Assignment 3

GEOS 300

Term 1 (Autumn 2024)

University of British Columbia

Instructor:

Marysa Laguë, Assistant Professor, Department of Geography

Preamble:

In this exercise you will use a 30-min data-set measured above an extensively flat cotton field near Kettleman City, CA, USA^1. You will use two datasets:

df1 = wind200008021530.xls
df2 = turbulence200008021530.xls

The "wind" dataframe lists horizontal wind speeds u measured with cup-anemometers installed at six heights on a profile tower averaged over 30 minutes. Screen-level air temperature is also provided.

The "turbulence" dataframe contains longitudinal wind u, lateral wind v and vertical wind w measured every second over the same 30 minutes by a fast-response anemometer located at 6.4 m height.

For all questions assume neutral conditions and $z_d = 0$. Assume a pressure of 100 kPa.

Instructions: Please return your answers including all calculations, graphs and discussions in a well-structured report (PDF, either your jupyter notebook or a word/google doc saved as a pdf).

Label the report document with your name, your student number, the course and year.

Marks are indicated in square brackets. This assignment is worth 10% of your final grade.

^{1}http://www.eol.ucar.edu/rtf/projects/ebex2000/

Import relevant packages:

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import datetime
from datetime import datetime as dt
import time
```

Load the datasets:

In [2]:
```
### First, open the dataset:

# Import the data  - upload this file from Canvas and put it in the same fol
data_file = 'wind200008021530.xls'

df1 = pd.read_excel(data_file,skiprows = 6,)

# a few post processing steps:

# 1. Remove units from variable names & rename variables
names_and_units = list(df1.columns.values)
just_names = [x.split(" ",1)[0] for x in names_and_units]

# 2. replace the column names with our new names that don't have units
df1.columns = just_names
df1 = df1.rename(columns={"Horizontal": "U"})

# 3. add a variable called logHeight:

df1['logHeight'] = np.log(df1['Height'])

df1.head()
```

Out[2]:

	Height	U	logHeight
0	0.95	1.54	-0.051293
1	1.55	1.83	0.438255
2	2.35	2.00	0.854415
3	3.72	2.22	1.313724
4	6.15	2.50	1.816452

In [3]:
```
df1
```

Out [3]:

	Height	U	logHeight
0	0.95	1.54	-0.051293
1	1.55	1.83	0.438255
2	2.35	2.00	0.854415
3	3.72	2.22	1.313724
4	6.15	2.50	1.816452
5	9.05	2.72	2.202765

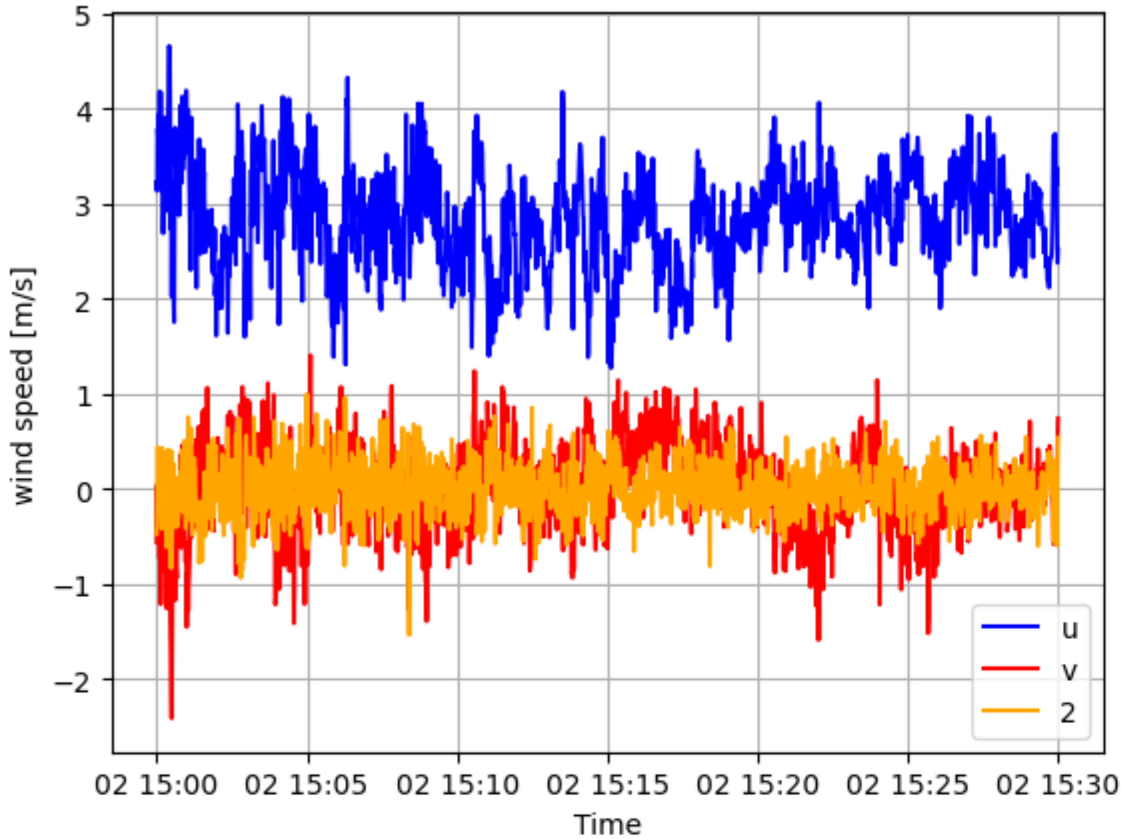
In [4]:

```
data_file = 'turbulence200008021530.xls'

dateparse = lambda x: dt.strptime(x, '%Y-%m-%d %H:%M:%S')

df2 = pd.read_excel(data_file,
                    # header = 0,
                    parse_dates=['Date'],
                    date_format='%Y-%m-%d %H:%M:%S',
                    index_col='Date')


# a few post processing steps:

# 1. add time from Date

df2['Time'] = df2.index.time

# 2. Remove units from variable names & rename variables
names_and_units = list(df2.columns.values)
just_names = [x.split(" ",1)[0] for x in names_and_units]

# 3. replace the column names with our new names that don't have units
df2.columns = just_names


# print out the top of the dataframe

df2.head()
```

Out [4]:

	u	v	w	Time
Date				
2000-08-02 15:00:00	3.2174	-0.5631	-0.4740	15:00:00
2000-08-02 15:00:01	3.1313	0.0400	0.0050	15:00:01
2000-08-02 15:00:02	3.7852	-0.4075	0.4388	15:00:02
2000-08-02 15:00:03	3.6670	-0.2518	0.0259	15:00:03
2000-08-02 15:00:04	3.9281	-0.1403	-0.2484	15:00:04

In [4]:
```
# visualize the data:


plt.plot(df2.index,df2.u,color="blue",label='u')
plt.plot(df2.index,df2.v,color="red",label='v')
plt.plot(df2.index,df2.w,color="orange",label='2')
plt.legend()
plt.grid(':')
plt.xlabel('Time')
plt.ylabel('wind speed [m/s]')

plt.show()
plt.close()
```



Question 1.

[1]

Did you use ai assistance with this assignment? If so, how?

Question 2:

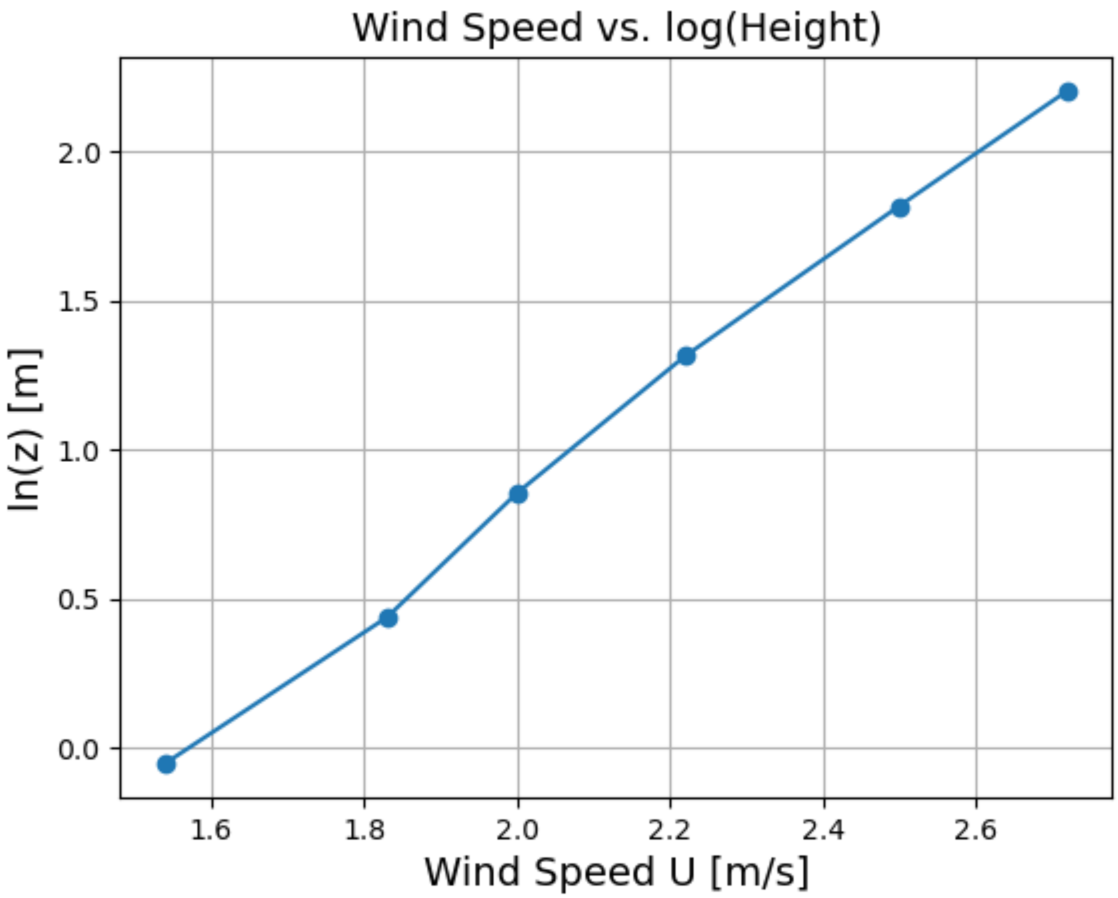
2. z_0 is the roughness length - the height at which a wind speed theoretically becomes zero in neutral atmospheric conditions. Estimate z_0 graphically from all measured values of the wind profile in df1 (the wind dataframe). You can either

use a spreadsheet/software (e.g. R, python, excel) or the semi-logarithmic paper provided on Canvas.

Note: If you solve this question using a semi-logarithmic paper, use a ruler and your graphical judgement (subjective) to create the best fit through the points. [6]

Rubric: [2] approach [3] graph [1] answer for z0

In [5]:
```
plt.plot(df1.U, df1.logHeight,marker='o')
plt.xlabel('Wind Speed U [m/s]',fontsize=14)
plt.ylabel('ln(z) [m]',fontsize=14)
plt.grid()
plt.title('Wind Speed vs. log(Height)',fontsize=14)

plt.show()
plt.close()
```



To estimate z_0, we need to calculate the y-intercept of the linear fit line that approximates the above. For a line $y=mx + b$, we have $x=u$, $y=ln(z)$, where m is the slope of the log-normal plot above, and $b=ln(z_0)$ is the intercept we're looking for. You can estimate this by hand by drawing the line back through the y axis on log-normal paper, or you can calculate the linear fit to the data. Both methods are fine for this assignment, and should give approximately similar answers. Answers within ±0.01 m are acceptable.

Solving for the y-intercept via linear regression gives us $ln(z_0)$. To get z_0 itself, we have: $b=ln(z_0) \Rightarrow z_0 = e^b$

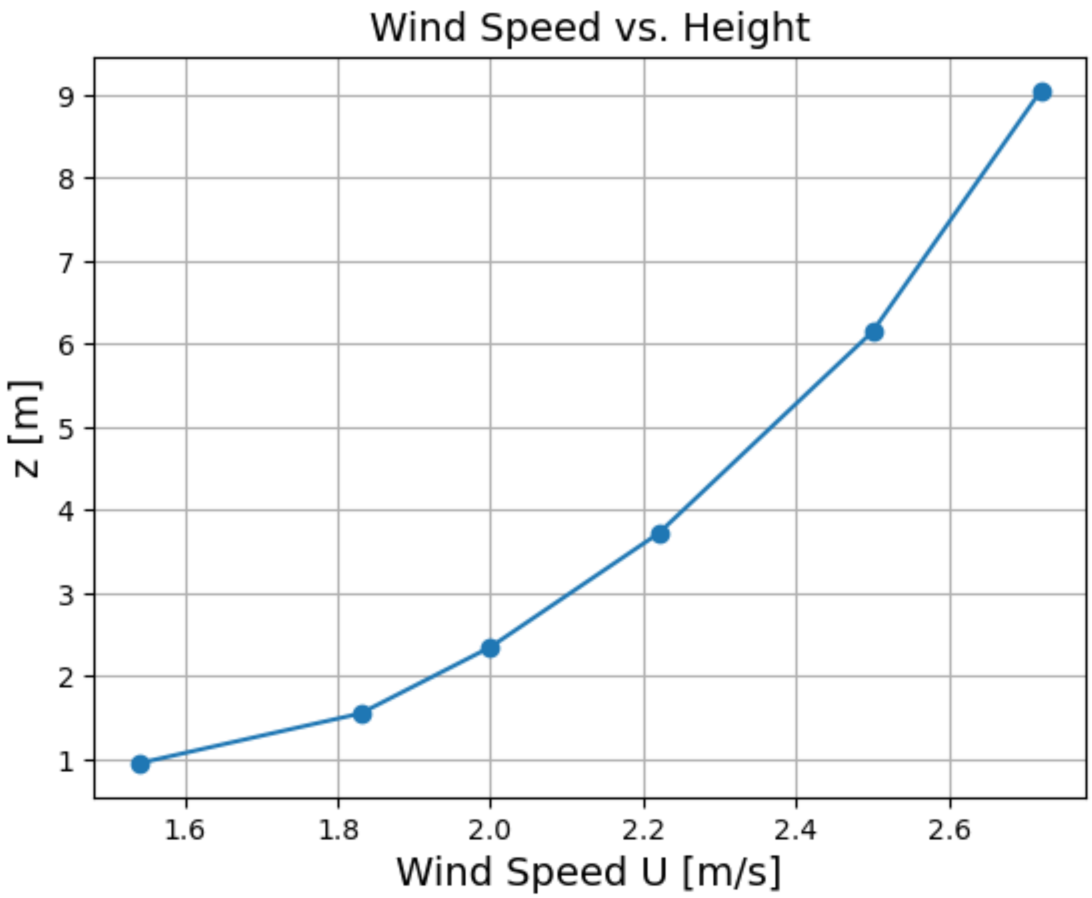
In [6]:
```
# In python, we can do a simple linear fit using a 1-degree polynomial
# using the numpy package. You can also use fancier statistical modeling
# packages; they should all give approximately the same answer for this data

m,b = np.polyfit(df1.U, df1.logHeight, 1)

z0 = np.exp(b)

print('The y-intercept z0 = %1.3f'%z0)
```

```
The y-intercept z0 = 0.047
```

In [ ]:

Solving for z_0 requires the use of $log(z)$ on the y axis. For demonstrational purposes, I'm also including the plot of u vs z, where you can see wind speed approaching zero near the surface (no-slip boundary condition), and wind speed increasing exponentially with height.

In [7]:
```
plt.plot(df1.U, df1.Height,marker='o')
plt.xlabel('Wind Speed U [m/s]',fontsize=14)
plt.ylabel('z [m]',fontsize=14)
plt.grid()
plt.title('Wind Speed vs. Height',fontsize=14)

plt.show()
plt.close()
```



In [ ]:

In [ ]:

In [ ]:

Question 3:

3. Based on the slope of the curve in Question 2, calculate the friction velocity u∗. How would the wind profile in Q2 look different if u* were larger than what you calculated? [6]

Rubric: [2] approach [1] u* value [3] discussion

In [8]:
```
# to answer this, we're going to use von-karman's constant k, and the fact t
# of a linear-log graph of wind vs log(z) is equal to k/u_star ... and we're
# for u_star, the friction velocity

# first, we have to calculate the slope of the graph:
m,b = np.polyfit(df1.U, df1.logHeight, 1)

print(m)

# then define von-karman's constant:
```

```
k = 0.41

# now calculate ustar:
u_star = k/m

print('The friction velocity ustar = %1.2f m/s'%u_star)
```

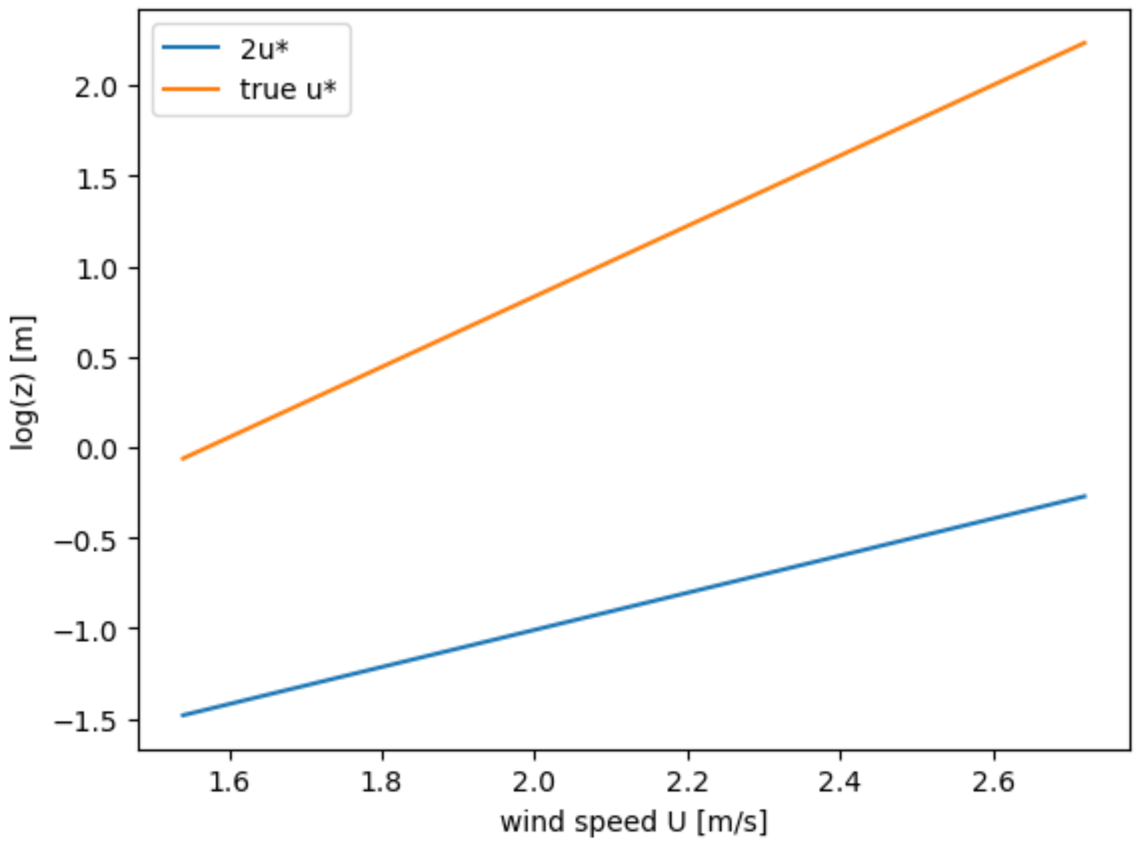
```
1.9460884981854518
The friction velocity ustar = 0.21 m/s
```

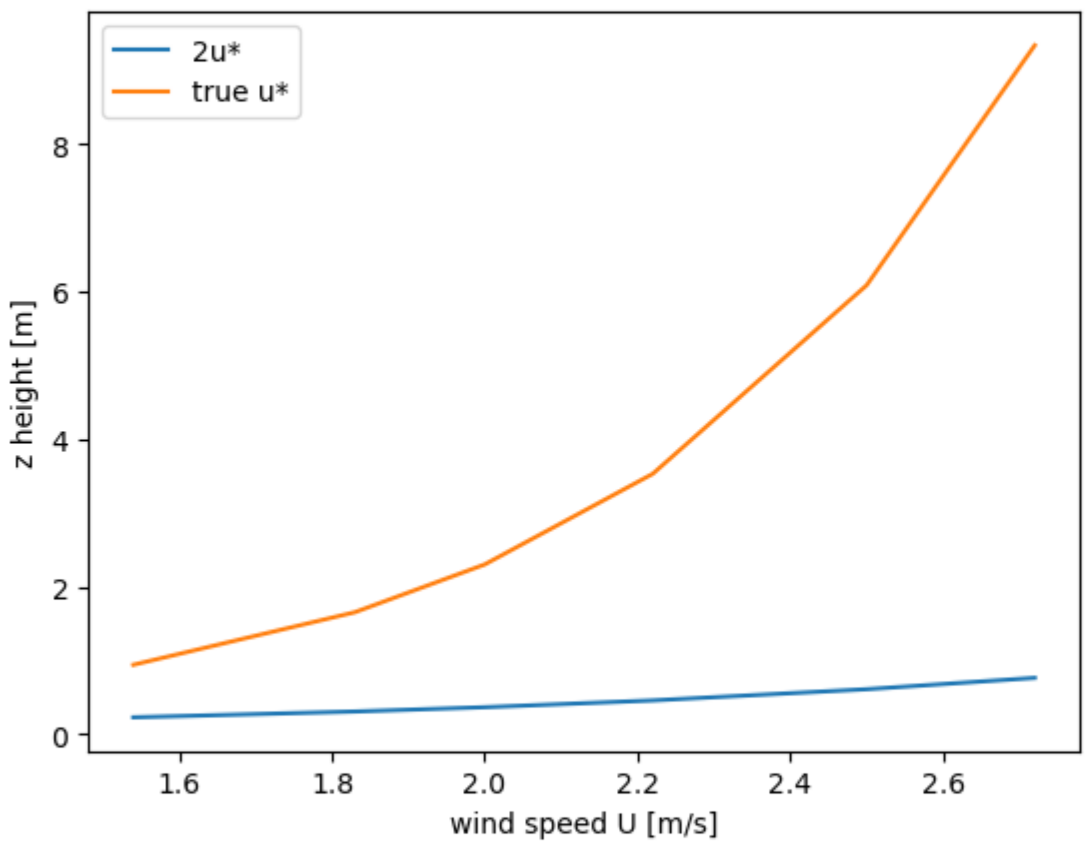
If the friction velocity were larger, since k is a constant, we could rearrange the equation $u*=k/m \Rightarrow m=k/u*$... if $u*$ were larger, then m the slope would be smaller [1], ie there would be less change in wind with height [1]. This conceptually is consisten with more surface friction leading to more drag and so winds aren't increasing as quickly above the surface as if the surface had less drag [1].

Students are NOT required to make a plot as part of the discussion, but I've plotted a log-linear plot with different slopes, and the exponential version of that plot that would correspond to the wind profile for the observed wind speeds assuming $u*$ vs 2x $u*$.

In [9]:

```
m_bigu = k/0.4
m_realu = k/u_star

plt.plot(df1.U, df1.U*m_bigu + b, label='2u*')
plt.plot(df1.U, df1.U*m_realu + b, label='true u*')
plt.legend()
plt.xlabel('wind speed U [m/s]')
plt.ylabel('log(z) [m]')

plt.show()
plt.close()

plt.plot(df1.U, np.exp(df1.U*m_bigu + b), label='2u*')
plt.plot(df1.U, np.exp(df1.U*m_realu + b), label='true u*')
plt.legend()
plt.xlabel('wind speed U [m/s]')
plt.ylabel('z height [m]')

plt.show()
plt.close()
```





In [10]:

```
9+1
```

Out[10]:

```
10
```

Question 4:

4. Estimate the eddy diffusivities for momentum K_M using the wind gradients Δu in the dataframe df1, between each layer:

(a) z= 0.95 and 1.55 m,

(b) z= 1.55 and 2.35 m,

(c) z = 2.35 and 3.72 m,

(d) z = 3.72 and 6.15 m, and

(e) z = 6.15 and 9.05 m.

Discuss how K_M changes with height, and explain why this happens. [6]

rubric: [2.5] values [.5 per value] [1.5] how KM changes with height [2] discussion

-1 for no units.

In [18]:
```
rho = 1.131 #kg/m3

tau0 = rho*(u_star**2)  #

print('tau = %1.2f N/m2'%tau0)

# using equation tau = rho * Km * du/dz, solve for Km
# can use mean u or instantaneous u then average.

dudz = np.diff(df1.U)/np.diff(df1.Height)
print('delta u / delta z:')
print(dudz)

Km = tau0 / (rho*dudz)

print('K_m:')
print(Km)

print('z:')
print(df1.Height.values)


# alternative method just using ustar:

print('Alternative method just using u*:')
Km = u_star**2/dudz

print('K_m:')
print(Km)

print('z:')
print(df1.Height.values)
```

```
tau = 0.05 N/m2
delta u / delta z:
[0.48333333 0.2125     0.16058394 0.11522634 0.07586207]
K_m:
[0.09183237 0.20887363 0.27640152 0.385204   0.58508351]
z:
[0.95 1.55 2.35 3.72 6.15 9.05]
Alternative method just using u*:
K_m:
[0.09183237 0.20887363 0.27640152 0.385204   0.58508351]
z:
[0.95 1.55 2.35 3.72 6.15 9.05]
```

a) K_m = 0.092 m2/s

b) K_m = 0.21 m2/s

c) K_m = 0.28 m2/s

d) K_m = 0.39 m2/s

e) K_m = 0.59 m2/s

K_M, the diffusivities, are increasing with height. This can be explained by an increasing average eddy size that mixes momentum more efficiently at higher layers (higher eddy diffusivity means more efficient exchange).

Also accept drawing that shows that eddy size is increasing with height.

Generally use the flux gradient relationship with K_M the eddy diffusivity for momentum: \begin{eqnarray} \tau_0 = \rho_a\, K_M \, \frac{\Delta \overline{u}}{\Delta z} \end{eqnarray} rearrange \begin{eqnarray} K_M = \frac{\tau_0}{\rho_a} \frac{\Delta z}{\Delta \overline{u}} = u_{\ast}^2 \frac{\Delta z}{\Delta \overline{u}} \end{eqnarray} Students can do calculation using either layer-individual $u_{\ast}$ or with 'average' $u_{\ast}$ (from fit above). Differences should be minor. Units of K_M are $\rm{m}^2\,\rm{s}^{-1}$.

In [ ]:

Question 5:

5. From the values in df1, calculate the aerodynamic resistance of the momentum flux r_{aM} for the layer from the surface to 9.05 m. How would an increased aerodynamic resistance alter the momentum flux? [4]

rubric: [1] approach [1] value [2] discussion

In [27]:

```
z0 = 0 # m
z1 = 9.05 # m

u0 = 0 # m/s (non-slip boundary condition)
u1 = df1.U.values[-1]
```

```
print('check wind at 9.05 m')
print(u1)

ram = rho * (u1 - u0) / tau0

print('ram = %1.2f s/m'%ram)
```

```
check wind at 9.05 m
2.72
ram = 61.28 s/m
```

Plug-in $z=0$ and $\overline{u}(0)=0$ as lower boundary condition:

\begin{eqnarray} r_{a_M} = \rho_a \frac{\Delta \overline{u}}{\tau_0} \end{eqnarray}

Or alternatively you could do the following, but you'd have to redo your K_M calculations based on the specific Δu and Δz for this question.

\begin{eqnarray} \rho_a \frac{\Delta \overline{u}}{r_{a_M}} = \rho_a \,K_M\, \frac{\Delta u}{\Delta z} \end{eqnarray}

rearrange:

\begin{eqnarray} r_{a_M} = \frac{\Delta z}{K_M} \end{eqnarray}

Units of r_{a_M} are $\textrm{s}\,\textrm{m}^{-1}$. Values are summarized in Tab. 5.1. [Total Marks: 2]

Discussion:

More aerodynamic resistance would mean smaller diffusivities K_M and smaller eddies [1], so a more aerodynacmially rough surface would be less efficient at mixing momentum, all else equal [1].

In [ ]:

```

```

Question 6:

From the turbulence data provided in the turbulence dataframe df2, calculate $\bar{u}$, $\bar{v}$, and $\bar{w}$ [4]

rubric: [2] approach [1] values [1] for equation.

-1 for missing units.

In [28]:

```
df2.head()
```

Out[28]:

Date	u	v	w	Time
2000-08-02 15:00:00	3.2174	-0.5631	-0.4740	15:00:00
2000-08-02 15:00:01	3.1313	0.0400	0.0050	15:00:01
2000-08-02 15:00:02	3.7852	-0.4075	0.4388	15:00:02
2000-08-02 15:00:03	3.6670	-0.2518	0.0259	15:00:03
2000-08-02 15:00:04	3.9281	-0.1403	-0.2484	15:00:04

In [ ]:

In [44]:

```
ubar = df2.u.mean()
vbar = df2.v.mean()
wbar = df2.w.mean()

print('using python averaging:')
print('mean u = %1.4f m/s'%ubar)
print('mean v = %1.4f m/s'%vbar)
print('mean w = %1.4f m/s'%wbar)

total_seconds = 1800 # seconds
print(total_seconds)
ubar_manual = df2.u.sum()/total_seconds
vbar_manual = df2.v.sum()/total_seconds
wbar_manual = df2.w.sum()/total_seconds


print('using manual averaging:')
print('mean u = %1.4f m/s'%ubar_manual)
print('mean v = %1.4f m/s'%vbar_manual)
print('mean w = %1.4f m/s'%wbar_manual)
```

```
using python averaging:
mean u = 2.8194 m/s
mean v = -0.0000 m/s
mean w = 0.0000 m/s
1800
using manual averaging:
mean u = 2.8194 m/s
mean v = -0.0000 m/s
mean w = 0.0000 m/s
```

$\overline{u}$ is the temporal average of u: \begin{eqnarray} \overline{u} &=& \frac{1}{1800} \sumf_{t=1}^{1800} u(t) \\ \end{eqnarray} same for v and w [Formula is required only for one of u, v, or w, but due to the orientation of the coordinate system (aligned into mean wind), both $\overline{v} = 0$ and $\overline{w} = 0$. Accept very small numbers that arise from numerical rounding errors.

Question 7:

From the data in the turbulence dataframe df2, calculate $\bar{u'^2}$, $\bar{v'^2}$, and $\bar{w'^2}$. Name and briefly define/describe these parameters and state what they are used to calculate. [4].

rubric: [1.5] values [2.5] discussion

In [45]:
```
df2.head()
```

Out[45]:

Date	u	v	w	Time
2000-08-02 15:00:00	3.2174	-0.5631	-0.4740	15:00:00
2000-08-02 15:00:01	3.1313	0.0400	0.0050	15:00:01
2000-08-02 15:00:02	3.7852	-0.4075	0.4388	15:00:02
2000-08-02 15:00:03	3.6670	-0.2518	0.0259	15:00:03
2000-08-02 15:00:04	3.9281	-0.1403	-0.2484	15:00:04

In [49]:
```
u_prime = df2.u - df2.u.mean()
v_prime = df2.v - df2.v.mean()
w_prime = df2.w - df2.w.mean()

up2 = [x ** 2 for x in u_prime]
vp2 = [x ** 2 for x in v_prime]
wp2 = [x ** 2 for x in w_prime]

up2_bar = np.sum(up2)/(1800)
vp2_bar = np.sum(vp2)/(1800)
wp2_bar = np.sum(wp2)/(1800)

print('the u variance is %1.3f m2/s2'%up2_bar)
print('the v variance is %1.3f m2/s2'%vp2_bar)
print('the w variance is %1.3f m2/s2'%wp2_bar)
```

```
the u variance is 0.265 m2/s2
the v variance is 0.204 m2/s2
the w variance is 0.075 m2/s2
```

Variances: Allow both, the biased (left) and the unbiased variance (right, makes no difference) and check for correct units: \begin{eqnarray} \overline{u^{\prime 2}} = \frac{1}{1800} \sum \limits_{t=1}^{1800} (u(t)-\overline{u})^{2} \hspace{0.5cm} \textrm{or} \hspace{0.5cm} \overline{u^{\prime 2}} = \frac{1}{1800-1} \sum \limits_{t=1}^{1800} (u(t)-\overline{u})^{2} \end{eqnarray}

Names: $\overline{u^{\prime 2}}$ is the variance of the longitudinal (also allow: horizontal) wind velocity, $\overline{v^{\prime 2}}$ is the variance of the lateral wind velocity. $\overline{w^{\prime 2}}$ is the variance of the vertical wind velocity [1.5]. These values are used to calculate the mean turbulent kinetic energy (TKE) [1]. [Total Marks: 4]

Question 8:

8. From the data in the turbulence dataframe df2, calculate the turbulence intensities Iu, Iv, and Iw. Briefly discuss what these values tell you. [4]

rubric: [1.5] values [2.5] discussion

In [50]:
```
# first we need the standard deviations:

sig_u = np.sqrt(up2_bar)
sig_v = np.sqrt(vp2_bar)
sig_w = np.sqrt(wp2_bar)

# Next we need the length (magnitude) of the mean wind vector, M = swrt(ubar

M = np.sqrt(ubar**2 + vbar**2 + wbar**2)

# Turbulence intensities are the dimensionless ratio between the standard de
# and the length of the mean wind vector M: I = sigma_u / M (or sigma_v, sig

Iu = sig_u / M
Iv = sig_v / M
Iw = sig_w / M

print('The turbulent intensities for u are %1.3f'%Iu)
print('The turbulent intensities for v are %1.3f'%Iv)
print('The turbulent intensities for w are %1.3f'%Iw)
```

```
The turbulent intensities for u are 0.183
The turbulent intensities for v are 0.160
The turbulent intensities for w are 0.097
```

In [ ]:

Question 9:

Turbulent kinetic energy:
Define turbulent kinetic energy, and write the equation used to calculate it. [2]

From the data provided, calculate the mean turbulent kinetic energy per unit mass $ \bar{e}$ [2].

What is the ratio of $\bar{e}$ to the mean kinetic energy per unit mass? [1]

Rubric:
a: [1] for equation, [1] for definition
b: [1] for approach, [1] for value
c: [1] for approach, [1] for value

In [56]:
```
TKE = 1/2 * (up2_bar + vp2_bar + wp2_bar)
print('TKE = %1.3f m2/s2'%TKE)

MKE = 1/2 * (ubar**2 + vbar**2 + wbar**2)
print('MKE = %1.3f m2/s2'%MKE)

ratio = TKE/MKE

print('The ratio of TKE to MKE is %1.3f (unitless)'%ratio)
```

```
TKE = 0.272 m2/s2
MKE = 3.974 m2/s2
The ratio of TKE to MKE is 0.068 (unitless)
```

In [ ]:

Turbulent kinetic energy is the average of the kinetic energy of the instantaneous deviations per unit mass [1] (vs mean kinetic energy, which is the energy in the mean flow rather than the deviations). The equation used to calculate it is [1]:

$$ \bar{e} = \frac{1}{2} \big(\overline{u'^2} + \overline{v'^2} + \overline{w'^2} \big) $$

Question 10:

Which of the three wind components, u, v or w, contains most turbulent kinetic energy per unit mass (in the dataframe df2)? Speculate about the shape of the eddies [2].

Rubric: [1] for answer [1] for discussion

From question 7, we can see the variance is largest for the u direction, so the most turbulent kinetic energy per unit mass is occuring as a result of the u component of the wind [1]. This means there is more tubulence horizontally than vertically (and in particular, in the horizontal u direction; for this dataset that means the east-west direction). The variance in the v direction is almost as big. The variance in the w direction is much smaller. Since the variances in the u and v direction are similar, and much larger than in the w direction, the eddies would have a pancake shape (and not a cigar or isotropic shape). This means flat-ish, circular-ish eddies.

In [ ]:

```

```