_layout=True)
axes[0].plot(-dpb_dy[grid,82:],yplus[grid,82:],label=r'$-\left(\frac{\partial \overline{
    p}} {\partial{y}}\right)$')
axes[0].plot(nu*d2vb_dydy[grid,82:],yplus[grid,82:],label=r'$\nu\left(\frac{\partial
    ^{2}\overline{v}} {\partial{y^{2}}}\right)$')
axes[0].plot(-dvfvf_dy[grid,82:],yplus[grid,82:],label=r'$-\left(\frac{\partial{\
    overline{v^{\prime2}}}} {\partial{y}}\right)$')
axes[1].plot(nu*d2vb_dxdx[grid,82:],yplus[grid,82:],label=r'$\nu\left(\frac{\partial
    ^{2}\overline{v}} {\partial{x^{2}}}\right)$')
axes[1].plot(-dufvf_dx[grid,82:],yplus[grid,82:],label=r'$-\left(\frac{\partial{\
    overline{u^{\prime}v^{\prime}}}} {\partial{x}}\right)$')
axes[1].plot((dubvb_dx[grid,82:]+dvbvb_dy[grid,82:]),yplus[grid,82:],label=r'LHS = $\
    left(\frac{\partial{\overline{u}}{\overline{v}}} {\partial{x}}\right)$ + $\left(\
    frac{\partial{\overline{v}}{\overline{v}}} {\partial{y}}\right)$')
for ax in axes:
    ax.tick_params(labelsize='medium')
    ax.xaxis.set_minor_locator(AutoMinorLocator())
    ax.yaxis.set_minor_locator(AutoMinorLocator())
    ax.grid(True, linestyle='-.')
    ax.legend(fontsize=16,loc='best')
    ax.set_xlabel('.', color=(0, 0, 0, 0),fontsize=20)
    ax.set_ylabel('.', color=(0, 0, 0, 0),fontsize=20)
fig4b.add_subplot(111, frameon=False)
plt.tick_params(labelcolor="none", top=False, bottom=False, left=False, right=False)
plt.xlabel("Values of various terms in X-momentum equation",fontsize=18)
plt.ylabel(r"$Y^+$",fontsize=18)
fig4b.suptitle('Terms in Y-momentum equation at X = '+str(np.around(x2d[grid,0],3))+' m
    in upper half of domain', fontsize=24)

# %%Assignment 1.3 - Plot the vector field F_N
k = 2
k1 = 1
Mag = np.hypot(dufuf_dx,dvfvf_dy)
V1 = (-dufuf_dx/Mag)
V2 = (-dvfvf_dy/Mag)
fig6 = plt.figure()
axes = fig6.add_axes([0.1,0.1,0.8,0.8])
cc = axes.quiver(x2d[::k1,::k1],y2d[::k1,::k1],V1[::k1,::k1],V2[::k1,::k1],Mag,pivot='
    mid',cmap='Reds',width=0.001,headwidth=5,headlength=6)
fig6.colorbar(cc,ax=axes)
axes.set_xlabel('X (m)')
axes.set_ylabel('Y (m)')
axes.set_title(r'Vector Field - $F_{N}$',fontsize=20)

fig6a = plt.figure()
axes = fig6a.add_axes([0.1,0.1,0.8,0.8])
cc = axes.contourf(x2d,y2d,vv2d,100,cmap='jet',alpha=0.8)
fig6a.colorbar(cc,ax=axes)
axes.quiver(x2d[::k,::k],y2d[::k,::k],V1[::k,::k],V2[::k,::k],pivot='mid',width=0.001,
    headwidth=4,headlength=5)
axes.set_xlabel('X (m)')
axes.set_ylabel('Y (m)')
axes.set_title(r'Vector Field - $F_{N}$',fontsize=20)


# %%Assignment 1.4 - Plot the vector field F_S
k = 2
```

```
k1 = 1
Mag1 = np.hypot(dufvf_dy,dufvf_dx)
VV1 = (-dufvf_dy/Mag1)
VV2 = (-dufvf_dx/Mag1)
fig7 = plt.figure()
axes = fig7.add_axes([0.1,0.1,0.8,0.8])
cc = axes.quiver(x2d[::k1,::k1],y2d[::k1,::k1],VV1[::k1,::k1],VV2[::k1,::k1],Mag,pivot='mid',cmap='Reds',width=0.001,headwidth=5,headlength=6)
fig7.colorbar(cc,ax=axes)
axes.set_xlabel('X (m)')
axes.set_ylabel('Y (m)')
axes.set_title(r'Vector Field - $F_{S}$',fontsize=20)

fig7a = plt.figure()
axes = fig7a.add_axes([0.1,0.1,0.8,0.8])
cc = axes.contourf(x2d,y2d,vv2d,100,cmap='jet',alpha=0.8)
fig6a.colorbar(cc,ax=axes)
axes.quiver(x2d[::k,::k],y2d[::k,::k],VV1[::k,::k],VV2[::k,::k],pivot='mid',width=0.001,headwidth=4,headlength=5)
axes.set_xlabel('X (m)')
axes.set_ylabel('Y (m)')
axes.set_title(r'Vector Field - $F_{S}$',fontsize=20)

# %%Assignment 1.5 - Plot the production terms
NS = (np.multiply(uu2d,dub_dx) + np.multiply(vv2d,dvb_dy))
SS = (np.multiply(uv2d,dub_dy) + np.multiply(uv2d,dvb_dx))
Pk = -(NS + SS)
SS[:,(0,-1)] = 1e-20
NS[:,(0,-1)] = 1e-20

fig8,axes = plt.subplots(nrows=2,ncols=1,constrained_layout=True)
axes[0].plot(yplus[gl1,0:81],Pk[gl1,0:81],label='X = '+str(np.around(x2d[gl1,0],3))+' m')
axes[1].plot(yplus[gl2,0:81],Pk[gl2,0:81],label='X = '+str(np.around(x2d[gl2,0],3))+' m')
for ax in axes:
    ax.tick_params(labelsize='medium')
    ax.xaxis.set_minor_locator(AutoMinorLocator())
    ax.yaxis.set_minor_locator(AutoMinorLocator())
    ax.grid(True, linestyle='-.')
    ax.legend(fontsize=16,loc='best')
    ax.set_xlabel('.', color=(0, 0, 0, 0),fontsize=20)
    ax.set_ylabel('.', color=(0, 0, 0, 0),fontsize=20)
fig8.add_subplot(111, frameon=False)
plt.tick_params(labelcolor="none", top=False, bottom=False, left=False, right=False)
plt.xlabel(r'$Y^+$',fontsize=18)
plt.ylabel(r'$P^{k}$',fontsize=18)
fig8.suptitle('Value of Production term in Turbulent KE equation (Upper Half of the Domain)', fontsize=24)

NS_contribution = np.divide(np.abs(NS),(np.abs(SS)+np.abs(NS)))
NS_contribution[:,0] = 0
NS_contribution[:,0] = 0
SS_contribution = np.divide(np.abs(SS),(np.abs(SS)+np.abs(NS)))
SS_contribution[:,0] = 0
SS_contribution[:,0] = 0

fig9,axes = plt.subplots(nrows=2,ncols=1,constrained_layout=True)
vmin1 = 0
vmax1 = np.max(np.max(Pk))
v1 = np.linspace(vmin1,vmax1,10)
vmin2 = np.min(np.min(Pk))
vmax2 = 0
v2 = np.linspace(vmin2,vmax2,10)
a1 = axes[0].contourf(x2d,y2d,Pk,cmap='gist_gray',levels=v1,extend='min')
a1.cmap.set_under('black')
```

```
a2 = axes[1].contourf(x2d,y2d,Pk,cmap='gist_yarg',levels=v2,extend='max')
a2.cmap.set_over('black')
cb1 = fig9.colorbar(a1,ax=axes[0])
cb2 = fig9.colorbar(a2,ax=axes[1])
cb1.ax.tick_params(labelsize=16)
cb2.ax.tick_params(labelsize=16)
a1.ax.tick_params(labelsize=14)
a2.ax.tick_params(labelsize=14)
axes[0].set_title(r'Contour of Production term $(P^k)$ - Areas of Postive Production',
    fontsize=20)
axes[1].set_title(r'Contour of Production term $(P^k)$ - Areas of Negative Production',
    fontsize=20)
for ax in axes:
    ax.set_xlabel('X (m)',fontsize=18)
    ax.set_ylabel('Y (m)',fontsize=18)

fig9a = plt.figure()
ax1 = plt.subplot2grid((2,4), (0,0), colspan=2)
ax2 = plt.subplot2grid((2,4), (0,2), colspan=2)
ax3 = plt.subplot2grid((2,4), (1,0))
ax4 = plt.subplot2grid((2,4), (1,1))
ax5 = plt.subplot2grid((2,4), (1,2))
ax6 = plt.subplot2grid((2,4), (1,3))
a1 = ax1.contourf(x2d,y2d,np.multiply(np.abs(Pk),NS_contribution),cmap='jet',levels=50)
fig9a.colorbar(a1,ax=ax1)
ax1.set_title(r'Contribution of Normal Stress to the Production term $(P^k)$',fontsize
    =18)
a2 = ax2.contourf(x2d,y2d,np.multiply(np.abs(Pk),SS_contribution),cmap='jet',levels=50)
fig9a.colorbar(a2,ax=ax2)
ax2.set_title(r'Contribution of Shear Stress to the Production term $(P^k)$',fontsize
    =18)
a3 = ax3.contourf(x2d,y2d,np.multiply(np.abs(Pk),NS_contribution),cmap='jet',levels=50)
fig9a.colorbar(a3,ax=ax3,orientation='horizontal')
ax3.set_xlim(0,2)
ax3.set_ylim(0.6,1.4)
ax3.set_title(r'Near First Hill',fontsize=14)
a4 = ax4.contourf(x2d,y2d,np.multiply(np.abs(Pk),NS_contribution),cmap='jet',levels=50)
fig9a.colorbar(a4,ax=ax4,orientation='horizontal')
ax4.set_xlim(8.2,9)
ax4.set_ylim(0.8,1.2)
ax4.set_title(r'Near Second Hill',fontsize=14)
a5 = ax5.contourf(x2d,y2d,np.multiply(np.abs(Pk),SS_contribution),cmap='jet',levels=50)
fig9a.colorbar(a5,ax=ax5,orientation='horizontal')
ax5.set_xlim(0,2)
ax5.set_ylim(0.5,1.3)
ax5.set_title(r'Near First Hill',fontsize=14)
a6 = ax6.contourf(x2d,y2d,np.multiply(np.abs(Pk),SS_contribution),cmap='jet',levels=50)
fig9a.colorbar(a6,ax=ax6,orientation='horizontal')
ax6.set_xlim(8.6,9)
ax6.set_ylim(0.8,1.1)
ax6.set_title(r'Near Second Hill',fontsize=14)
for ax in [ax1,ax2]:
    ax.set_xlabel('X (m)',fontsize=16)
    ax.set_ylabel('Y (m)',fontsize=16)

# %%Assignment 1.6 - Plot the dissipation and compare with production
# Dissipation plot
vmin = 0.1
vmin2 = 0.05
vmax = np.max(np.max(diss_RANS_2d))
v = np.linspace(vmin, vmax, 50, endpoint=True)
v2 = np.linspace(vmin2, vmax, 50, endpoint=True)
fig10 = plt.figure()
ax1 = plt.subplot2grid((2,2), (0,0), colspan=2)
ax2 = plt.subplot2grid((2,2), (1,0))
ax3 = plt.subplot2grid((2,2), (1,1))
```

```
cax1 = ax1.contourf(x2d,y2d,diss_RANS_2d,cmap='jet',levels=50)
cax2 = ax2.contourf(x2d,y2d,diss_RANS_2d,cmap='jet',levels=v,extend='both')
cax2.cmap.set_over('darkred')
cax2.cmap.set_under('darkblue')
cax3 = ax3.contourf(x2d,y2d,diss_RANS_2d,cmap='jet',levels=v2,extend='both')
cax3.cmap.set_over('darkred')
cax3.cmap.set_under('darkblue')
for ax in [ax1,ax2,ax3]:
    ax.set_xlabel('X (m)')
    ax.set_ylabel('Y (m)')
cb1 =fig10.colorbar(cax1,ax=ax1,fraction=0.05)
cb2 = fig10.colorbar(cax2,ax=ax2,fraction=0.1)
cb3 =fig10.colorbar(cax3,ax=ax3,fraction=0.1)
ax2.set_xlim(0,0.6)
ax2.set_ylim(0.9,1.05)
ax3.set_xlim(8.4,9)
ax3.set_ylim(0.8,1.1)
plt.tight_layout()
ax1.set_title(r'Contour of Dissipation ($\epsilon$) of Turbulent KE', fontsize=24)
ax2.set_title(r'Dissipation ($\epsilon$) near the first hill', fontsize=20)
ax3.set_title(r'Dissipation ($\epsilon$) near the second hill', fontsize=20)

# Production and dissipation equilibrium
vmin = -0.01
vmax = 0.01
v = np.linspace(vmin, vmax, 50, endpoint=True)
#v = np.append(v,0.24)
fig11, axes = plt.subplots()
css = axes.contourf(x2d,y2d,Pk-diss_RANS_2d,cmap='seismic',levels=v,extend='both')
#cs.cmap.set_over('dimgray')
#cs.cmap.set_under('bisque')
#cs.changed()
cb = fig11.colorbar(css,ticks=[-0.01,-0.005,0,0.005,0.01])
axes.set_title(r'Local Equilibrium of Production ($P_k$) and Dissipation ($\epsilon$)
    terms',fontsize=24)

# %%Assign 1.7 - Reynolds Stress Equation
# Constants for Modelling
C_mu = 0.09
C_1 = 1.5
C_2 = 0.6
C_1w = 0.5
C_2w = 0.3
sigma_k = 1

# Damping function - f
#f = k_RANS_2d**1.5 / 2.55 * diss_RANS_2d * [(np.absolute(nx[:,0] * (x2d-x2d[:,0]))) + (
    np.absolute(ny[:,0] * (y2d-y2d[:,0])))]
distance = np.zeros((ni,nj))
for i in range(ni):
    for j in range(nj):
        distance[i,j] = WallDistance(x2d,y2d,x2d[i,j],y2d[i,j])
distance[:,(0,-1)] = 1e-10

f = (k_RANS_2d**1.5) / (2.55 * diss_RANS_2d * distance)
f[:,(0,-1)] = 1e-10
f_abs = f.copy()

# Normalize damping function
for i in range(ni):
    for j in range(nj):
        if j <= 81:
            f[i,j] = f_abs[i,j]/(np.amax(f_abs[:,:81], axis=1)[i])
        else:
            f[i,j] = f_abs[i,j]/(np.amax(f_abs[:,81:], axis=1)[i])

```

```
# Wall Normals for Pressure Strain Term Calculations
nx_bottom,ny_bottom,nx_top,ny_top = WallNormal(xf2d,yf2d)
nx_bottom = np.transpose(np.array([nx_bottom,]*nj))
ny_bottom = np.transpose(np.array([ny_bottom,]*nj))
nx_top = np.transpose(np.array([nx_top,]*nj))
ny_top = np.transpose(np.array([ny_top,]*nj))

# Model Diffusion terms
Constant = (np.divide(np.multiply(k_RANS_2d,k_RANS_2d),diss_RANS_2d) * (C_mu/sigma_k))

d2ufuf_dxdx,d2ufuf_dxdy = dphidx_dy(xf2d,yf2d,dufuf_dx)
d2ufuf_dydx,d2ufuf_dydy = dphidx_dy(xf2d,yf2d,dufuf_dy)
d2vfvf_dxdx,d2vfvf_dxdy = dphidx_dy(xf2d,yf2d,dvfvf_dx)
d2vfvf_dydx,d2vfvf_dydy = dphidx_dy(xf2d,yf2d,dvfvf_dy)
d2ufvf_dxdx,d2ufvf_dxdy = dphidx_dy(xf2d,yf2d,dufvf_dx)
d2ufvf_dydx,d2ufvf_dydy = dphidx_dy(xf2d,yf2d,dufvf_dy)

# For Reynolds Stress - u'v'
LHS_12 = np.multiply(u2d,dufvf_dx) + np.multiply(v2d,dufvf_dy)
LHS_21 = LHS_12
# Production - u'v'
P_12 = -(np.multiply(uu2d,dvb_dx) + np.multiply(uv2d,dvb_dy) + np.multiply(uv2d,dub_dx)
    + np.multiply(vv2d,dub_dy))
P_21 = P_12
# Turbulent Diffusion - u'v'
diff_12 = np.multiply(Constant,(d2ufvf_dxdx+d2ufvf_dydy))
diff_21 = diff_12
# Viscous Diffusion - u'v'
diffvisc_12 = nu * (d2ufvf_dxdx + d2ufvf_dydy)
diffvisc_21 = diffvisc_12
# Dissipation - u'v'
diss_12 = np.zeros((ni,nj))
diss_21 = np.zeros((ni,nj))
# Pressure-strain term - u'v'
phi12_1 = -C_1 * np.divide(diss_RANS_2d,k_RANS_2d) * (uv2d - (2/3 * k_RANS_2d))
phi12_2 = -C_2 * (P_12 - (2/3 * Pk))
phi21_1 = phi12_1
phi21_2 = phi12_2

# For Reynolds Stress - u'u'
LHS_11 = np.multiply(u2d,dufuf_dx) + np.multiply(v2d,dufuf_dy)
# Production - u'u'
P_11 = -2 * (np.multiply(uu2d,dub_dx) + np.multiply(uv2d,dub_dy))
# Turbulent Diffusion - u'u'
diff_11 = np.multiply(Constant,(d2ufuf_dxdx+d2ufuf_dydy))
# Viscous Diffusion - u'u'
diffvisc_11 = nu * (d2ufuf_dxdx + d2ufuf_dydy)
# Dissipation - u'u'
diss_11 = (2/3) * diss_RANS_2d
# Pressure-strain term - u'u'
phi11_1 = -C_1 * np.divide(diss_RANS_2d,k_RANS_2d) * (uu2d - (2/3 * k_RANS_2d))
phi11_2 = -C_2 * (P_11 - (2/3 * Pk))

# For Reynolds Stress - v'v'
LHS_22 = np.multiply(u2d,dvfvf_dx) + np.multiply(v2d,dvfvf_dy)
# Production - v'v'
P_22 = -2 * (np.multiply(uv2d,dvb_dx) + np.multiply(vv2d,dvb_dy))
# Turbulent Diffusion - v'v'
diff_22 = np.multiply(Constant,(d2vfvf_dxdx+d2vfvf_dydy))
# Viscous Diffusion - v'v'
diffvisc_22 = nu * (d2vfvf_dxdx + d2vfvf_dydy)
# Dissipation - v'v'
diss_22 = (2/3) * diss_RANS_2d
# Pressure-strain term - v'v'
phi22_1 = -C_1 * np.divide(diss_RANS_2d,k_RANS_2d) * (vv2d - (2/3 * k_RANS_2d))
phi22_2 = -C_2 * (P_22 - (2/3 * Pk))
```

```
778
779 # Pressure Strain IP Terms (near wall terms)
780 phi11_1w = np.zeros((ni,nj))
781 phi11_2w = np.zeros((ni,nj))
782 phi12_1w = np.zeros((ni,nj))
783 phi12_2w = np.zeros((ni,nj))
784 phi22_1w = np.zeros((ni,nj))
785 phi22_2w = np.zeros((ni,nj))
786 phi21_1w = np.zeros((ni,nj))
787 phi21_2w = np.zeros((ni,nj))
788 for i in range(ni):
789     for j in range (nj):
790         if j<=81:
791             phi11_1w[i,j] = C_1w * (diss_RANS_2d[i,j]/k_RANS_2d[i,j]) * ((uu2d[i,j]*
    nx_bottom[i,j]*nx_bottom[i,j])+(2*uv2d[i,j]*nx_bottom[i,j]*ny_bottom[i,j])+(vv2d[i,j
    ]*ny_bottom[i,j]*ny_bottom[i,j])-(1.5*uu2d[i,j]*nx_bottom[i,j]*nx_bottom[i,j])-(1.5*
    uv2d[i,j]*nx_bottom[i,j]*ny_bottom[i,j])-(1.5*uu2d[i,j]*nx_bottom[i,j]*nx_bottom[i,j
    ])-(1.5*uv2d[i,j]*nx_bottom[i,j]*ny_bottom[i,j])) * f[i,j]
792             phi11_2w[i,j] = C_2w * (diss_RANS_2d[i,j]/k_RANS_2d[i,j]) * ((phi11_2[i,j]*
    nx_bottom[i,j]*nx_bottom[i,j])+(2*phi12_2[i,j]*nx_bottom[i,j]*ny_bottom[i,j])+(
    phi22_2[i,j]*ny_bottom[i,j]*ny_bottom[i,j])-(1.5*phi11_2[i,j]*nx_bottom[i,j]*
    nx_bottom[i,j])-(1.5*phi12_2[i,j]*nx_bottom[i,j]*ny_bottom[i,j])-(1.5*phi11_2[i,j]*
    nx_bottom[i,j]*nx_bottom[i,j])-(1.5*phi12_2[i,j]*nx_bottom[i,j]*ny_bottom[i,j])) * f
    [i,j]
793             phi22_1w[i,j] = C_1w * (diss_RANS_2d[i,j]/k_RANS_2d[i,j]) * ((uu2d[i,j]*
    nx_bottom[i,j]*nx_bottom[i,j])+(2*uv2d[i,j]*nx_bottom[i,j]*ny_bottom[i,j])+(vv2d[i,j
    ]*ny_bottom[i,j]*ny_bottom[i,j])-(1.5*uv2d[i,j]*nx_bottom[i,j]*ny_bottom[i,j])-(1.5*
    vv2d[i,j]*ny_bottom[i,j]*ny_bottom[i,j])-(1.5*uv2d[i,j]*nx_bottom[i,j]*ny_bottom[i,j
    ])-(1.5*vv2d[i,j]*ny_bottom[i,j]*ny_bottom[i,j])) * f[i,j]
794             phi22_2w[i,j] = C_2w * (diss_RANS_2d[i,j]/k_RANS_2d[i,j]) * ((phi11_2[i,j]*
    nx_bottom[i,j]*nx_bottom[i,j])+(2*phi12_2[i,j]*nx_bottom[i,j]*ny_bottom[i,j])+(
    phi22_2[i,j]*ny_bottom[i,j]*ny_bottom[i,j])-(1.5*phi12_2[i,j]*nx_bottom[i,j]*
    ny_bottom[i,j])-(1.5*phi22_2[i,j]*ny_bottom[i,j]*ny_bottom[i,j])-(1.5*phi12_2[i,j]*
    nx_bottom[i,j]*ny_bottom[i,j])-(1.5*phi22_2[i,j]*ny_bottom[i,j]*ny_bottom[i,j])) * f
    [i,j]
795             phi12_1w[i,j] = C_1w * (diss_RANS_2d[i,j]/k_RANS_2d[i,j]) * ((1.5*uu2d[i,j]*
    nx_bottom[i,j]*ny_bottom[i,j])-(1.5*uv2d[i,j]*ny_bottom[i,j]*ny_bottom[i,j])-(1.5*
    uv2d[i,j]*nx_bottom[i,j]*nx_bottom[i,j])-(1.5*vv2d[i,j]*nx_bottom[i,j]*ny_bottom[i,j
    ])) * f[i,j]
796             phi12_2w[i,j] = C_2w * (diss_RANS_2d[i,j]/k_RANS_2d[i,j]) * ((1.5*phi11_2[i,
    j]*nx_bottom[i,j]*ny_bottom[i,j])-(1.5*phi12_2[i,j]*ny_bottom[i,j]*ny_bottom[i,j])
    -(1.5*phi12_2[i,j]*nx_bottom[i,j]*nx_bottom[i,j])-(1.5*phi22_2[i,j]*nx_bottom[i,j]*
    ny_bottom[i,j])) * f[i,j]
797         else:
798             phi11_1w[i,j] = C_1w * (diss_RANS_2d[i,j]/k_RANS_2d[i,j]) * ((uu2d[i,j]*
    nx_top[i,j]*nx_top[i,j])+(2*uv2d[i,j]*nx_top[i,j]*ny_top[i,j])+(vv2d[i,j]*ny_top[i,j
    ]*ny_top[i,j])-(1.5*uu2d[i,j]*nx_top[i,j]*nx_top[i,j])-(1.5*uv2d[i,j]*nx_top[i,j]*
    ny_top[i,j])-(1.5*uu2d[i,j]*nx_top[i,j]*nx_top[i,j])-(1.5*uv2d[i,j]*nx_top[i,j]*
    ny_top[i,j])) * f[i,j]
799             phi11_2w[i,j] = C_2w * (diss_RANS_2d[i,j]/k_RANS_2d[i,j]) * ((phi11_2[i,j]*
    nx_top[i,j]*nx_top[i,j])+(2*phi12_2[i,j]*nx_top[i,j]*ny_top[i,j])+(phi22_2[i,j]*
    ny_top[i,j]*ny_top[i,j])-(1.5*phi11_2[i,j]*nx_top[i,j]*nx_top[i,j])-(1.5*phi12_2[i,j
    ]*nx_top[i,j]*ny_top[i,j])-(1.5*phi11_2[i,j]*nx_top[i,j]*nx_top[i,j])-(1.5*phi12_2[i
    ,j]*nx_top[i,j]*ny_top[i,j])) * f[i,j]
800             phi22_1w[i,j] = C_1w * (diss_RANS_2d[i,j]/k_RANS_2d[i,j]) * ((uu2d[i,j]*
    nx_top[i,j]*nx_top[i,j])+(2*uv2d[i,j]*nx_top[i,j]*ny_top[i,j])+(vv2d[i,j]*ny_top[i,j
    ]*ny_top[i,j])-(1.5*uv2d[i,j]*nx_top[i,j]*ny_top[i,j])-(1.5*vv2d[i,j]*ny_top[i,j]*
    ny_top[i,j])-(1.5*uv2d[i,j]*nx_top[i,j]*ny_top[i,j])-(1.5*vv2d[i,j]*ny_top[i,j]*
    ny_top[i,j])) * f[i,j]
801             phi22_2w[i,j] = C_2w * (diss_RANS_2d[i,j]/k_RANS_2d[i,j]) * ((phi11_2[i,j]*
    nx_top[i,j]*nx_top[i,j])+(2*phi12_2[i,j]*nx_top[i,j]*ny_top[i,j])+(phi22_2[i,j]*
    ny_top[i,j]*ny_top[i,j])-(1.5*phi12_2[i,j]*nx_top[i,j]*ny_top[i,j])-(1.5*phi22_2[i,j
    ]*ny_top[i,j]*ny_top[i,j])-(1.5*phi12_2[i,j]*nx_top[i,j]*ny_top[i,j])-(1.5*phi22_2[i
    ,j]*ny_top[i,j]*ny_top[i,j])) * f[i,j]
802             phi12_1w[i,j] = C_1w * (diss_RANS_2d[i,j]/k_RANS_2d[i,j]) * ((1.5*uu2d[i,j]*
    nx_top[i,j]*ny_top[i,j])-(1.5*uv2d[i,j]*ny_top[i,j]*ny_top[i,j])-(1.5*uv2d[i,j]*
```

```
    nx_top[i,j]*nx_top[i,j])-(1.5*vv2d[i,j]*nx_top[i,j]*ny_top[i,j])) * f[i,j]
803             phi12_2w[i,j] = C_2w * (diss_RANS_2d[i,j]/k_RANS_2d[i,j]) * ((1.5*phi11_2[i,
    j]*nx_top[i,j]*ny_top[i,j])-(1.5*phi12_2[i,j]*ny_top[i,j]*ny_top[i,j])-(1.5*phi12_2[
    i,j]*nx_top[i,j]*nx_top[i,j])-(1.5*phi22_2[i,j]*nx_top[i,j]*ny_top[i,j])) * f[i,j]
804
805 phi21_1w = phi12_1w
806 phi21_2w = phi12_2w
807
808 PS_11 = phi11_1 + phi11_2 + phi11_1w + phi11_2w
809 PS_12 = phi12_1 + phi12_2 + phi12_1w + phi12_2w
810 PS_22 = phi22_1 + phi22_2 + phi22_1w + phi22_2w
811 PS_21 = phi21_1 + phi21_2 + phi21_1w + phi21_2w
812
813 # Take the terms in stress equation of 11 and 12 for comparision along line i=5
814 gl3 = 10
815 fig12,axes = plt.subplots(nrows=2,ncols=1,constrained_layout=True)
816 axes[0].plot(yplus[gl3,0:81],P_11[gl3,0:81],label=r'Production term - $P_{11}$')
817 axes[1].plot(yplus[gl3,0:81],PS_11[gl3,0:81],label=r'Pressure Strain Term - $\Phi_{11}$
    ')
818 axes[0].plot(yplus[gl3,0:81],-diss_11[gl3,0:81],label=r'Dissipation - $\epsilon_{11}$')
819 axes[0].plot(yplus[gl3,0:81],diff_11[gl3,0:81],label=r'Turbulent Diffusion - $(D_{11})_{
    t}$')
820 axes[1].plot(yplus[gl3,0:81],diffvisc_11[gl3,0:81],label=r'Viscous Diffusion - $(D_{11})
    _{v}$')
821 axes[1].plot(yplus[gl3,0:81],LHS_11[gl3,0:81],label=r'LHS')
822 #axes[1].plot(yplus[gl3,0:81],P_12[gl3,0:81],label=r'Production term - $P_12$')
823 #axes[1].plot(yplus[gl3,0:81],PS_12[gl3,0:81],label=r'Pressure Strain Term - $\Phi_{12}$
    ')
824 #axes[1].plot(yplus[gl3,0:81],-diss_12[gl3,0:81],label=r'Dissipation - $\epsilon_{12}$')
825 #axes[1].plot(yplus[gl3,0:81],diff_12[gl3,0:81],label=r'Turbulent Diffusion - $(D_{12})_
    {t}$')
826 #axes[1].plot(yplus[gl3,0:81],diffvisc_12[gl3,0:81],label=r'Viscous Diffusion - $(D_
    {12})_{v}$')
827 #axes[1].plot(yplus[gl3,0:81],LHS_12[gl3,0:81],label=r'LHS')
828 for ax in axes:
829     ax.tick_params(labelsize='x-large')
830     ax.xaxis.set_minor_locator(AutoMinorLocator())
831     ax.yaxis.set_minor_locator(AutoMinorLocator())
832     ax.grid(True, linestyle='-.')
833     ax.legend(fontsize=14,loc='best')
834     ax.set_xlabel('.', color=(0, 0, 0, 0),fontsize=20)
835     ax.set_ylabel('.', color=(0, 0, 0, 0),fontsize=20)
836 fig12.add_subplot(111, frameon=False)
837 plt.tick_params(labelcolor="none", top=False, bottom=False, left=False, right=False)
838 plt.xlabel(r'$Y^+$',fontsize=18)
839 #plt.ylabel(r'$P^{k}$',fontsize=18)
840 fig12.suptitle(r'Terms in $\overline{{V_{1}^{\prime2}}}$ Equation at X = '+str(np.around
    (x2d[gl3,0],3))+' m (Lower Domain)', fontsize=20)
841
842 fig12a,axes = plt.subplots(nrows=2,ncols=1,constrained_layout=True)
843 #axes[0].plot(yplus[gl3,0:81],P_11[gl3,0:81],label=r'Production term - $P_{11}$')
844 #axes[0].plot(yplus[gl3,0:81],PS_11[gl3,0:81],label=r'Pressure Strain Term - $\Phi_{11}$
    ')
845 #axes[0].plot(yplus[gl3,0:81],-diss_11[gl3,0:81],label=r'Dissipation - $\epsilon_{11}$')
846 #axes[0].plot(yplus[gl3,0:81],diff_11[gl3,0:81],label=r'Turbulent Diffusion - $(D_{11})_
    {t}$')
847 #axes[0].plot(yplus[gl3,0:81],diffvisc_11[gl3,0:81],label=r'Viscous Diffusion - $(D_
    {11})_{v}$')
848 #axes[0].plot(yplus[gl3,0:81],LHS_11[gl3,0:81],label=r'LHS')
849 axes[0].plot(yplus[gl3,0:81],P_12[gl3,0:81],label=r'Production term - $P_12$')
850 axes[1].plot(yplus[gl3,0:81],PS_12[gl3,0:81],label=r'Pressure Strain Term - $\Phi_{12}$
    ')
851 axes[0].plot(yplus[gl3,0:81],-diss_12[gl3,0:81],label=r'Dissipation - $\epsilon_{12}$')
852 axes[0].plot(yplus[gl3,0:81],diff_12[gl3,0:81],label=r'Turbulent Diffusion - $(D_{12})_{
    t}$')
853 axes[1].plot(yplus[gl3,0:81],diffvisc_12[gl3,0:81],label=r'Viscous Diffusion - $(D_{12})
```

```
    _{v}$')
axes[1].plot(yplus[gl3,0:81],LHS_12[gl3,0:81],label=r'LHS')
for ax in axes:
    ax.tick_params(labelsize='x-large')
    ax.xaxis.set_minor_locator(AutoMinorLocator())
    ax.yaxis.set_minor_locator(AutoMinorLocator())
    ax.grid(True, linestyle='-.')
    ax.legend(fontsize=14,loc='best')
    ax.set_xlabel('.', color=(0, 0, 0, 0),fontsize=20)
    ax.set_ylabel('.', color=(0, 0, 0, 0),fontsize=20)
fig12a.add_subplot(111, frameon=False)
plt.tick_params(labelcolor="none", top=False, bottom=False, left=False, right=False)
plt.xlabel(r'$Y^+$',fontsize=18)
#plt.ylabel(r'$P^{k}$',fontsize=18)
fig12a.suptitle(r'Terms in $\overline{{V_{1}^{\prime}}{V_{2}^{\prime}}}$ Equation at X =
    '+str(np.around(x2d[gl3,0],3))+' m (Lower Domain)', fontsize=20)


fig12b,axes = plt.subplots(nrows=2,ncols=1,constrained_layout=True)
axes[0].plot(yplus[gl3,82:],P_11[gl3,82:],label=r'Production term - $P_11$')
axes[1].plot(yplus[gl3,82:],PS_11[gl3,82:],label=r'Pressure Strain Term')
axes[0].plot(yplus[gl3,82:],-diss_11[gl3,82:],label=r'Dissipation')
axes[0].plot(yplus[gl3,82:],diff_11[gl3,82:],label=r'Turbulent Diffusion')
axes[1].plot(yplus[gl3,82:],diffvisc_11[gl3,82:],label=r'Viscous Diffusion')
axes[1].plot(yplus[gl3,82:],LHS_11[gl3,82:],label=r'LHS')
#axes[1].plot(yplus[gl3,82:],P_12[gl3,82:],label=r'Production term - $P_12$')
#axes[1].plot(yplus[gl3,82:],PS_12[gl3,82:],label=r'Pressure Strain Term')
#axes[1].plot(yplus[gl3,82:],-diss_12[gl3,82:],label=r'Dissipation')
#axes[1].plot(yplus[gl3,82:],diff_12[gl3,82:],label=r'Turbulent Diffusion')
#axes[1].plot(yplus[gl3,82:],diffvisc_12[gl3,82:],label=r'Viscous Diffusion')
#axes[1].plot(yplus[gl3,82:],LHS_12[gl3,82:],label=r'LHS')
for ax in axes:
    ax.tick_params(labelsize='x-large')
    ax.xaxis.set_minor_locator(AutoMinorLocator())
    ax.yaxis.set_minor_locator(AutoMinorLocator())
    ax.grid(True, linestyle='-.')
    ax.legend(fontsize=14,loc='best')
    ax.set_xlabel('.', color=(0, 0, 0, 0),fontsize=20)
    ax.set_ylabel('.', color=(0, 0, 0, 0),fontsize=20)
fig12b.add_subplot(111, frameon=False)
plt.tick_params(labelcolor="none", top=False, bottom=False, left=False, right=False)
plt.xlabel(r'$Y^+$',fontsize=18)
#plt.ylabel(r'$P^{k}$',fontsize=18)
fig12b.suptitle(r'Terms in $\overline{{V_{1}^{\prime2}}}$ Equation at X = '+str(np.
    around(x2d[gl3,0],3))+' m (Upper Domain)', fontsize=20)

fig12c,axes = plt.subplots(nrows=2,ncols=1,constrained_layout=True)
#axes[0].plot(yplus[gl3,82:],P_11[gl3,82:],label=r'Production term - $P_11$')
#axes[0].plot(yplus[gl3,82:],PS_11[gl3,82:],label=r'Pressure Strain Term')
#axes[0].plot(yplus[gl3,82:],-diss_11[gl3,82:],label=r'Dissipation')
#axes[0].plot(yplus[gl3,82:],diff_11[gl3,82:],label=r'Turbulent Diffusion')
#axes[0].plot(yplus[gl3,82:],diffvisc_11[gl3,82:],label=r'Viscous Diffusion')
#axes[0].plot(yplus[gl3,82:],LHS_11[gl3,82:],label=r'LHS')
axes[0].plot(yplus[gl3,82:],P_12[gl3,82:],label=r'Production term - $P_12$')
axes[1].plot(yplus[gl3,82:],PS_12[gl3,82:],label=r'Pressure Strain Term')
axes[0].plot(yplus[gl3,82:],-diss_12[gl3,82:],label=r'Dissipation')
axes[0].plot(yplus[gl3,82:],diff_12[gl3,82:],label=r'Turbulent Diffusion')
axes[1].plot(yplus[gl3,82:],diffvisc_12[gl3,82:],label=r'Viscous Diffusion')
axes[1].plot(yplus[gl3,82:],LHS_12[gl3,82:],label=r'LHS')
for ax in axes:
    ax.tick_params(labelsize='x-large')
    ax.xaxis.set_minor_locator(AutoMinorLocator())
    ax.yaxis.set_minor_locator(AutoMinorLocator())
    ax.grid(True, linestyle='-.')
    ax.legend(fontsize=14,loc='best')
    ax.set_xlabel('.', color=(0, 0, 0, 0),fontsize=20)
```

```
    ax.set_ylabel('.', color=(0, 0, 0, 0),fontsize=20)
fig12c.add_subplot(111, frameon=False)
plt.tick_params(labelcolor="none", top=False, bottom=False, left=False, right=False)
plt.xlabel(r'$Y^+$',fontsize=18)
#plt.ylabel(r'$P^{k}$',fontsize=18)
fig12c.suptitle(r'Terms in $\overline{{V_{1}^{\prime}}{V_{2}^{\prime}}}$ Equation at X =
    '+str(np.around(x2d[gl3,0],3))+' m (Upper Domain)', fontsize=20)

#%% Assignment 1.8 - Reynold's Stresses using Boussinesq Assumption
# Calculating Reynolds Stresses
visc_turb = C_mu * ((k_RANS_2d**2)/diss_RANS_2d)
S11 = 0.5 * (dub_dx + dub_dx)
S12 = 0.5 * (dub_dy + dvb_dx)
S22 = 0.5 * (dvb_dy + dvb_dy)

uu2d_B = (-2 * visc_turb * S11) + ((2/3) * k_RANS_2d)
vv2d_B = (-2 * visc_turb * S22) + ((2/3) * k_RANS_2d)
uv2d_B = (-2 * visc_turb * S12)

# Comparision of Eddy-viscosity and Reynolds stress - Contour plots
vmax1 = np.max(uu2d_B-uu2d)
vmax2 = np.max(vv2d_B-vv2d)
vmax3 = np.max(uv2d_B-uv2d)
v1 = np.linspace(0,vmax1,100)
v2 = np.linspace(0,vmax2,100)
v3 = np.linspace(0,vmax3,100)
fig13,axes = plt.subplots(3,2,constrained_layout=True)
axes = axes.reshape(-1)
C1 = axes[0].contourf(x2d,y2d,uu2d_B-uu2d,levels=v1,cmap='binary',extend='min')
C1.cmap.set_under('papayawhip')
C2 = axes[1].contourf(x2d,y2d,uu2d_B,100,cmap='jet')
cb1 = fig13.colorbar(C1,ax=axes[0])
cb2 = fig13.colorbar(C2,ax=axes[1])
cb1.ax.tick_params(labelsize=16)
cb2.ax.tick_params(labelsize=16)
axes[0].set_title(r'Eddy Viscosity Stress - Reynolds Stress ($\overline{{V_{1}^{\prime2
    }}}$)', fontsize=20)
axes[1].set_title(r'Eddy Viscosity Stress - $\overline{{V_{1}^{\prime2}}}$', fontsize
    =20)
C3 = axes[2].contourf(x2d,y2d,vv2d_B-vv2d,levels=v2,cmap='binary',extend='min')
C3.cmap.set_under('papayawhip')
C4 = axes[3].contourf(x2d,y2d,vv2d_B,100,cmap='jet')
cb3 = fig13.colorbar(C3,ax=axes[2])
cb4 = fig13.colorbar(C4,ax=axes[3])
cb3.ax.tick_params(labelsize=16)
cb4.ax.tick_params(labelsize=16)
axes[2].set_title(r'Eddy Viscosity Stress - Reynolds Stress ($\overline{{V_{2}^{\prime2
    }}}$)', fontsize=20)
axes[3].set_title(r'Eddy Viscosity Stress - $\overline{{V_{2}^{\prime2}}}$', fontsize
    =20)
C5 = axes[4].contourf(x2d,y2d,uv2d_B-uv2d,levels=v3,cmap='binary',extend='min')
C5.cmap.set_under('papayawhip')
C6 = axes[5].contourf(x2d,y2d,uv2d_B,100,cmap='jet')
cb5 = fig13.colorbar(C5,ax=axes[4])
cb6 = fig13.colorbar(C6,ax=axes[5])
cb5.ax.tick_params(labelsize=16)
cb6.ax.tick_params(labelsize=16)
axes[4].set_title(r'Eddy Viscosity Stress - Reynolds Stress ($\overline{{V_{1}^{\prime
    }}{V_{2}^{\prime}}}$)', fontsize=20)
axes[5].set_title(r'Eddy Viscosity Stress - $\overline{{V_{1}^{\prime}}{V_{2}^{\prime}}}
    $', fontsize=20)

# Comparision of Eddy-viscosity and Reynolds stress - Line plots
fig14,axes = plt.subplots(3,2,constrained_layout=True)
axes = axes.reshape(-1)
gl4 = 0
```

```
gl5 = 100
axes[0].plot(yplus[gl4,:81],uu2d[gl4,:81],label='Reynolds Stress')
axes[0].plot(yplus[gl4,:81],uu2d_B[gl4,:81],label='Eddy Viscosity Stress')
axes[0].set_title(r'$\overline{{V_{1}^{\prime2}}}$ at X = '+str(np.around(x2d[gl4,0],3))
    +' m',fontsize=16)
axes[1].plot(yplus[gl5,:81],uu2d[gl5,:81],label='Reynolds Stress')
axes[1].plot(yplus[gl5,:81],uu2d_B[gl5,:81],label='Eddy Viscosity Stress')
axes[1].set_title(r'$\overline{{V_{1}^{\prime2}}}$ at X = '+str(np.around(x2d[gl5,0],3))
    +' m',fontsize=16)
axes[2].plot(yplus[gl4,:81],vv2d[gl4,:81],label='Reynolds Stress')
axes[2].plot(yplus[gl4,:81],vv2d_B[gl4,:81],label='Eddy Viscosity Stress')
axes[2].set_title(r'$\overline{{V_{2}^{\prime2}}}$ at X = '+str(np.around(x2d[gl4,0],3))
    +' m',fontsize=16)
axes[3].plot(yplus[gl5,:81],vv2d[gl5,:81],label='Reynolds Stress')
axes[3].plot(yplus[gl5,:81],vv2d_B[gl5,:81],label='Eddy Viscosity Stress')
axes[3].set_title(r'$\overline{{V_{2}^{\prime2}}}$ at X = '+str(np.around(x2d[gl5,0],3))
    +' m',fontsize=16)
axes[4].plot(yplus[gl4,:81],uv2d[gl4,:81],label='Reynolds Stress')
axes[4].plot(yplus[gl4,:81],uv2d_B[gl4,:81],label='Eddy Viscosity Stress')
axes[4].set_title(r'$\overline{{V_{1}^{\prime}}{V_{2}^{\prime}}}$ at X = '+str(np.
    around(x2d[gl4,0],3))+' m',fontsize=16)
axes[5].plot(yplus[gl5,:81],uv2d[gl5,:81],label='Reynolds Stress')
axes[5].plot(yplus[gl5,:81],uv2d_B[gl5,:81],label='Eddy Viscosity Stress')
axes[5].set_title(r'$\overline{{V_{1}^{\prime}}{V_{2}^{\prime}}}$ at X = '+str(np.
    around(x2d[gl5,0],3))+' m',fontsize=16)
for ax in axes:
    ax.tick_params(labelsize='x-large')
    ax.xaxis.set_minor_locator(AutoMinorLocator())
    ax.yaxis.set_minor_locator(AutoMinorLocator())
    ax.grid(True, linestyle='-.')
    ax.legend(fontsize=16,loc='best')
    ax.set_xlabel('.', color=(0, 0, 0, 0),fontsize=20)
    ax.set_ylabel('.', color=(0, 0, 0, 0),fontsize=20)
fig14.add_subplot(111, frameon=False)
plt.tick_params(labelcolor="none", top=False, bottom=False, left=False, right=False)
plt.ylabel("Values of Eddy viscosity and Reynolds stress",fontsize=18)
plt.xlabel(r"$Y^+$",fontsize=18)
fig14.suptitle('Comparison between Reynolds and Eddy viscosity stresses - $\overline{{V_
    {1}^{\prime2}}}$, $\overline{{V_{2}^{\prime2}}}$ and $\overline{{V_{1}^{\prime}}{V_
    {2}^{\prime}}}$ (Lower part of domain)',fontsize=24)

fig14a,axes = plt.subplots(3,2,constrained_layout=True)
axes = axes.reshape(-1)
axes[0].plot(yplus[gl4,82:],uu2d[gl4,82:],label='Reynolds Stress')
axes[0].plot(yplus[gl4,82:],uu2d_B[gl4,82:],label='Eddy Viscosity Stress')
axes[0].set_title(r'$\overline{{V_{1}^{\prime2}}}$ at X = '+str(np.around(x2d[gl4,0],3))
    +' m',fontsize=16)
axes[1].plot(yplus[gl5,82:],uu2d[gl5,82:],label='Reynolds Stress')
axes[1].plot(yplus[gl5,82:],uu2d_B[gl5,82:],label='Eddy Viscosity Stress')
axes[1].set_title(r'$\overline{{V_{1}^{\prime2}}}$ at X = '+str(np.around(x2d[gl5,0],3))
    +' m',fontsize=16)
axes[2].plot(yplus[gl4,82:],vv2d[gl4,82:],label='Reynolds Stress')
axes[2].plot(yplus[gl4,82:],vv2d_B[gl4,82:],label='Eddy Viscosity Stress')
axes[2].set_title(r'$\overline{{V_{2}^{\prime2}}}$ at X = '+str(np.around(x2d[gl4,0],3))
    +' m',fontsize=16)
axes[3].plot(yplus[gl5,82:],vv2d[gl5,82:],label='Reynolds Stress')
axes[3].plot(yplus[gl5,82:],vv2d_B[gl5,82:],label='Eddy Viscosity Stress')
axes[3].set_title(r'$\overline{{V_{2}^{\prime2}}}$ at X = '+str(np.around(x2d[gl5,0],3))
    +' m',fontsize=16)
axes[4].plot(yplus[gl4,82:],uv2d[gl4,82:],label='Reynolds Stress')
axes[4].plot(yplus[gl4,82:],uv2d_B[gl4,82:],label='Eddy Viscosity Stress')
axes[4].set_title(r'$\overline{{V_{1}^{\prime}}{V_{2}^{\prime}}}$ at X = '+str(np.
    around(x2d[gl4,0],3))+' m',fontsize=16)
axes[5].plot(yplus[gl5,82:],uv2d[gl5,82:],label='Reynolds Stress')
axes[5].plot(yplus[gl5,82:],uv2d_B[gl5,82:],label='Eddy Viscosity Stress')
axes[5].set_title(r'$\overline{{V_{1}^{\prime}}{V_{2}^{\prime}}}$ at X = '+str(np.
```

```
    around(x2d[gl5,0],3))+' m',fontsize=16)
for ax in axes:
    ax.tick_params(labelsize='x-large')
    ax.xaxis.set_minor_locator(AutoMinorLocator())
    ax.yaxis.set_minor_locator(AutoMinorLocator())
    ax.grid(True, linestyle='-.')
    ax.legend(fontsize=16,loc='best')
    ax.set_xlabel('.', color=(0, 0, 0, 0),fontsize=20)
    ax.set_ylabel('.', color=(0, 0, 0, 0),fontsize=20)
fig14a.add_subplot(111, frameon=False)
plt.tick_params(labelcolor="none", top=False, bottom=False, left=False, right=False)
plt.xlabel("Values of Eddy viscosity and Reynolds stress",fontsize=18)
plt.ylabel(r"$Y^+$",fontsize=18)
fig14a.suptitle('Comparison between Reynolds and Eddy viscosity stresses - $\overline{{
    V_{1}^{\prime2}}}$, $\overline{{V_{2}^{\prime2}}}$ and $\overline{{V_{1}^{\prime}}{
    V_{2}^{\prime}}}}$ (Upper part of domain)',fontsize=24)

# Production term of Reynolds stress - Boussinesq assumption
Pk_B = 2 * visc_turb * (S11*S11 + 2*S12*S12 + S22*S22)
#Pk_BB = 2 * visc_turb_modified * (S11*S11 + 2*S12*S12 + S22*S22)
vmin = np.min(Pk_B)
vmax = 3
v = np.linspace(vmin,vmax,100)
fig15,axes = plt.subplots(2,1,constrained_layout=True)
C1 = axes[0].contourf(x2d,y2d,Pk_B,levels=v,extend='max',cmap='jet')
C2 = axes[1].contourf(x2d,y2d,Pk_B,100,cmap='jet')
axes[1].set_ylim(2.5,y2d[0,-1])
cb1 = fig15.colorbar(C1,ax=axes[0])
cb2 = fig15.colorbar(C2,ax=axes[1])
cb1.ax.tick_params(labelsize=16)
cb2.ax.tick_params(labelsize=16)
axes[0].set_title(r'Production term - Boussinesq Assumption', fontsize=20)
axes[1].set_title(r'Production term - Boussinesq Assumption (Zoomed near top wall)',
    fontsize=20)

fig15a,axes = plt.subplots(1,1,constrained_layout=True)
C1 = axes.contourf(x2d,y2d,Pk_BB,100,cmap='jet')
cb1 = fig15a.colorbar(C1,ax=axes)
cb1.ax.tick_params(labelsize=16)
axes.set_title(r'Production term - Boussinesq Assumption - After Limiter Function',
    fontsize=20)

#%% Assignment 1.9 - Eigenvectors of Reynolds Stress and Strain rate tensor
EVx_SRT = np.zeros((ni,nj))
EVy_SRT = np.zeros((ni,nj))
EVx_RS = np.zeros((ni,nj))
EVy_RS = np.zeros((ni,nj))
EigenValue1 = np.zeros((ni,nj))
EigenValue2 = np.zeros((ni,nj))

for i in range(ni):
    for j in range(nj):
        value1,vector1 = np.linalg.eig(np.array([[S11[i,j],S12[i,j]],[S12[i,j],S22[i,j
    ]]]))
        EVx_SRT[i,j]=vector1[0,0]
        EVy_SRT[i,j]=vector1[1,0]
        EigenValue1[i,j] = value1[0]
        EigenValue2[i,j] = value1[1]
        value2,vector2 = np.linalg.eig(np.array([[uu2d_B[i,j],uv2d_B[i,j]],[uv2d_B[i,j],
    vv2d_B[i,j]]]))
        EVx_RS[i,j]=vector2[0,0]
        EVy_RS[i,j]=vector2[1,0]

k=5
fig16 = plt.figure()
axes = fig16.add_axes([0.1,0.1,0.75,0.75])
```

```
q1 = axes.quiver(x2d[::k,::k],y2d[::k,::k],-EVx_SRT[::k,::k],-EVy_SRT[::k,::k],color='blue',pivot='mid',width=0.001,scale=30,headwidth=5,headlength=5)
q2 = axes.quiver(x2d[::k,::k],y2d[::k,::k],EVx_RS[::k,::k],EVy_RS[::k,::k],color='red',pivot='mid')
axes.quiverkey(q1, X=1.1, Y=0.6, U=3,label='Strain Rate Tensor', labelpos='S', fontproperties={'weight':'bold','size':16})
axes.quiverkey(q2, X=1.1, Y=0.4, U=5,label='Reynolds Stress', labelpos='S', fontproperties={'weight':'bold','size':16})
axes.set_title('Eigen vectors for Strain rate tensor and Reynolds stress',fontsize=20)
axes.set_xlabel('X (m)',fontsize=16)
axes.set_ylabel('Y (m)',fontsize=16)

#%% Assignment 1.10 -
lambda1 = np.zeros((ni,nj))
visc_turb_modified = np.zeros((ni,nj))

for i in range(ni):
    for j in range(nj):
        if EigenValue1[i,j]>=EigenValue2[i,j]:
            lambda1[i,j] = EigenValue1[i,j]
        else:
            lambda1[i,j] = EigenValue2[i,j]

for i in range(ni):
    for j in range(nj):
        if visc_turb[i,j]>=((k_RANS_2d[i,j]/3)*(1/np.abs(lambda1[i,j]))):
            visc_turb_modified[i,j] = ((k_RANS_2d[i,j]/3)*(1/np.abs(lambda1[i,j])))
        else:
            visc_turb_modified[i,j] = visc_turb[i,j]

uu2d_B_lim = (-2 * visc_turb_modified * S11) + ((2/3) * k_RANS_2d)
vv2d_B_lim = (-2 * visc_turb_modified * S22) + ((2/3) * k_RANS_2d)
uv2d_B_lim = (-2 * visc_turb_modified * S12)

fig17 = plt.figure()
ax1 = plt.subplot2grid((2,2), (0,0), colspan=2)
ax2 = plt.subplot2grid((2,2), (1,0))
ax3 = plt.subplot2grid((2,2), (1,1))
cax1 = ax1.contourf(x2d,y2d,visc_turb-visc_turb_modified,cmap='jet',levels=50)
cax2 = ax2.contourf(x2d,y2d,visc_turb,cmap='jet',levels=50)
cax3 = ax3.contourf(x2d,y2d,visc_turb_modified,cmap='jet',levels=50)
for ax in [ax1,ax2,ax3]:
    ax.set_xlabel('X (m)')
    ax.set_ylabel('Y (m)')
cb1 = fig17.colorbar(cax1,ax=ax1,fraction=0.1)
cb1.ax.tick_params(labelsize=16)
cb2 = fig17.colorbar(cax2,ax=ax2,fraction=0.1)
cb2.ax.tick_params(labelsize=16)
cb3 = fig17.colorbar(cax3,ax=ax3,fraction=0.1)
cb3.ax.tick_params(labelsize=16)
plt.tight_layout()
ax1.set_title(r'Regions affected by limiter function', fontsize=24)
ax2.set_title(r'Turbulent Viscosity ($\nu_{t}$) before limiter function', fontsize=20)
ax3.set_title(r'Turbulent Viscosity ($\nu_{t}$) after limiter function', fontsize=20)
for ax in [ax1,ax2,ax3]:
    ax.tick_params(labelsize='x-large')

fig18,axes = plt.subplots(2,2,constrained_layout=True)
axes = axes.reshape(-1)
vmax1 = np.max(uu2d_B)
vmax2 = np.max(uu2d_B_lim)
vmax3 = np.max(vv2d_B)
vmax4 = np.max(vv2d_B_lim)
vmin1 = np.min(uu2d_B)
vmin2 = np.min(uu2d_B_lim)
vmin3 = np.min(vv2d_B)
```

```
vmin4 = np.min(vv2d_B_lim)
v1 = np.linspace(0,vmax1,100)
v2 = np.linspace(vmin2,vmax2,100)
v3 = np.linspace(0,vmax3,100)
v4 = np.linspace(vmin4,vmax4,100)
cax1 = axes[0].contourf(x2d,y2d,uu2d_B,levels=v1,cmap='Blues',extend='min')
cax1.cmap.set_under('black')
cax2 = axes[1].contourf(x2d,y2d,uu2d_B_lim,levels=v2,cmap='Blues')
cax3 = axes[2].contourf(x2d,y2d,vv2d_B,levels=v3,cmap='Blues',extend='min')
cax3.cmap.set_under('black')
axes[2].set_xlim(6.5,9)
axes[2].set_ylim(0,1.5)
cax4 = axes[3].contourf(x2d,y2d,vv2d_B_lim,levels=v4,cmap='Blues')
axes[3].set_xlim(6.5,9)
axes[3].set_ylim(0,1.5)
axes[0].set_title(r'Normal Stress ($\overline{{V_{1}^{\prime2}}}$) before limiter
    function',fontsize=18)
axes[1].set_title(r'Normal Stress ($\overline{{V_{1}^{\prime2}}}$) after limiter
    function',fontsize=18)
axes[2].set_title(r'Normal Stress ($\overline{{V_{2}^{\prime2}}}$) before limiter
    function',fontsize=18)
axes[3].set_title(r'Normal Stress ($\overline{{V_{2}^{\prime2}}}$) after limiter
    function',fontsize=18)
for axe,cax in zip(axes,[cax1,cax2,cax3,cax4]):
    axe.set_xlabel('X (m)')
    axe.set_ylabel('Y (m)')
    fig18.colorbar(cax,ax=axe)

fig19,axes = plt.subplots(2,2,constrained_layout=True)
axes = axes.reshape(-1)
axes[0].plot(yplus[gl1,:81],visc_turb[gl1,:81],label='Before Limiting Function')
axes[0].plot(yplus[gl1,:81],visc_turb_modified[gl1,:81],label='Before Limiting Function'
    )
axes[0].set_title('Turbulent viscosity in Lower domain at X = ' + str(np.around(x2d[gl1
    ,0],3)) + ' m',fontsize=20)
axes[1].plot(yplus[gl2,:81],visc_turb[gl2,:81],label='Before Limiting Function')
axes[1].plot(yplus[gl2,:81],visc_turb_modified[gl2,:81],label='Before Limiting Function'
    )
axes[1].set_title('Turbulent viscosity in Lower domain at X = ' + str(np.around(x2d[gl2
    ,0],3)) + ' m',fontsize=20)
axes[2].plot(yplus[gl1,82:],visc_turb[gl1,82:],label='Before Limiting Function')
axes[2].plot(yplus[gl1,82:],visc_turb_modified[gl1,82:],label='Before Limiting Function'
    )
axes[2].set_title('Turbulent viscosity in Upper domain at X = ' + str(np.around(x2d[gl1
    ,0],3)) + ' m',fontsize=20)
axes[3].plot(yplus[gl2,82:],visc_turb[gl2,82:],label='Before Limiting Function')
axes[3].plot(yplus[gl2,82:],visc_turb_modified[gl2,82:],label='Before Limiting Function'
    )
axes[3].set_title('Turbulent viscosity in Upper domain at X = ' + str(np.around(x2d[gl2
    ,0],3)) + ' m',fontsize=20)
for ax in axes:
    ax.tick_params(labelsize='x-large')
    ax.xaxis.set_minor_locator(AutoMinorLocator())
    ax.yaxis.set_minor_locator(AutoMinorLocator())
    ax.grid(True, linestyle='-.')
    ax.legend(fontsize=16,loc='best')
    ax.set_ylabel(r'Turbulent Viscosity ($\nu_{t}$)',fontsize=20)
    ax.set_xlabel(r'$Y^+$',fontsize=20)

fig20,axes = plt.subplots(1,1,constrained_layout=True)
cax1 = axes.contourf(x2d,y2d,visc_turb-visc_turb_modified,cmap='jet',levels=100)
axes.set_xlabel('X (m)')
axes.set_ylabel('Y (m)')
cb1 = fig20.colorbar(cax1,ax=axes,fraction=0.1)
cb1.ax.tick_params(labelsize=16)
axes.set_title(r'Regions affected by limiter function', fontsize=24)
```

```
'''
limiter = np.zeros((ni,nj))
for i in range(ni):
    for j in range(nj):
        limiter[i,j] = ((k_RANS_2d[i,j]/3)*(1/np.abs(lambda1[i,j])))
result = np.zeros((ni,nj))
for i in range(ni):
    for j in range(nj):
        if limiter[i,j]>=visc_turb[i,j]:
            result[i,j] = 1
        else:
            result[i,j] = 0
'''
```

```
# -*- coding: utf-8 -*-
"""
Created on Sun Apr 19 18:22:49 2020

@author: Amey
"""
def WallDistance(x2d,y2d,point_x,point_y):
    import numpy as np
    ni = len(x2d)
    dist_LW = np.zeros(ni)
    dist_UP = np.zeros(ni)
    for i in range(ni):
        dist_LW[i] = np.sqrt(((point_x-x2d[i,0])**2) + ((point_y-y2d[i,0])**2))
        dist_UP[i] = np.sqrt(((point_x-x2d[i,-1])**2) + ((point_y-y2d[i,-1])**2))

    if min(dist_LW)<=min(dist_UP):
        min_dist = min(dist_LW)
    else:
        min_dist = min(dist_UP)

    return min_dist
```

```
# -*- coding: utf-8 -*-
"""
Created on Sun Apr 19 20:06:05 2020

@author: Amey
"""
def WallNormal(xf2d,yf2d):
    import numpy as np
# Bottom Wall
    sx_bottom = np.diff(xf2d[:,0],axis=0)
    sy_bottom = np.diff(yf2d[:,0],axis=0)

    d=np.sqrt(sx_bottom**2+sy_bottom**2)
    nx_bottom=-sy_bottom/d
    ny_bottom=sx_bottom/d
# Top Wall
    sx_top = np.diff(xf2d[:,-1],axis=0)
    sy_top = np.diff(yf2d[:,-1],axis=0)

    d=np.sqrt(sx_top**2+sy_top**2)
    nx_top=-sy_top/d
    ny_top=sx_top/d

    nx_bottom = np.insert(nx_bottom,0,nx_bottom[0],axis=0)
    ny_bottom = np.insert(ny_bottom,0,ny_bottom[0],axis=0)
    nx_top = np.insert(nx_top,0,nx_top[0],axis=0)
    ny_top = np.insert(ny_top,0,ny_top[0],axis=0)
    nx_bottom = np.insert(nx_bottom,-1,nx_bottom[-1],axis=0)
    ny_bottom = np.insert(ny_bottom,-1,ny_bottom[-1],axis=0)
    nx_top = np.insert(nx_top,-1,nx_top[-1],axis=0)
```

```
31      ny_top = np.insert(ny_top,-1,ny_top[-1],axis=0)
32
33      return nx_bottom,ny_bottom,nx_top,ny_top
```